Online Election System

By

M. Imran Yaqub Malik
(242022-002)

Supervised By
Mr. Fazal Wahab

A project Submitted in partial fulfillment of the requirements for the degree of MCS

Department of Computer Sciences
Bahria Institute of Management and Computer Sciences (BIM&CS)

Bahria University Islamabad

CERTIFICATE

Certified that we accept the work contained in this report as a confirmation to the required standard for the partial fulfillment and found satisfactory for the requirements of the Degree.

Supervisor _____

Internal Examiner: _____

External Examiner: _____

Head of Department _____

Department of Computer Sciences

Bahria Institute of Management and Computer Sciences (BIM&CS)

Bahria University Islamabad

Dedication

Dedicated to my loving family

Table of Contents

Abstract

Online Election System (OES) is a computerized system to carry out secure Local Government Elections through Online Polling Stations connected to OES Central Web Server. All management tasks like Candidate Management, Voter Management, Employees Management, Polling Station lists Management, Voting Management, Result preparation etc. will be generated electronically using the system. This entire List will be approved by Election Commission Management. Employees will be assigned to particular designations at particular destinations. Employees will have logins and responsibilities, they are required to perform. These lists will be electronically distributed to the Computerized Polling Stations in a hierarchal order from Chief Election Commissioner till Presiding Officer.

Voter will be verified from NADRA web Services. A voter can cast its vote through OES Online System in voting booths. Votes from all the locations will be sent to a central server from their particular servers. Result will be compiled from the local servers and will be dynamically updated in central election commission office after specific intervals.

Acknowledgment

Innumerable thanks to **ALMIGHTY ALLAH** the creator and the sustainer of the universe for providing us with the abilities and prospect to complete this work..

All the words of gratitude and thankfulness fall incapacitated to express my deepest gratefulness to my supervisor Mr. Fazal Wahab for his continuous moral and professional support and guidance throughout project work. Discussions and meetings with him provided me very useful insight in this project.

I am grateful to all my honorable teachers for their efforts and valuable moral support provided during the entire course and my friends Mr. Humayun Aziz & Mr. Adil Malik who supported me a lot during the project.

Finally, I would also extend my gratitude to my Family for their continued prayers and encouragement

Project in Brief

Project Name:	Online Election System
Developed by:	Malik Imran
Supervised By:	Mr. Fazal Wahab
Degree:	MCS
Institute Name:	Bahria Institute of Management & Computer Sciences

CHAPTER 1

INTRODUCTION

1.1 Introduction

In the present area, the user of computer technology is increasing day by day. Different organizations have a computerized system to meet their objective and those who have manual system are taking decisions to develop such a system. Now lots of questions raised in mind that why those organizations feel to need of a computerized system and what are the reasons to develop such a new system. The answer of these questions is very simple that the organizations faced this problem due to the old manual system. Therefore in order to overcome such problems of manual system, the organizations are approaching computerized systems

1.2 About OES

OES (Online Election System) will facilitate both voters and election commission administration in their tasks. Voters will be able to cast their votes through computerized polling stations. Election commission will be able to get compiled result at any time. OES will be reliable and secure mean of conducting election as compare to the paper pencil election system. OES will work under real environment. Candidates and voters lists will be generated automatically using our system in election commission office by using given database. These lists will be electronically distributed to the computerized polling stations. A voter can cast its vote through computerized polling stations or through website. Votes from all the three sites will be sent to a central server from their particular servers. Result will be compiled from the central server and will be dynamically updated in central election commission office after specific intervals. Final results will be sent to all main polling stations and other specific locations. There will be also a module for reporting. The candidates and voters database management will be out of OES. We will assume that Areas and resources are already defined by election system and will be provided to OES in some electronic form.

OES (Online Election System) will facilitate both voters and election commission administration in their tasks. Voters will be able to cast their votes through computerized polling stations, phones and through Internet. Election commission will be able to get compiled result at any time. OES will be reliable and secure mean of conducting election as compare to the paper pencil election system.

1.3 Why OES & Advantages

To computerize the manual election system in order to provide maximum ease to voters and efficient, effective retrieval of voting results for election commission.

1.3.1 Customers and benefits

Primary: This software system specifically target to the Pakistan Election Commission as reference. It will provide them ease of management activities. Management will get quick and accurate voting results.

Secondary: An alternative for conducting elections in other government and private sectors.

It will give ease to voters to caste their votes.

1.3.2 Key factors used to judge quality

- Candidates and voters lists will be automatically generated and send to the appropriate polling stations.
- Electronic vote casting and result processing will enhance reliability and quality as compared to the manual system.
- Through manuals and training, the system developed will be easy to use for management of election commission. The system also facilitates the voter by providing more than one ways i.e. by internet and computerized polling stations, for vote casting.

1.3.3 Key features and technology

- Automatic candidate and voters list generation.
- To send these lists to different polling stations from one location automatically.
- Online vote casting.
- Vote casting through computerized polling systems.
- Dynamic update of results to a central location.
- Security and reliability in vote casting and in processing of result calculation.

- Reports generation.

1.3.4 Crucial product factors

- It will interact with old election system data base to get candidate and voter data to generate lists.
- Its design can be modified and can be grown according to the need.
- It will run in election office and in polling stations.
- Maintenance and proper training will be provided to the users of this system to make their understanding about the system.

CHAPTER 2

ANALYSIS

2.1 Introduction

Object Oriented Analysis is the foundation phase on which whole object-oriented software engineering depends. It should be carried out effectively and efficiently. The reason of object-oriented analysis is to model the real system so that it can be understood and to do this, we must examine requirements, analyze their implications, and restate them rigorously, and abstracting important real-world features first and defers small details until later. The successful analysis model state, "What must be done", without restricting how it is done, and avoid implementations decisions. The result of analysis should understand the problem as a preparation for design. Since we are documenting this whole phase so end result would be "ANALYSIS MODEL" or "REQUIREMENTS SPECIFICATION DOCUMENT"

Why Document All?

- Serves as a contract between the system user and the system developer
- Serves as a source of test plans
- Serves to specify projects goals and plan development cycles and increments

2.2 Step Involved (Each of them are reviewed against Customer Requirements)

i. Requirements Analysis
ii. Use Cases & Use Case Diagrams
iii. Class /Object Diagrams
iv. Interaction Diagrams (Sequence & Collaboration Diagrams)
v. State Diagrams representing system behavior
vi. Activity Diagrams describing sequencing of activities
vii. Deployment Diagrams (Physical Diagram)

2.3 Use of UML (Unified Modeling Language)

The UML, is the outcome of combination of the best features of Grady Booch, James Rumbaugh, and Ivar Jacobson Object Oriented Analysis & Design methods into a unified process, which went through a standardization process with the OMG (Object Management Group) and is now an OMG standard.

UML would allows to put across an analysis model using a modeling notation that is governed by a set of syntactic (tells us how symbols should look and combined), semantic (tells us what each symbol means) and pragmatic rules (defines intentions of symbols through which the purpose of the model is achieved and becomes understandable for others)

2.4 Five Different views for system Representation/Description:

2.4.1 User Model View:

This view represents the system/software from the end user's (action in UML) perspective and is defined by a set of USE CASES and USE CASE DIAGRAMS.

2.4.2 Structure Model view:

Data and functionality are viewed from inside the system, that is static structure (classes, object and relationships) is modeled.

2.4.3 Behavior Model view:

Represent the dynamic or behavior aspects of system and depicts the interactions or collaborations between various structural elements described in the user and structural models.

2.4.4 Implementation Model view

The structural and behavioral aspects of the system are represented as they are to be built

2.4.5 Environment Model view

The structural and behavioral aspects of environment in which the system is to be implemented are represented.

2.5 Requirement Analysis:

2.5.1 Introduction:

Prior to requirements analysis, modeling or specifying they must be gathered through an Elicitation process. “Elicitation Process” is one by which customers needs are understood and documented. Again express “what” is to be built and not ”how” it is to be built.

C and D Requirements

C Customer requirements and needs; articulated in language understood by the customer

Types of Customer (C) Requirements:

- **Normal Requirements:**
  Requirements demanded or stated by customers directly / explicitly.

- **Expected Requirements:**
  Implicit requirements and may be so fundamental that the customer doesn’t explicitly state them and their absence will cause for significant dissatisfaction. e.g. ease of human/machine interaction, overall operational correctness and reliability, and ease of software installation etc.

- **Exciting Requirements:**
  These features go beyond customer’s expectations and prove to be very satisfying when present.

D For the developers, may be more formal.

Types of Developer (D) Requirements:

- **Functional Requirements:**
  The requirements stated by developers that are directly linked to the functions being performed by their system/software to be developed.

- **Non- Functional Requirements:**
  The requirements that are not directly related with system to be built functionalities e.g. Performance, Reliability, Constraints etc.

2.5.2 Roadmap:

- Classify Customer and End-user of the system
- Interview with customer representative (Detail Analysis of Existing System/Environment)
- After Information analysis of existing system, write requirements of customer, examination with customer and update when it is necessary.
- Developer requirements to be written, check and to make sure that there is no discrepancy between the requirements of the system customer and developer.

2.6 High Level System Components

2.6.1 Management

- Upload Candidate Forms on the OES Server
- Add Candidate Parties
- Add Symbols
- Assign Symbol To Candidates
- Create Candidate List

2.6.2 Employee Management

- Create Employee List
- Assign Employee to various Designations at various Locations Like(DRO in Districts, RO in Town-Tehsil , PO in Union Councils ,Polling Stations)

2.6.3 Voter Management

- Verify Voters from NADRA
- Create Voter List
- Approve Voter List
- Send Voter List From CEC down to Polling Stations

2.6.4 Polling Station Management

- Create List Of Polling Stations in All Union Councils
- Approve Polling Station List
- Send Polling Station List

2.6.5 Result Management

- Get Result From All Polling Stations
- Combine Result
- View Result
- Send Result from Polling Stations to all Upper Levels Automatically after specific interval of time

2.6.6 Voting Management

- Select candidate and cast vote online.

2.6.7 Security Management

- Every System user has his specific rights & View in OES System.
- All the System Users much pass through Secure Login Wall.
- Outdoor Network Traffic wraps in separate layer through Encryption & Decryption techniques.
- Voter is verified before Voting through their NIC.
- Once Voter has cast his vote his status will be disabled from both online

2.6.8 Web Services NADRA

- This Web service is *assumed* to be provided by NADRA through which OES verifies that either NIC# provided by Voter is valid or not.

2.7 Summary of Requirements

Each Assistant Chief Election Commissioner (ACEC) prepares lists of voters of his province. Chief Election Commissioner (CEC) receives these lists from all ACECs. CEC proves the lists for further processing. CEC proves the lists of District Returning Officers (DROs) and Returning Officers (ROs) prepared by ACEC. Each ACEC prepares the lists of DROs for all districts under his province. Each DRO prepares the lists of ROs for all Towns/Tehsil under his district. Each RO prepares the lists of Presiding Officers, Assistant Presiding Officers and of polling staffs. CEC distributes the lists of voters to ACEC. Each ACEC distributes the lists to DROs of his province. Each DRO distributes the lists to ROs of his district. Each RO distributes the lists to POs of all polling stations in a town/tehsil. RO prepares the lists of candidates and allocate them symbol. DRO approves those lists. Result will be updated automatically during polling from PO to RO and from RO to DRO and to CEC. At the end of polling, PO will compile the result and will send unofficial result to DRO. DRO will validate the result and will send to CEC for approval.

2.8 High Level Usecase Diagram

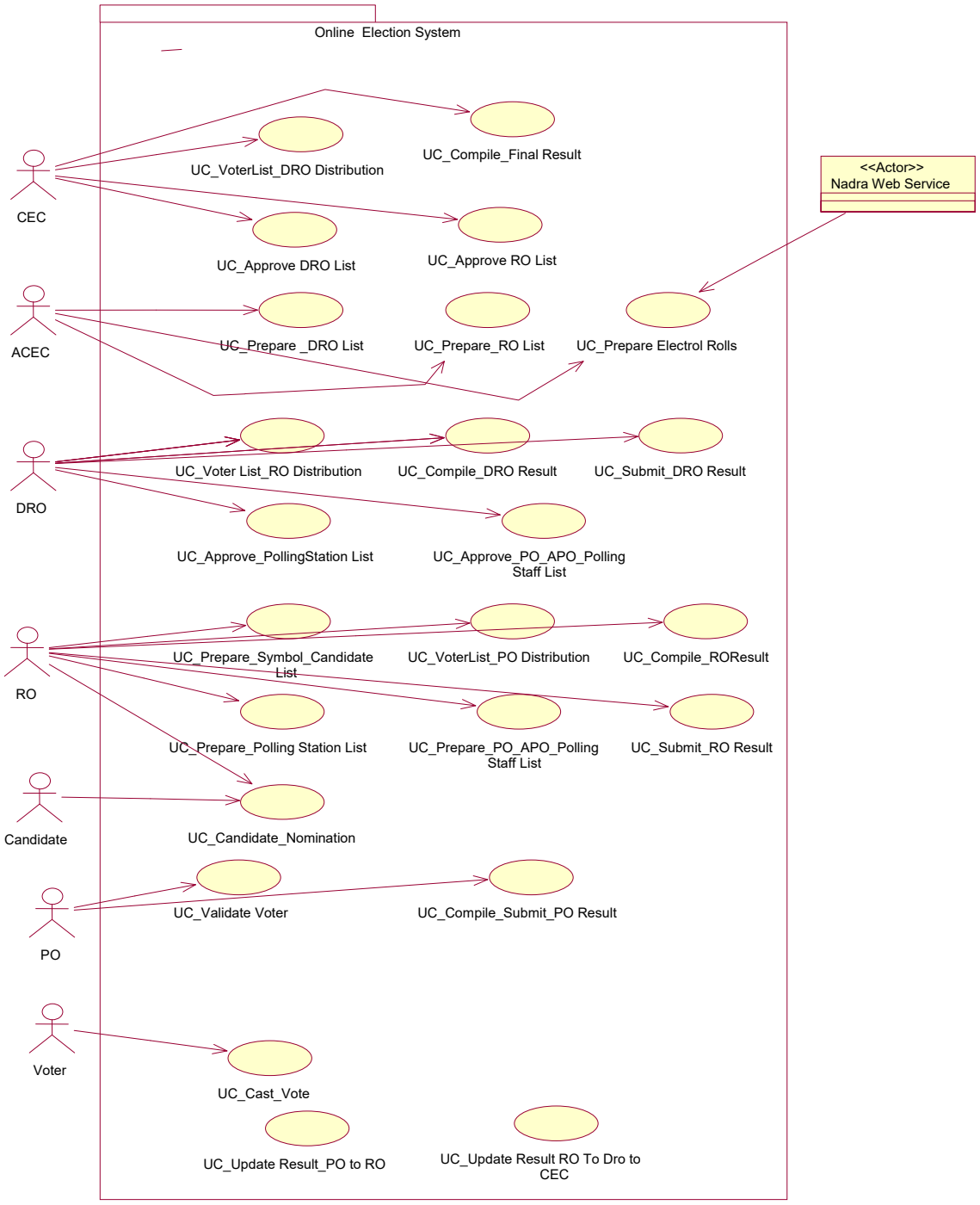


Figure: 2.8 High Level use case diagram

2.9 Analysis Level Use case Diagram

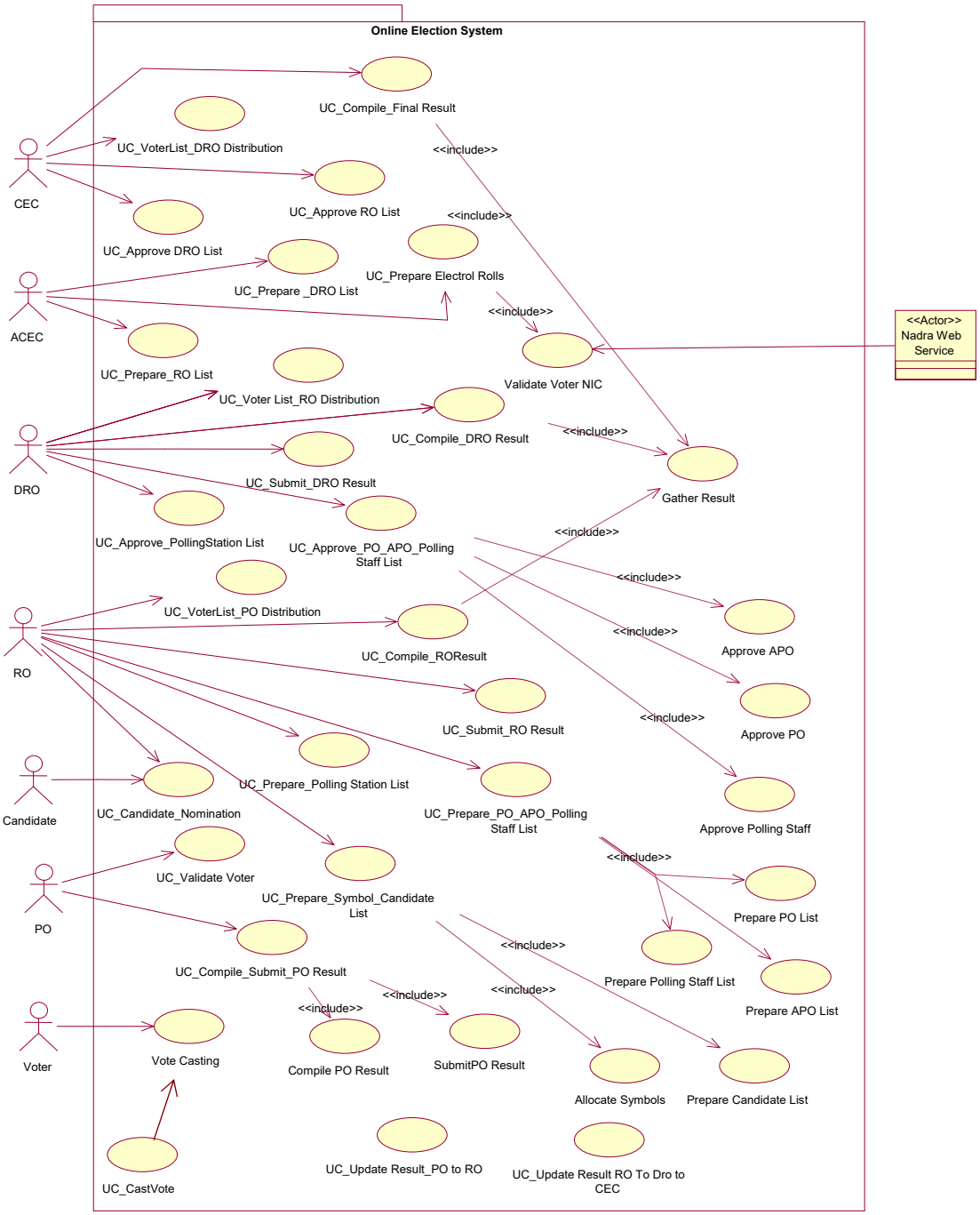


Figure: 2.9 Analysis use case diagram

2.10 Use Case Description

Use Case Id	**Use Case Name**
UC_1	UC_Prepare_ElectorlRoll
Primary Actor:	Assistant Chief Election Commission (ACEC)
Brief Description:	ACEC enters the data. The data is validated and saved into the system. After entering all the data, electoral rolls are generated from the system.
Preconditions:	ACEC is identified and authenticated. Voter's data entry screen is presented to ACEC for data entry.
Basic Flow:	1. ACEC enters voter's information. 2. System verifies the voter's NIC from web service provided by NADRA. 3. ACEC confirms saving. (ACEC repeats steps 1,2,3,4 until all data is entered) 4. System generates and presents the electoral rolls lists.
Post condition:	Data of a voter will be saved and a message of confirmation will be shown to the user. Voter verification information is also saved.
Alternate Flow:	1a. Invalid Information: System signals error and asks for correct entry. 2a. Invalid voter: User will be flagged as invalid user.

Use Case Id	**Use Case Name**
UC_2	UC_Candidate_Nomination
Primary Actor:	Returning Officer (RO) ,Candidate
Brief Description:	After the publication of Election Schedule by the Election Commission, nomination papers are invited from interested contesting candidates.
Preconditions:	Returning Officer (RO) is identified and authenticated for particular district. Nomination form is available to the candidates from election commission official website or gets the Nomination Form from RO office.

Basic Flow:	1. Candidate downloads the nomination form from Election Commission website. 2. Candidate fills in the form. 3. Attached the required documentation. e.g. educational background, NIC photocopy, picture etc 4. Candidate submits the nomination form along with the required documents back to RO. 5. RO accepts/rejects the nomination papers of candidate.
Post condition:	At the end of this use case the nominated candidate will be collected by the Returning Officer and symbols assign to qualified candidates.
Alternate Flow:	1a. Nomination form is not available at Election Commission site System shows message to the candidate Candidate can get nomination form from RO office. 1b. Election Commission web server is down. System signals error to the candidate Candidate can get nomination form from RO office.

Use Case Id	**Use Case Name**
UC_3	UC_Prepare_Symbol_CandidateList
Primary Actor:	Returning Officer(RO)
Brief Description:	Inspection of nomination papers is carried out by the Returning Officers and nomination papers are either accepted or rejected on the basis of set rules by RO.
Preconditions:	Returning Officer (RO) is identified and authenticated for particular district. Nomination forms of candidates are available to RO. RO rejects the candidate who does not fulfill the set criteria. List of Election Symbols approved by the Election Commission is available to RO.
Basic Flow:	1. RO enters the id and name of selected candidate. 2. RO assigns the symbol to selected candidate according to his party affiliation. 3. RO assigns the symbol to individual candidates if there is any. 4. RO prepares a list of selected candidates with symbols

	5. RO publishes the list of selected candidates with symbols.
Post condition:	Final list of contesting candidates is prepared and published in the prescribed manner by the Returning Officer after incorporation of the decisions on appeals and after withdrawal of candidature by the candidates if any.
Alternate Flow:	3b. Db server is down System signals error to the candidate

Use Case Id	**Use Case Name**
UC_4	UC_VoterList_DRODistribution
Primary Actor:	Chief Election Commission (CEC)
Brief Description:	CEC will distribute the voters' lists to specific District Regional Officers. CEC will do this by uploading the lists on Web server.
Preconditions:	Data about Districts and DROs is already entered. CEC is identified and authenticated. Voters' lists distribution screen is presented to CEC for distribution.
Basic Flow:	1. CEC selects or enters District name. 2. System shows basic information of district. 3. CEC selects or enters DRO ID from the list. 4. System shows basic information of DRO. 5. CEC selects the path of the Voter Lists of the specified district. 6. CEC confirms for uploading the list. (CEC repeats steps 1, 2, 3, 4, 5, 6 until all voters' lists are uploaded.) 7. System generates and presents the lists distribution information.

Post condition:	Voters' lists are uploaded on server. Distribution information is saved in database.
Alternate Flow:	1a. Invalid District Name: System signals error and asks for correct entry. 3a. Invalid DRO ID: System signals error and asks for correct entry. 5a. Invalid List Path: System signals error and asks for correct selection.

Use Case Id	**Use Case Name**
UC_5	UC_VoterList_RODistribution
Primary Actor:	District Returning Officer (DRO)
Brief Description:	DRO will distribute the voters' lists to Regional Officers. DRO will do this by uploading the lists on Web server.
Preconditions:	DRO has received lists from CEC. Data about ROs is already entered. DRO is identified and authenticated. Voters' lists distribution screen is presented to DRO for distribution.
Basic Flow:	1. DRO selects or enters District name. 2. System shows Towns/Tehsils under that district. 3. DRO selects or enters Town/Tehsil name. 4. System shows basic information of Town/Tehsil. 5. DRO selects or enters RO ID from the list. 6. System shows basic information of RO. 7. DRO selects the path of the Voter Lists of the specified Town/Tehsil. 8. DRO confirms for uploading the list. 9. (DRO repeats steps 1, 2, 3, 4, 5, 6, 7, 8 until all voters' lists are uploaded.) 10. System generates and presents the lists distribution information.

Post condition:	Voters' lists are uploaded on server. Distribution information is saved in database.
Alternate Flow:	1a. Invalid District Name: DRO can select or can enter his own district. System signals error and asks for correct entry. 3a. Invalid Town/Tehsil Name: DRO can select or can enter Town/Tehsil from the list. System signals error and asks for correct entry. 5a. Invalid RO ID: System signals error and asks for correct entry. 7a. Invalid List Path: System signals error and asks for correct selection.

Use Case Id	**Use Case Name**
UC_6	UC_VoterList_PODistribution
Primary Actor:	Returning Officer (RO)
Brief Description:	RO will distribute the voters' lists to Presiding Officers to all polling stations under his Town/Tehsil. RO will do this by uploading the lists on Web server.
Preconditions:	RO has received lists from DRO. Data about POs is already entered. RO is identified and authenticated. Voters' lists distribution screen is presented to RO for distribution.

Basic Flow:	1. RO selects or enters Town/Tehsil name. 2. System shows basic information of Town/Tehsil and Polling station names. 3. RO selects or enters Polling station name. 4. System shows basic information of Polling station and PO name. 5. RO selects the location of the voter list of the specified Polling station. 6. RO confirms for uploading the list. 7. (RO repeats steps 1, 3, 4, 5, 6 until all voters' lists are uploaded.) 8. System generates and presents the lists distribution information.
Post condition:	Voters' lists are uploaded on server. Distribution information is saved in database.
Alternate Flow:	1a. Invalid Town/Tehsil Name: 1. System signals error and asks for correct entry. 3a. Invalid Polling station name: System signals error and asks for correct entry. 5a. Invalid List Path: System signals error and asks for correct selection.

Use Case Id	**Use Case Name**
UC_7	UC_Validate_Voter
Primary Actor:	Presiding Officer (PO)
Brief Description:	PO will validate voter on poling station on the basis of his/her NIC number.
Preconditions:	1. PO is already login. 2. Voter lists are available for polling station. 3. Voter must come with his/her own NIC.
Basic Flow:	1. PO will enter the NIC# in appropriate field 2. System will validate voter from registered polling station voters list. 3. System will check voter status to be true. 4. PO will allocate polling booth to voter.
Post condition:	Registered authenticated voters will only access polling booth to cast vote.
Alternate Flow:	1a. PO will enter wrong data System will show error message

	System allows him to reenter again. 2a. If voter is not registered in polling station System will show Unregistered Voter message. 3a. If voter status is FALSE. 1. System will show Duplicate Vote message.

Use Case Id	**Use Case Name**
UC_8	UC_Cast_Vote
Primary Actor:	Voter
Brief Description:	This use case is related to vote casting by voter at polling station, which is computerized and connected with server on a network.
Preconditions:	Voter must enter Polling Station along with his NIC Voter must be validated by the PO PO must allocate a separate polling both to the Voter
Basic Flow:	1. Voter will enter the polling both allocated by PO. 2. Voter will click or will press Enter button to start voting. 3. Voter will select his favorite Candidate and press NEXT button. 4. Voter will repeat step 3 until all categories are fulfilled 5. Voter will press DONE button to end vote casting 6. System will prepare final ballet paper showing his selected candidates of each category for confirmation. 7. Voter will press OK button to successfully casting his vote. 8. Voter status for vote casting will be disabled on the basis of NIC to prohibit multiple vote casts from single voter on polling station either from polling station or cell phone.
Post condition:	System will save voter vote in local DATABASE of polling station. A successful vote casting message will be show to voter.

	After a specific time local DATABASE data will be send to upper level servers.
Alternate Flow:	4a. Voter can press Back button or select any category to go back Voter will change candidate for selected category Voter can repeat step 4a until to visit all previous categories. 5a. Voter can press REFRESH button to recycle for a fresh vote. 5b. Voter can press EXIT button to end Voting. 7a. Voter will press CANCEL button to go back and continue voting.

Use Case Id	**Use Case Name**
UC_9	UC_Compile_Submit_POResult
Primary Actor:	Presiding Officer(PO)
Brief Description:	Polling Station results are carried out by the Presiding Officer (PO) at the end of polling after the given time of polling. A report of results is generated for polling station. The results send to concerned Returning Officer (RO).
Preconditions:	Voting is completed. Presiding Officer (PO) is identified and authenticated for particular polling station.
Basic Flow:	1. Presiding Officer(PO) starts compiling results 2. Results are calculated according to sum formula defined by Election Commission. 3. Presiding Officer prepares a statement of the count indicating the number of votes secured by a candidate 4. A compiled results report of particular polling station is generated. 5. PO sends the report to RO.
Post condition:	The result displays locally at polling station. A report is send to concerned RO by PO

Use Case Id	**Use Case Name**

UC_10	UC_Compile_ROResult
Primary Actor:	Returning Officer (RO)
Brief Description:	RO receives results from all Polling stations under his Town/Tehsil. RO will compile the result for further processing. Compiled result will be shown in a Report form. This result will be of a single Town or Tehsil.
Preconditions:	All polling stations under a town or tehsil has uploaded result on the web server under their town or tehsil. Compilation screen is presented to RO for compiling result.
Basic Flow:	1. RO downloads the results of all polling stations under his Town/Tehsil from the web server 2. RO apply validation checks to validate the result of a polling station. 3. System checks the validation of result. 4. RO clicks OK button for confirmation. 5. (RO repeats steps 2, 3, 4 until all results of all polling stations are validated.) 6. System converts the results into a single document under the name of Town/Tehsil of RO. 7. System shows the compiled result of a Town/Tehsil in a report format.
Post condition:	Validation criteria are saved. Compiled result is saved. Validation information is saved.
Alternate Flow:	1a. Invalid or broken download link RO sends message to specific PO for uploading the result again. 3a. Invalid Result: System signals error and saved the result in invalid results list.

Use Case Id	**Use Case Name**
UC_11	UC_Submit_ROResult
Primary Actor:	Returning Officer (RO)
Brief Description:	All ROs submit their Towns or Tehsils compiled unofficial results to DRO.

Preconditions:	RO has compiled and prepared the unofficial result. Result uploading screen is presented to RO for submitting result to DRO.
Basic Flow:	1. RO selects or enters the District name. 2. System shows basic information about the district. 3. RO selects or enters his Town/Tehsil name. 4. System shows basic information about the Town/Tehsil. 5. RO selects or enters DRO ID. 6. System shows basic information about the DRO. 7. RO enters some description about the result. 8. RO selects the path of compiled result document. 9. RO confirms for uploading the result to DRO. 10. System shows confirmation message for successful uploading.
Post condition:	Uploading information is saved. Result is uploaded on the web server.
Alternate Flow:	1a. Invalid District name: System signals error and asks for correct entry. 3a. Invalid Town/Tehsil name: System signals error and asks for correct entry. 5a. Invalid DRO ID: System signals error and asks for correct entry. 8a. Invalid Result document Path: System signals error and asks for correct document path selection.

Use Case Id	**Use Case Name**
UC_12	UC_Compile_DROResult
Primary Actor:	District Returning Officer (DRO)
Brief Description:	DRO receives results from all Returning Officers under his district. DRO will compile the result for further processing. Compiled result will be shown in a Report form. This result will be of a single district.

Preconditions:	All returning officers have uploaded results of their towns or Tehsils on the web server. Compilation screen is presented to DRO for compiling result.
Basic Flow:	1. DRO downloads the results of all ROs under his district from the web server. 2. DRO apply validation checks to validate the result of a Town/Tehsil. 3. System checks the validation of result. 4. DRO clicks OK button for confirmation. 5. (DRO repeats steps 2, 3, 4 until all results of all Towns/Tehsils under his districts are validated.) 6. System converts the results into a single document under the name of district of DRO. 7. System shows the compiled result of a district in a report format.
Post condition:	Validation criteria are saved. Compiled result is saved. Validation information is saved.
Alternate Flow:	1a. Invalid or broken download link DRO sends message to specific RO for uploading the result again. 3a. Invalid Result: System signals error and saved the result in invalid results list.

Use Case Id	**Use Case Name**
UC_13	UC_Submit_DROResult
Primary Actor:	District Returning Officer (DRO)
Brief Description:	All DROs submit their districts compiled unofficial results to Assistant Chief Election Commission (ACEC) of specific province and Chief Election Commission (CEC).
Preconditions:	DRO has compiled and prepared the unofficial result. Result uploading screen is presented to DRO for submitting result to ACEC and CEC.
Basic Flow:	1. System shows the name of CEC. 2. DRO selects or enters ACEC ID of his Province. 3. System shows basic information about the Province. 4. DRO selects or enters the District name. 5. System shows basic information about the district.

	6. DRO enters some description about the result. 7. DRO selects the path of compiled result document. 8. DRO confirms to upload the result to ACEC and CEC. 9. System shows confirmation message for successful uploading.
Post condition:	Uploading information is saved. Result is uploaded on the web server.
Alternate Flow:	2a. Invalid ACEC ID: System signals error and asks for correct entry. 4a. Invalid district name: System signals error and asks for correct entry. 6a. Invalid Result document Path: System signals error and asks for correct document path selection.

Use Case Id	**Use Case Name**
UC_14	UC_Compile_FinalResult
Primary Actor:	Chief Election Commission (CEC)
Brief Description:	Chief Election Commission (CEC) compiles and approves the final result.
Preconditions:	All DROs have uploaded results of their districts on the web server. Compilation screen is presented to CEC for compiling result.
Basic Flow:	1. CEC downloads the results of all DROs under all districts. 2. CEC apply validation checks to validate the result of a district. 3. System checks the validation of result. 4. CEC clicks OK button for confirmation. 5. (CEC repeats steps 2, 3, 4 until all results of all districts are validated.) 6. System converts the results into a single document under the name Final result. 7. System shows the Final compiled result in a report format. 8. CEC approves the results.

Post condition:	Validation criteria are saved. Compiled result is saved. Validation information is saved.
Alternate Flow:	1a. Invalid or broken download link DRO sends message to specific RO for uploading the result again. 3a. Invalid Result: System signals error and saved the result in invalid results list.

Use Case Id	**Use Case Name**
UC_15	UC_Prepare_DROList
Primary Actor:	Assistant Chief Election Commission (ACEC)
Brief Description:	ACEC prepares the lists of selected District Returning officers of all districts.
Preconditions:	ACEC is identified and authenticated. Data of districts and DROs is already entered.
Basic Flow:	1. ACEC selects or enters the District Name. 2. System shows information of district. 3. ACEC selects the DRO ID. 4. System shows information of DRO. 5. ACEC enters the start date of the job. 6. ACEC enters the end date of the job. 7. ACEC confirms the appointment. 8. (ACEC repeats steps 1, 3, 5, 6, 7 until all DROs are appointed to their districts.) 9. System generates the list of all the appointed DROs.
Post condition:	Appointed list of DROs are saved
Alternate Flow:	1a. Invalid District name: System signals error and asks for correct entry. 3a. Invalid DRO ID: System signals error and asks for correct entry. 5a. Invalid Date: System signals error and asks for correct entry. 6a. Invalid Date: System signals error and asks for correct entry.

Use Case Id	Use Case Name
UC_16	UC_Approve_DROList

Primary Actor:	Assistant Chief Election Commissioner (ACEC), Chief Election Commissioner (CEC)
Brief Description:	ACEC submits the lists of selected DRO to CEC. CEC receives the lists. CEC applies some validation criteria and approves after validating the lists.
Preconditions:	CEC and ACEC are identified and authenticated. Lists of District Returning Officers are prepared.
Basic Flow:	ACEC selects the path of list of District Returning Officers of a province. ACEC enters some description about the lists. ACEC confirms for uploading the list to CEC. System shows confirmation message for successful uploading. CEC receives list. CEC applies validation criteria. CEC approves the list.
Post condition:	Uploading information is saved. Valid and Invalid information is saved.
Alternate Flow:	1a. Invalid lists Path: System signals error and asks for correct document path selection. 5a. Invalid or broken download link CEC sends message to specific ACEC for uploading the lists again.

Use Case Id	Use Case Name
UC_17	UC_Prepare_ROList

Primary Actor:	Assistant Chief Election Commission (ACEC)
Brief Description:	ACEC prepares the lists of selected Returning officers of all Towns/Tehsils under each district.
Preconditions:	ACEC is identified and authenticated. Data of districts, Towns/Tehsils and ROs is already entered.

Basic Flow:	1. ACEC selects or enters the District Name. 2. System shows Town/ Tehsil ID and name. 3. ACEC selects or enters the Town/Tehsil ID. 4. System shows basic information about Town/Tehsil. 5. ACEC selects the RO ID. 6. System shows information of RO. 7. ACEC enters the start date of the job. 8. ACEC enters the end date of the job. 9. ACEC confirms the appointment. 10. (ACEC repeats steps 1, 3, 5, 7, 8, 9 until all ROs are appointed to their Town/Tehsil.) 11. System generates the list of all the appointed ROs.
Post condition:	Appointed list of ROs are saved

Use Case Id	**Use Case Name**
UC_18	UC_Approve_ROList
Primary Actor:	Assistant Chief Election Commissioner (ACEC), Chief Election Commissioner (CEC)
Brief Description:	ACEC submits the lists of selected RO to CEC. CEC receives the lists. CEC applies some validation criteria and approves after validating the lists.
Preconditions:	CEC and ACEC are identified and authenticated. Lists of all Returning Officers are prepared.
Basic Flow:	1. ACEC selects or enters the District name. 2. System shows basic information about the district. 3. ACEC selects the path of list of Returning Officers of a district. 4. ACEC enters some description about the list. 5. ACEC confirms for uploading the list to CEC. 6. System shows confirmation message for successful uploading. 7. (ACEC repeats steps 1, 3, 4, 5 until lists of all ROs of all districts are uploaded.) 8. CEC receives list. 9. CEC applies validation criteria. 10. CEC approves the list.

Post condition:	Uploading information is saved. Valid and Invalid information is saved.
Alternate Flow:	1a. Invalid District name: System signals error and asks for correct entry. 3a. Invalid lists Path: System signals error and asks for correct document path selection. 7a. Invalid or broken download link CEC sends message to specific ACEC for uploading the lists again.

Use Case Id	**Use Case Name**
UC_19	UC_Prepare_PollingStationList
Primary Actor:	Returning Officer (RO)
Brief Description:	RO prepares the list of polling stations under his Town/Tehsil for approval from DRO.
Preconditions:	RO is identified and authenticated. Data of districts, Towns/Tehsils and polling stations is already entered.
Basic Flow:	1. RO selects or enters the District Name. 2. System shows Town/ Tehsil ID and name. 3. RO selects or enters the Town/Tehsil ID. 4. System shows basic information about Town/Tehsil. 5. RO enters information about a polling station. 6. RO confirms the entry of polling station 7. (RO repeats steps 5, 6 until all polling stations under a Town/Tehsil are created.) 8. System generates the list of all the polling stations under the Town/Tehsil.
Post condition:	Polling stations assignment list is saved.
Alternate Flow:	1a. Invalid District name: System signals error and asks for correct entry. 3a. Invalid Town/Tehsil name: System signals error and asks for correct entry. 5a. Wrong entry: System signals error and asks for correct entry.

Use Case Id	Use Case Name
UC_20	UC_Approve_PollingStationList

Primary Actor:	District Returning Officer (DRO), Returning Officer (RO)
Brief Description:	RO submits the lists of polling stations to DRO. DRO receives the lists. DRO applies some validation criteria and approves after validating the lists.
Preconditions:	RO and DRO are identified and authenticated. Lists of polling stations are prepared.
Basic Flow:	1. RO selects or enters the District name. 2. System shows basic information about the district. 3. RO selects or enters his Town/Tehsil name. 4. System shows basic information about the Town/Tehsil. 5. RO selects or enters DRO ID. 6. System shows basic information about the DRO. 7. RO selects the path of list of polling stations under a Town/Tehsil. 8. RO enters some description about the lists. 9. RO confirms for uploading the result to DRO. 10. System shows confirmation message for successful uploading. 11. DRO receives list. 12. DRO applies validation criteria. 13. DRO approves the list.
Post condition:	Uploading information is saved. Valid and Invalid information is saved.
Alternate Flow:	1a. Invalid District name: System signals error and asks for correct entry. 3a. Invalid Town/Tehsil name: System signals error and asks for correct entry. 5a. Invalid DRO ID: System signals error and asks for correct entry. 7a. Invalid lists Path: System signals error and asks for correct document path selection.

Use Case Id	Use Case Name
UC_21	UC_Prepare_PO_APO_PollingStaffList

Primary Actor:	Returning Officer (RO)
Brief Description:	RO prepares the list of Presiding officers, Assistant Presiding Officers and Polling staff of all polling stations under his Town/Tehsil for approval from DRO.
Preconditions:	RO is identified and authenticated. Data of districts, Towns/Tehsils, Polling Stations, Presiding officers (PO), Assistant Presiding officers (APO) is already entered.
Basic Flow:	1. RO selects or enters the District Name. 2. System shows Town/ Tehsil ID and name. 3. RO selects or enters the Town/Tehsil ID. 4. System shows basic information about Town/Tehsil. 5. RO selects or enters the Polling station ID. 6. System shows information of Polling station. 7. RO selects or enters the PO ID. 8. System shows information of PO. 9. RO selects or enters the APO ID. 10. System shows information of APO. 11. RO selects or enters the Polling staff IDs. 12. System shows information of polling staff information. 13. RO confirms the assignment. 14. (RO repeats steps 5, 7, 9, 11, 13 until all PO, APO and polling staff are appointed to all polling stations.) 15. System generates the list of all the assigned PO, APO and polling staffs.
Post condition:	All POs, APOs and polling staff list are saved.
Alternate Flow:	1a. Invalid District name: System signals error and asks for correct entry. 3a. Invalid Town/Tehsil name: System signals error and asks for correct entry. 5a. Invalid Polling station ID: System signals error and asks for correct entry. 7a. Invalid PO ID: System signals error and asks for correct entry. 9a. Invalid APO ID: System signals error and asks for correct entry. 11a. Invalid Polling staff ID: System signals error and asks for correct entry.

Use Case Id	Use Case Name
UC_22	UC_Approve_PO_APO_PollingStaffList

Primary Actor:	District Returning Officer (DRO), Returning Officer (RO)
Brief Description:	RO submits the lists of POs, APOs, Polling staff to DRO. DRO receives the lists. DRO applies some validation criteria and approves after validating the lists.
Preconditions:	RO and DRO are identified and authenticated. Lists of POs, APOs, polling staff are prepared.
Basic Flow:	1. RO selects or enters the District name. 2. System shows basic information about the district. 3. RO selects or enters his Town/Tehsil name. 4. System shows basic information about the Town/Tehsil. 5. RO selects or enters DRO ID. 6. System shows basic information about the DRO. 7. RO selects the path of lists of POs, APOs, polling staff under a polling station. 8. RO enters some description about the lists. 9. RO confirms for uploading the lists to DRO. 10. (RO repeats steps 7, 8, 9 until all PO, APO and polling staff lists are uploaded.) 11. System shows confirmation message for successful uploading. 12. DRO receives lists. 13. DRO applies validation criteria. 14. DRO approves the lists.
Post condition:	Uploading information is saved. Valid and Invalid information is saved.
Alternate Flow:	1a. Invalid District name: System signals error and asks for correct entry. 3a. Invalid Town/Tehsil name: System signals error and asks for correct entry. 5a. Invalid DRO ID: System signals error and asks for correct entry. 7a. Invalid lists Path: System signals error and asks for correct document path selection. 11a. Invalid or broken download link

	DRO sends message to specific RO for uploading the lists again.

Use Case Id	**Use Case Name**
UC_23	UC_UpdateResultPOToRO
Primary Actor:	System
Brief Description:	System at every PO level will send the result to specific RO system after a specific time. On RO side system will update database and will show the updating of results dynamically.
Preconditions:	System is in running state. Votes are being cast on Polling stations.
Basic Flow:	1. System checks the updating time. 2. System prepares a result for updating to RO data base. 3. System attaches some information of Polling station, DRO etc with the result. 4. System sends result to RO system. 5. RO System receives the results from each polling station after an interval and update in database. 6. RO system shows the updating result on screen.
Post condition:	Updating result is saved in RO database. Changes are recorded in database.
Alternate Flow:	2a. No new vote is cast in specific time: System will send only a message to RO system. 4a. Result sending problem: System internally saves the problem and asks user for specific settings.

Use Case Id	**Use Case Name**
UC_24	UC_UpdateResultROToDROCEC

Primary Actor:	System
Brief Description:	System at every RO level will send the result to specific DRO system and to CEC system after a specific time. On RO side system will update the result and will show the updating of results dynamically.
Preconditions:	System is in running state. Results have been updated on ROs levels.
Basic Flow:	1. System checks the updating time. 2. System prepares a result for updating to specific DRO and CEC data bases. 3. System attaches some information with the result. 4. System sends result to specific DRO and CEC systems. 5. Specific DRO and CEC System receive the results from each RO after an interval and update in databases. 6. Specific DRO and CEC system shows the updating result on screen.
Post condition:	Updating result is saved in Specific DRO and CEC databases. Changes are recorded in database.
Alternate Flow:	2a. No new result is updated on RO system. System will send only a message to DRO and CEC systems. 4a. Result sending problem: System internally saves the problems and asks user for specific settings.

2.11 Domain Models

2.11.1 Domain Model of UC_Prepare_ElectorlRoll (UC_1)

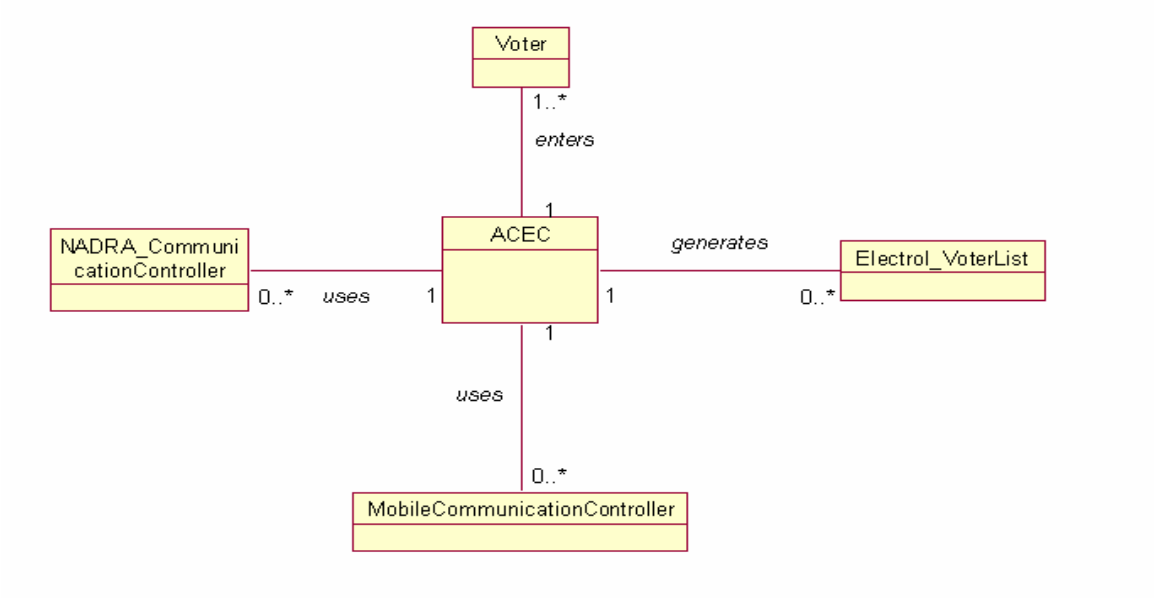


Figure 2.11.1

2.11.2 Domain Model of UC_Candidate_Nomination (UC_2)

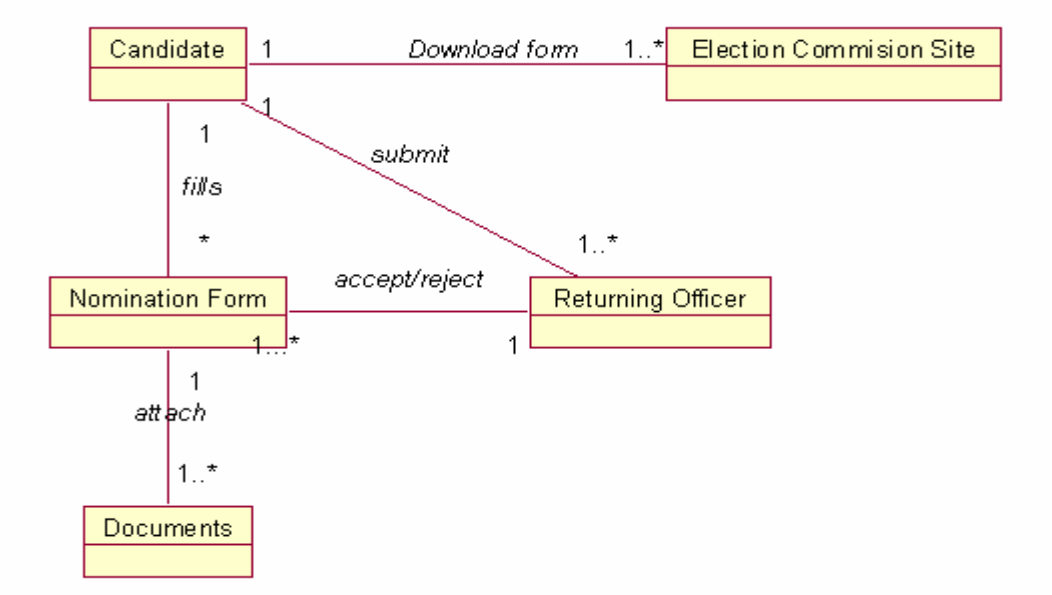


Figure 2.11.2

2.11.3 Domain Model of UC_Prepare_Symbol_CandidateList (UC_3)

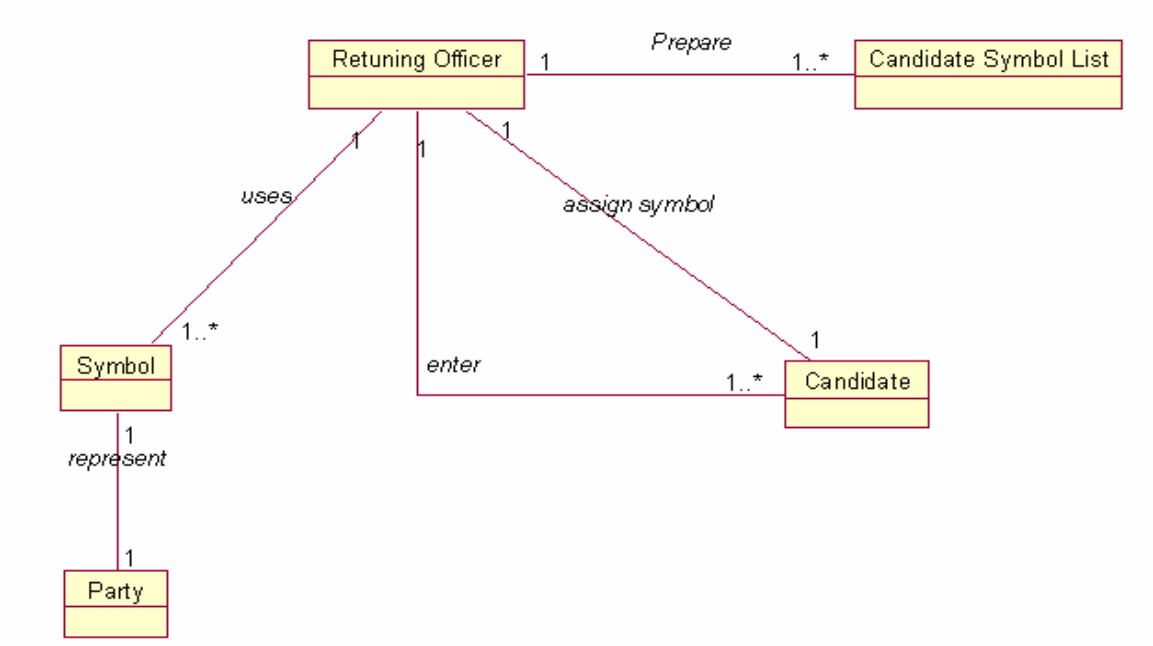


Figure 2.11.3

2.11.4 Domain Model of UC_VoterList_DRODistribution (UC_4)

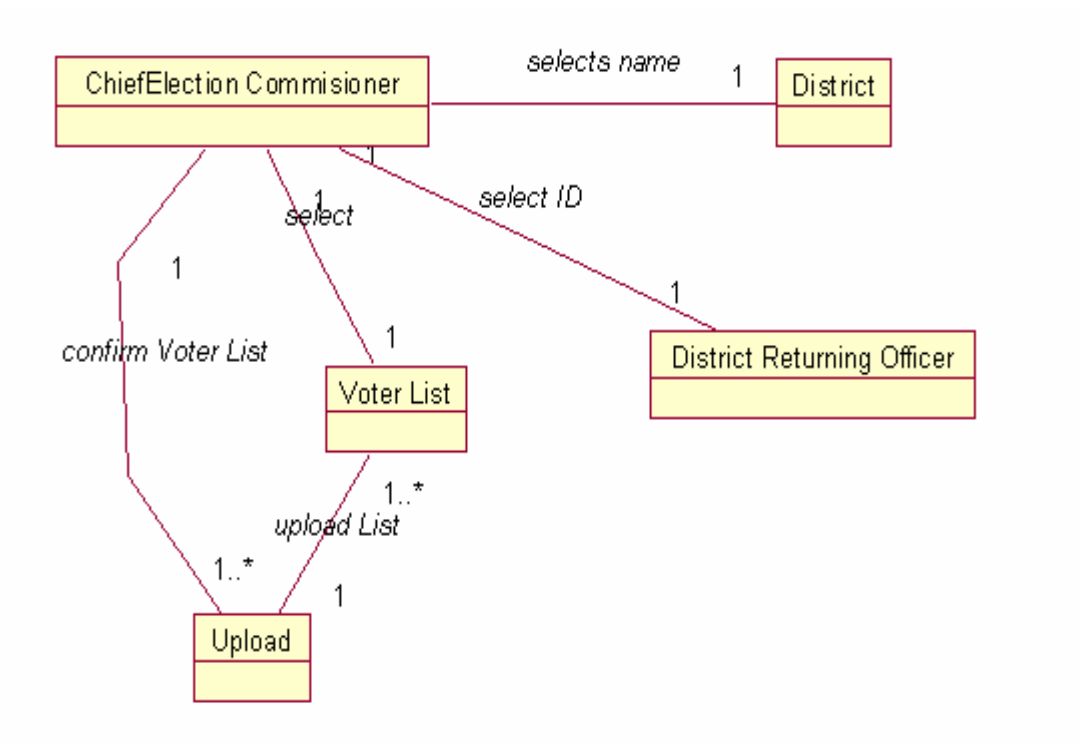


Figure 2.11.4: Domain Mode

2.11.5 Domain Model of UC_VoterList_RODistribution (UC_5)

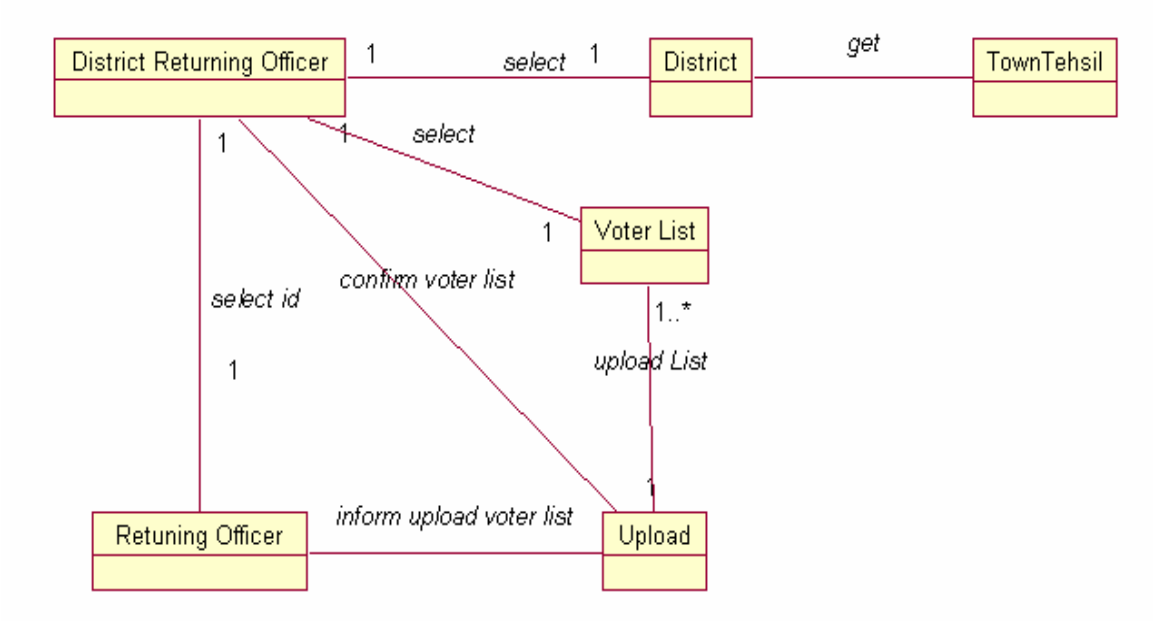


Figure 2.11.5

2.11.6 Domain Model of UC_VoterList_PODistribution (UC_6)

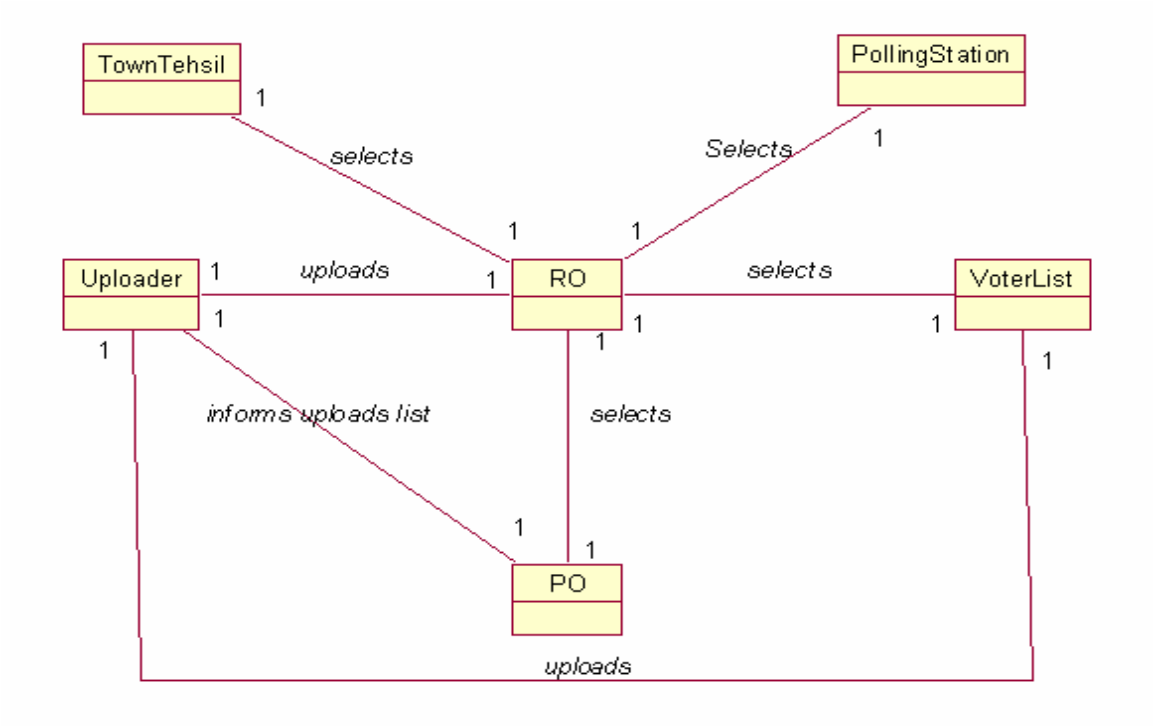


Figure 2.11.6

2.11.7 Domain Model of UC_Validate_Voter (UC_7)

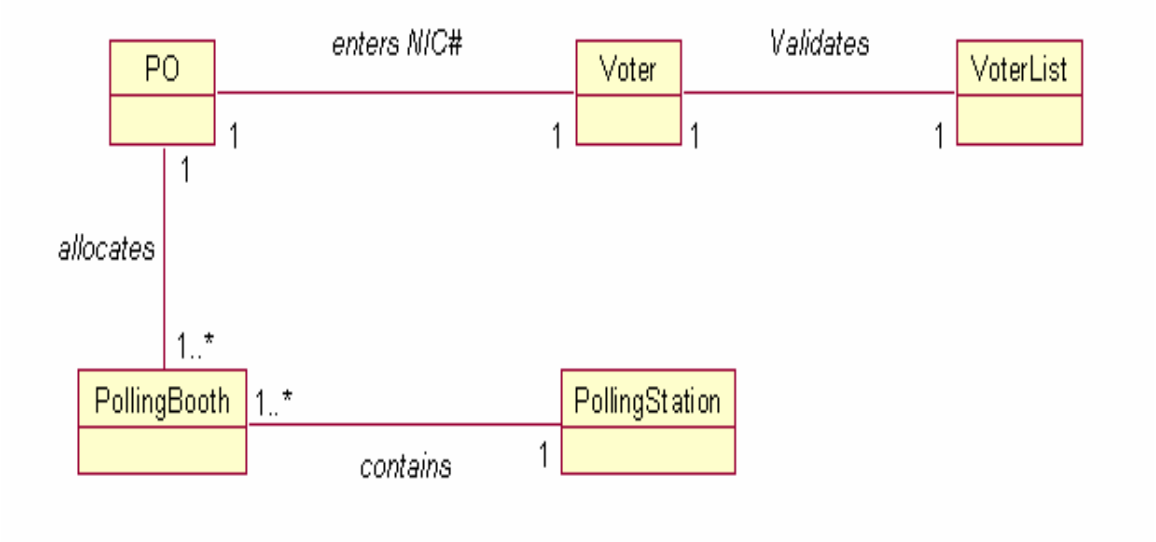


Figure 2.11.7:

2.11.8 Domain Model of UC_Cast_Vote (UC_8)

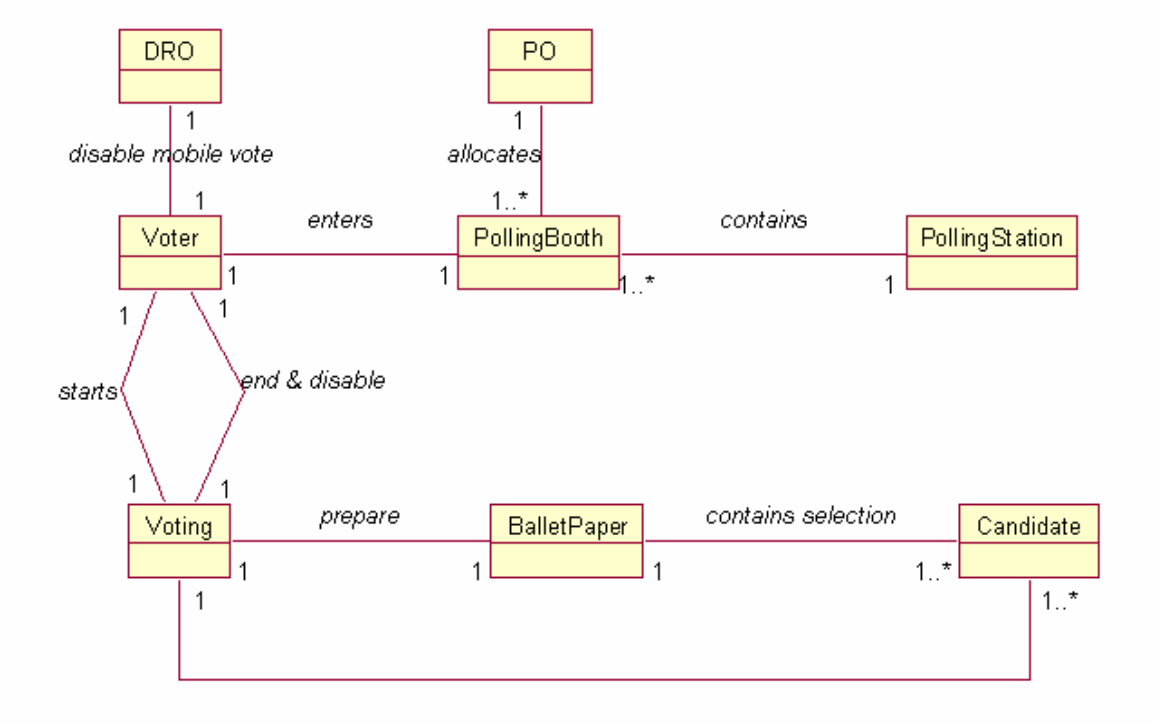


Figure 2.11.8

2.11.9 Domain Model of UC_Compile_Submit_POResult (UC_9)

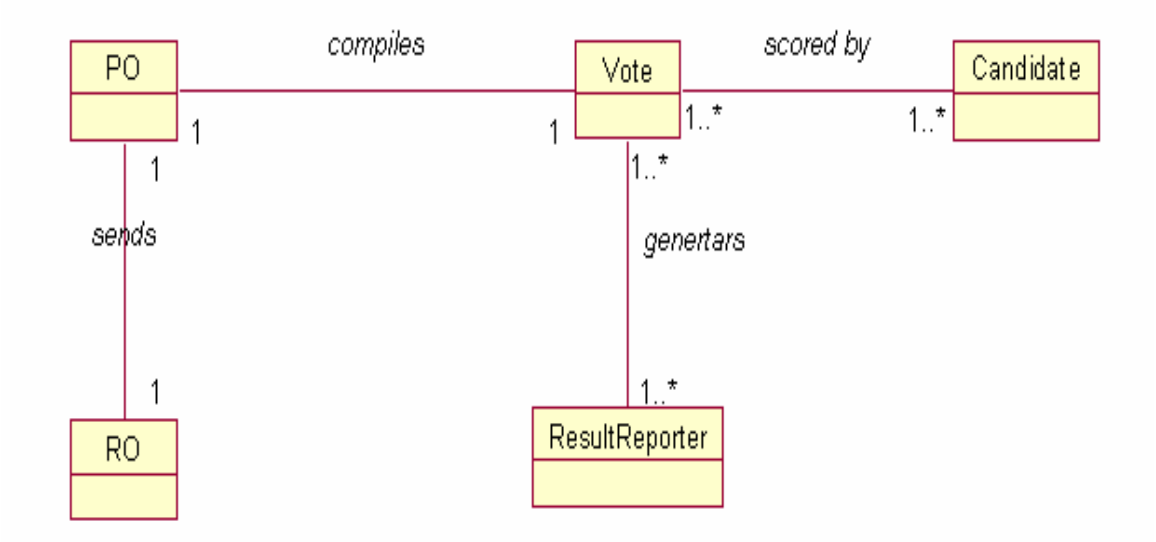


Figure 2.11.9

2.11.10 Domain Model of UC_Compile_ROResult (UC_10)

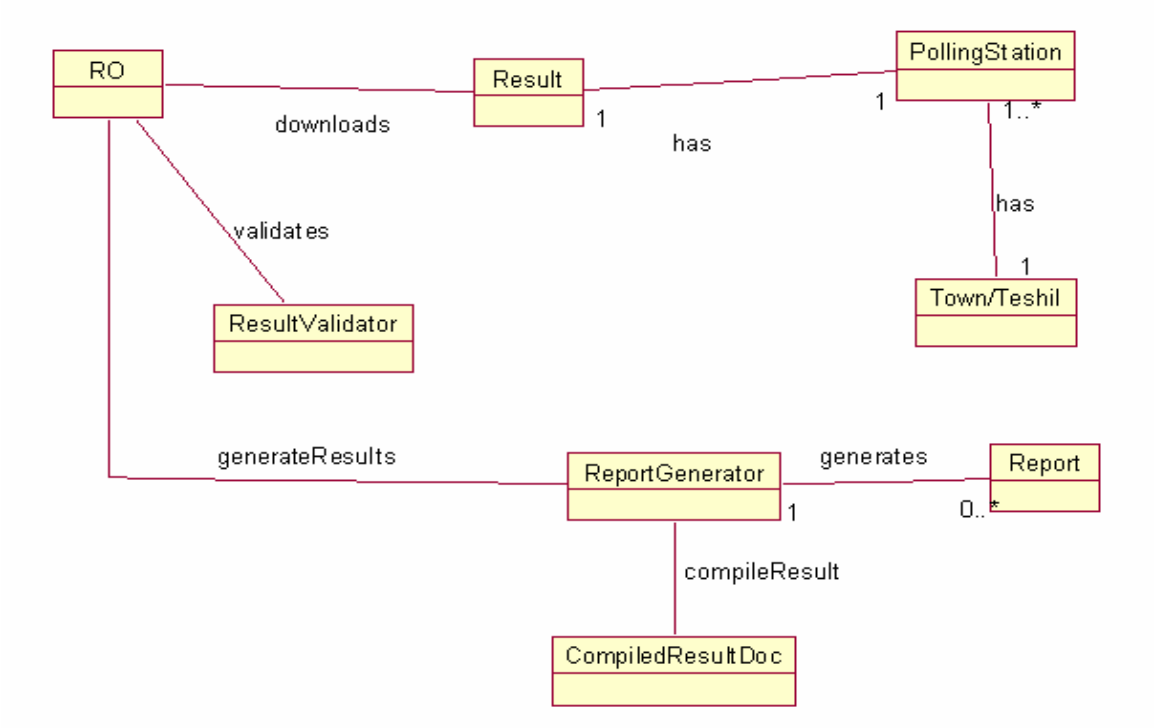


Figure 2.11.10

2.11.11 Domain Model of UC_Submit_ROResult (UC_11)

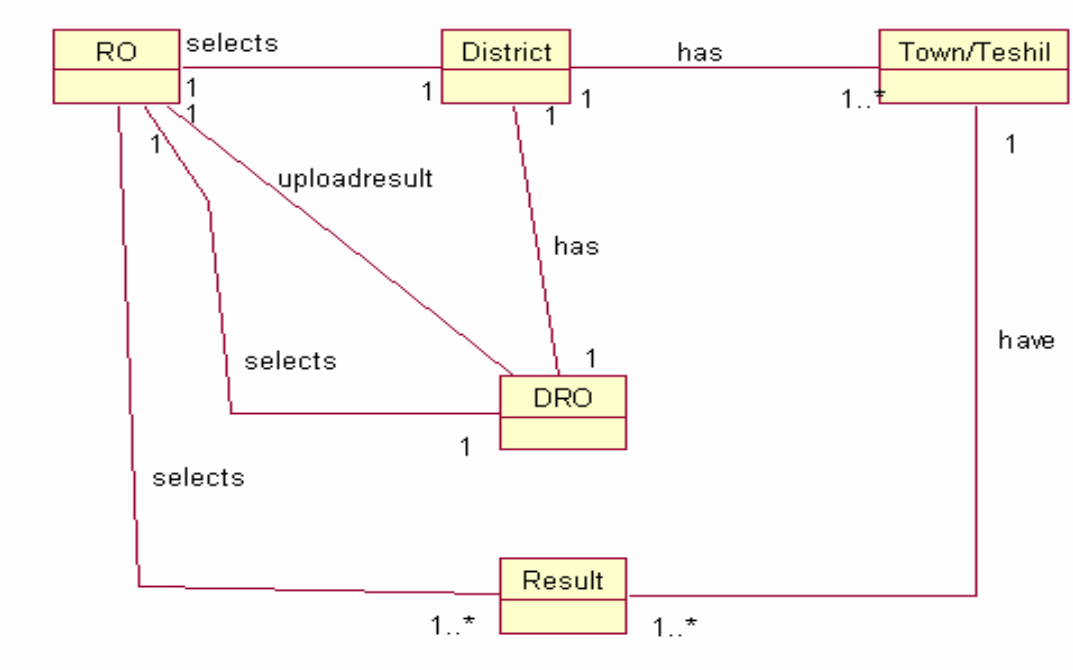


Figure 2.11.11

2.11.12 Domain Model of UC_Compile_DROResult (UC_12)

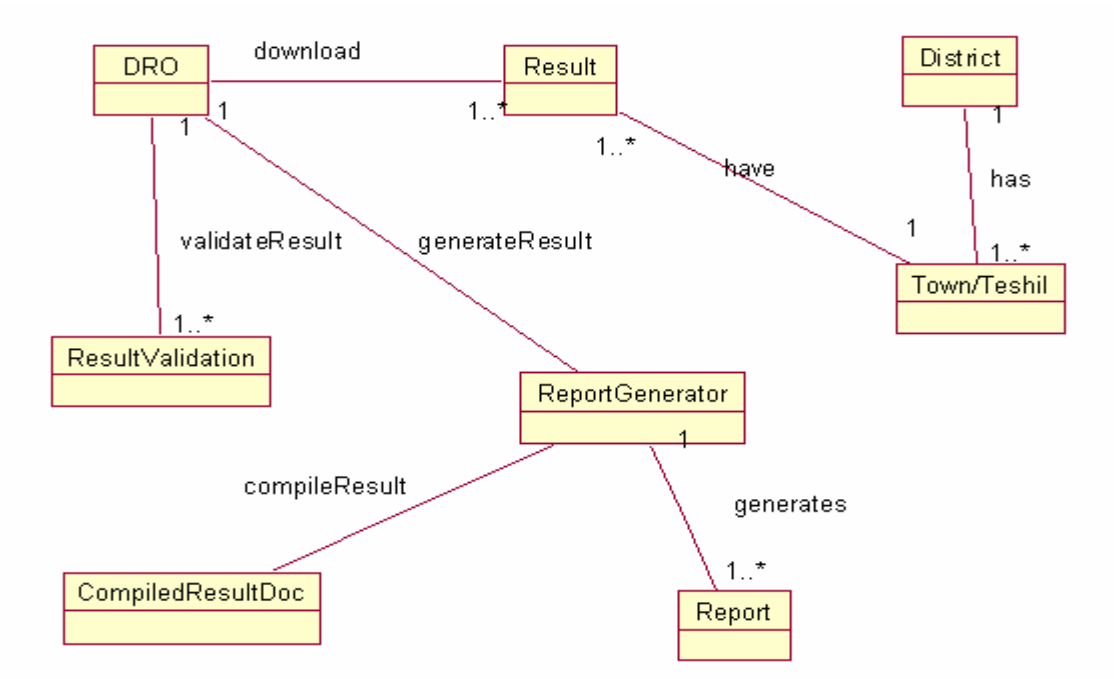


Figure 2.11.12

2.11.13 Domain Model of UC_Submit_DROResult (UC_13)

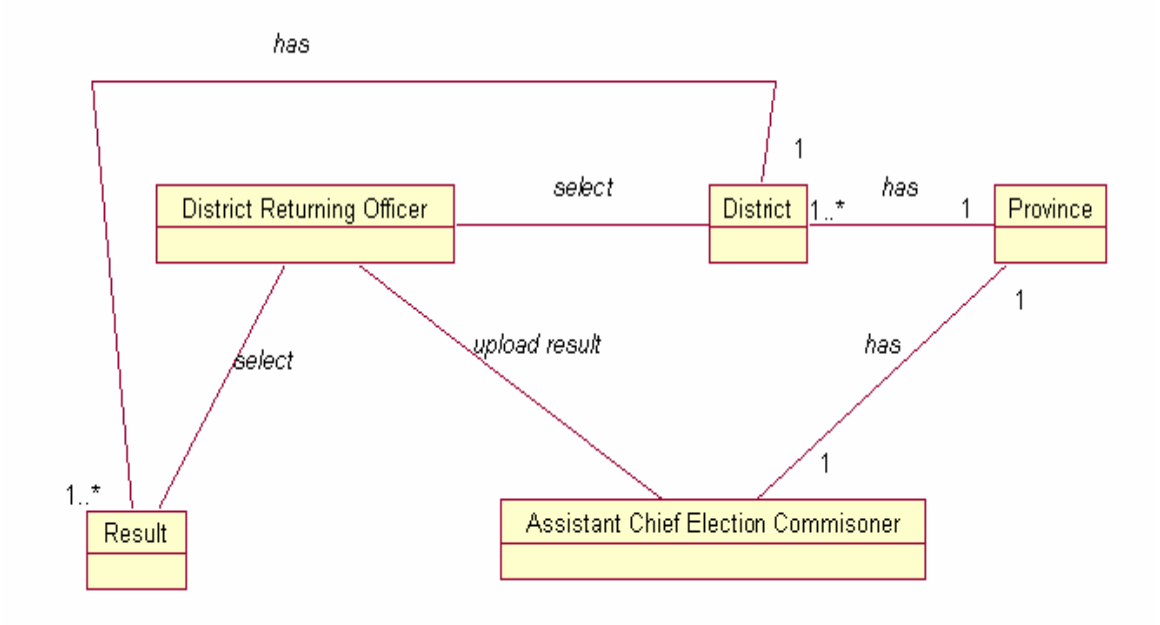


Figure 2.11.13

2.11.14 Domain Model of UC_Compile_FinalResult (UC_14)

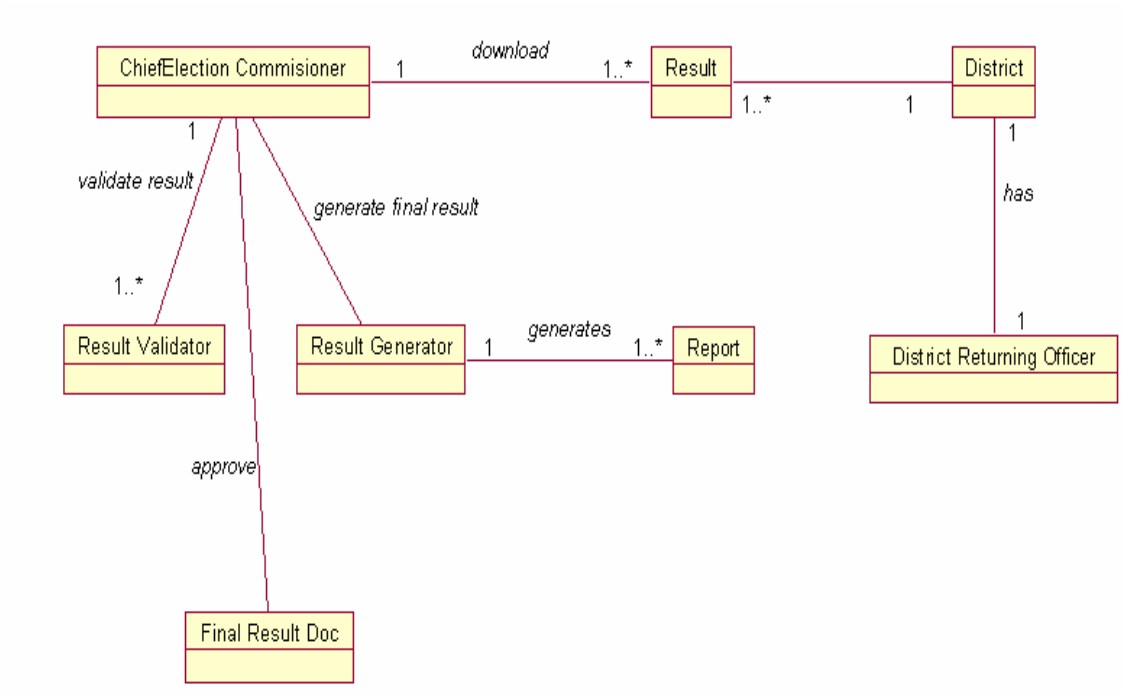


Figure 2.11.14

2.11.15 Domain Model of UC_Prepare_DROList (UC_15)

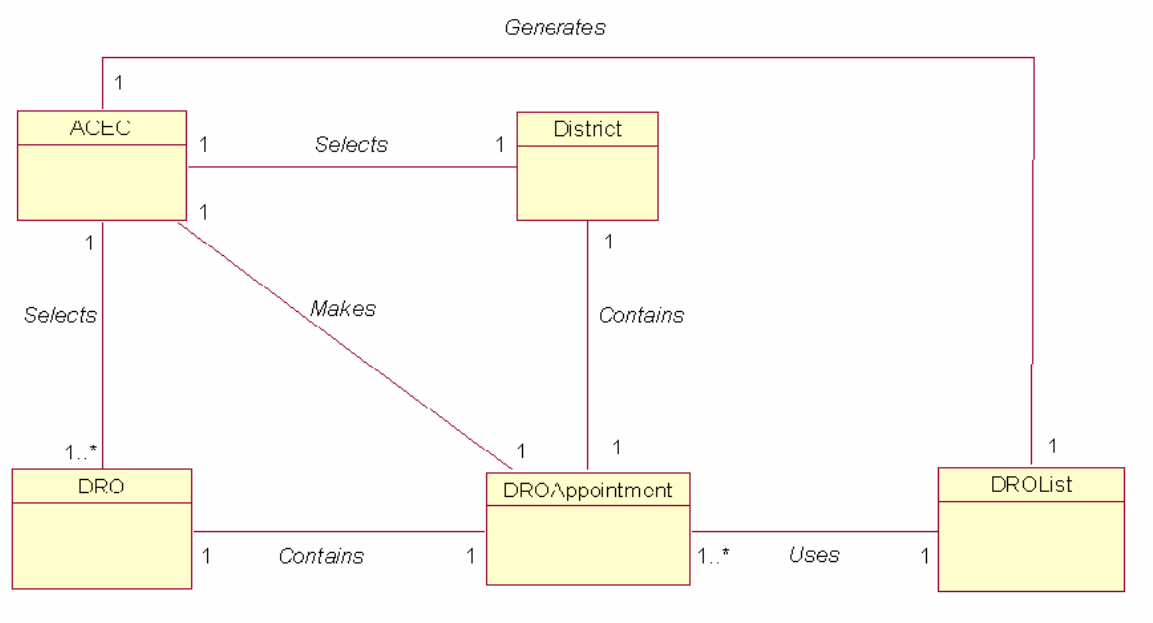


Figure 2.11.15

2.11.16 Domain Model of UC_Approve_DROList (UC_16)

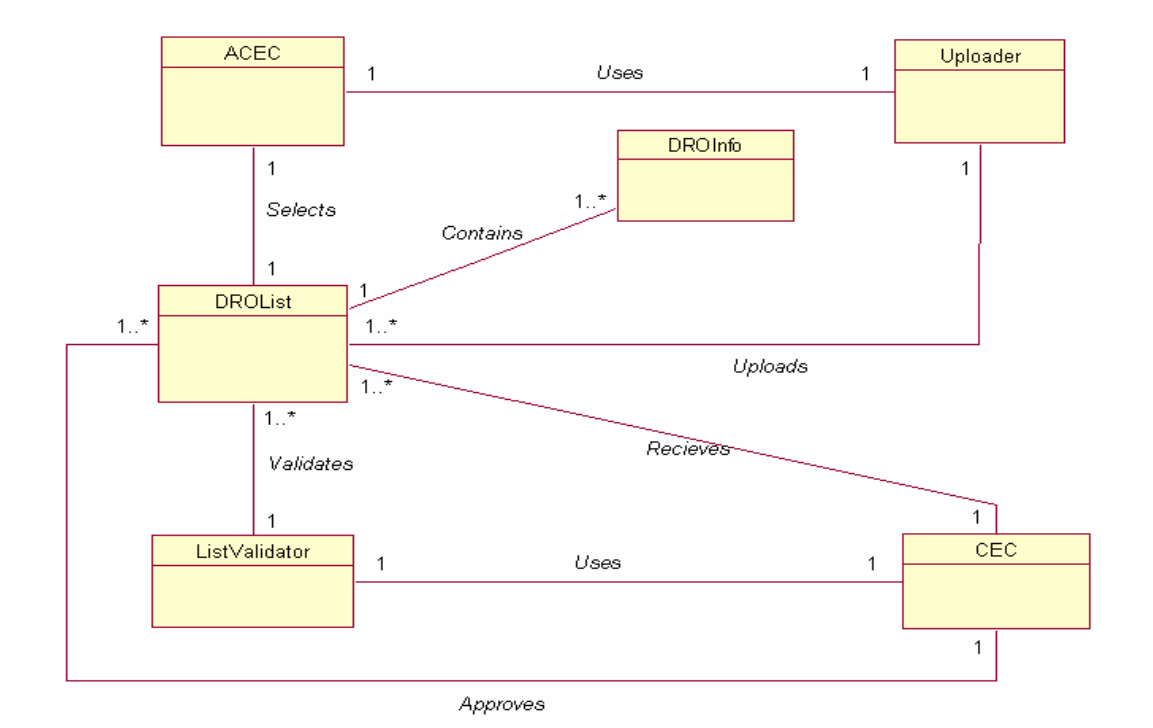


Figure 2.11.16

2.11.17 Domain Model of UC_Prepare_ROList (UC_17)

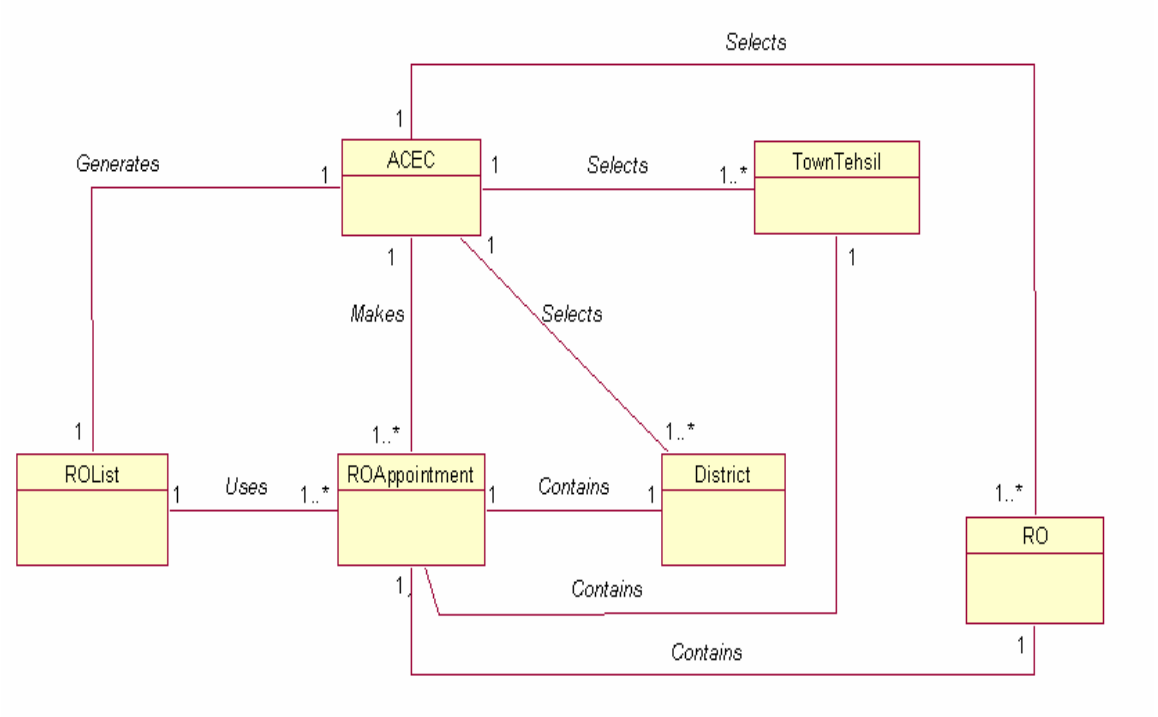


Figure 2.11.17

2.11.18 Domain Model of UC_Approve_ROList (UC_18)

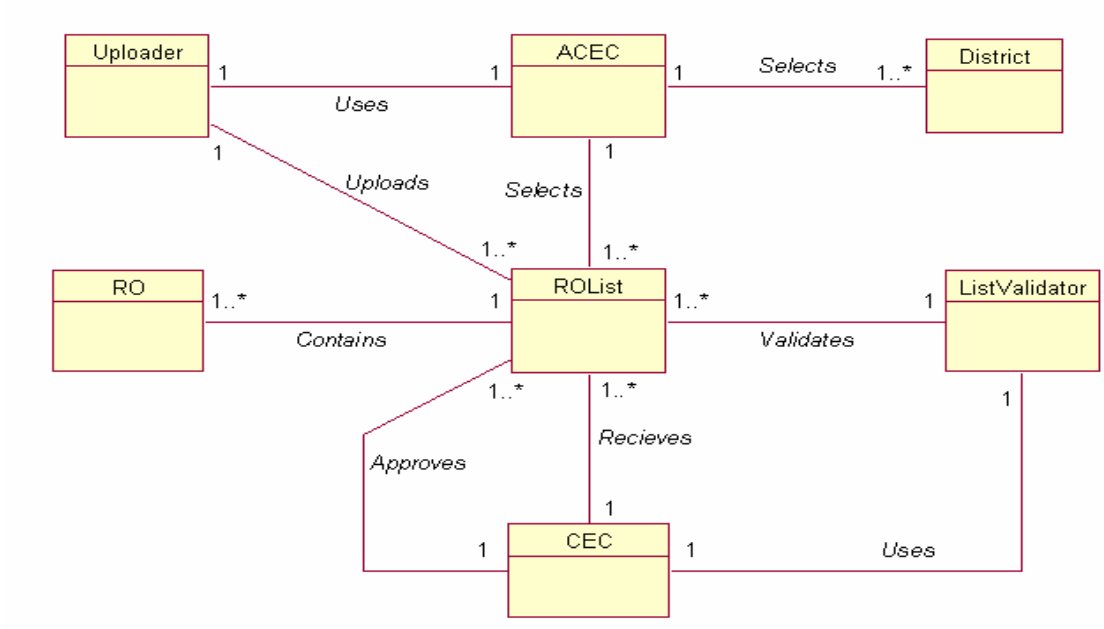


Figure 2.11.18

2.11.19 Domain Model of UC_Prepare_PollingStationList (UC_19)

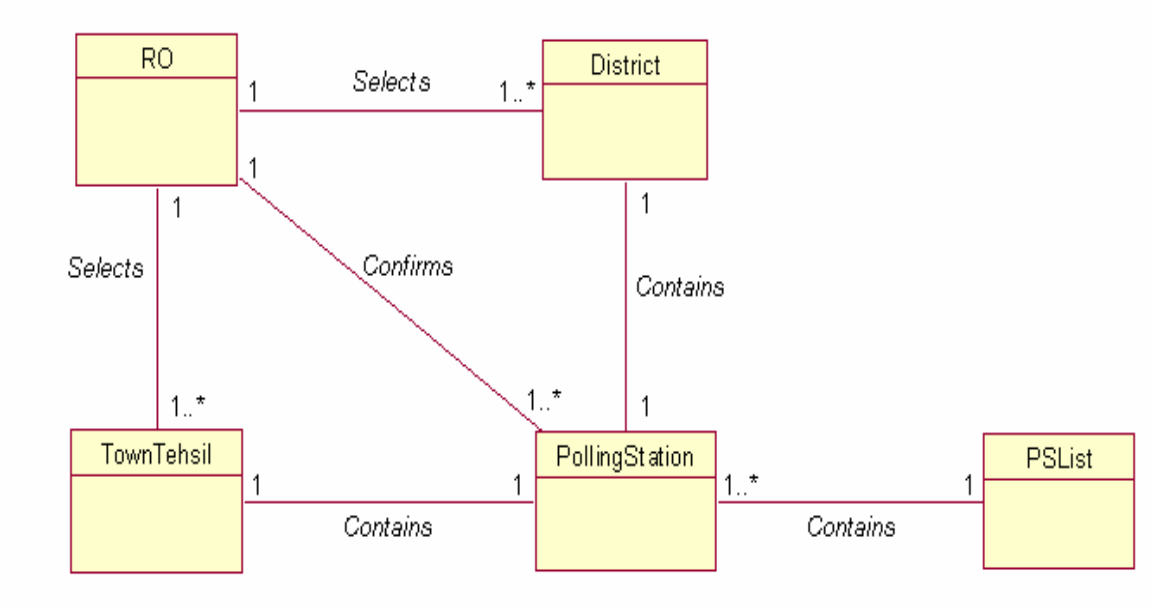


Figure 2.11.19

2.11.20 Domain Model of UC_Approve_PollingStationList (UC_20)

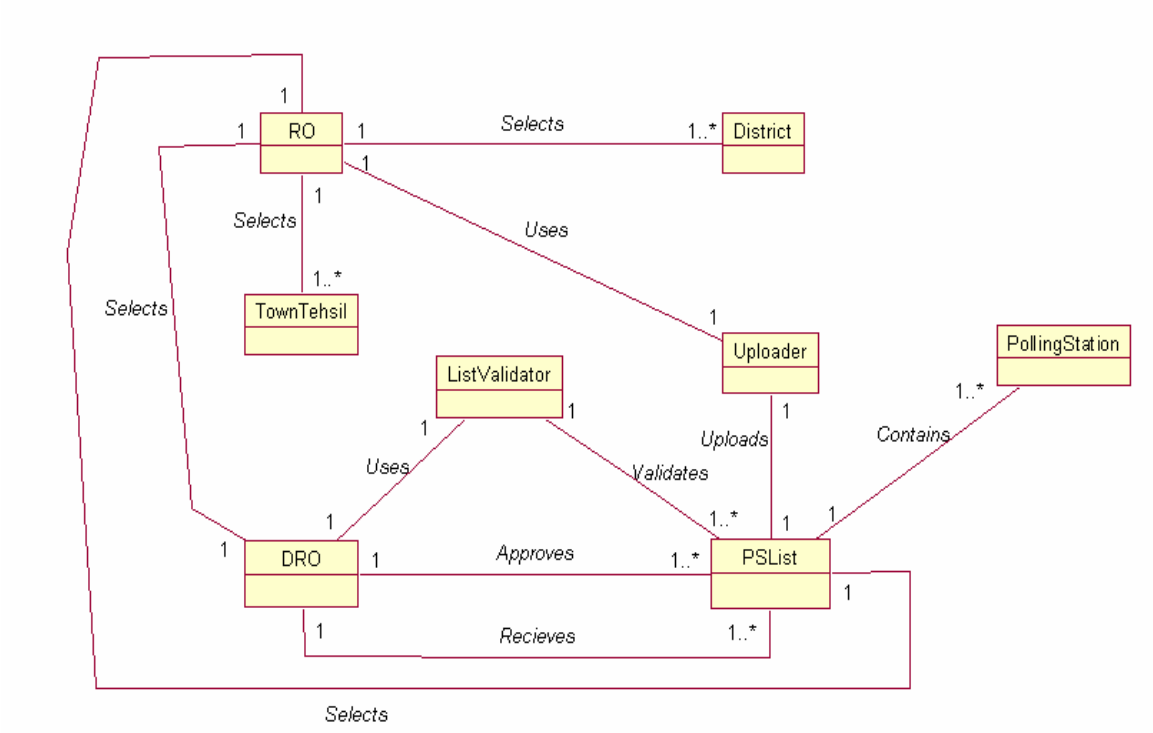


Figure 2.11.20

2.11.21 Domain Model of UC_Prepare_PO_APO_PollingStaffList (UC_21)

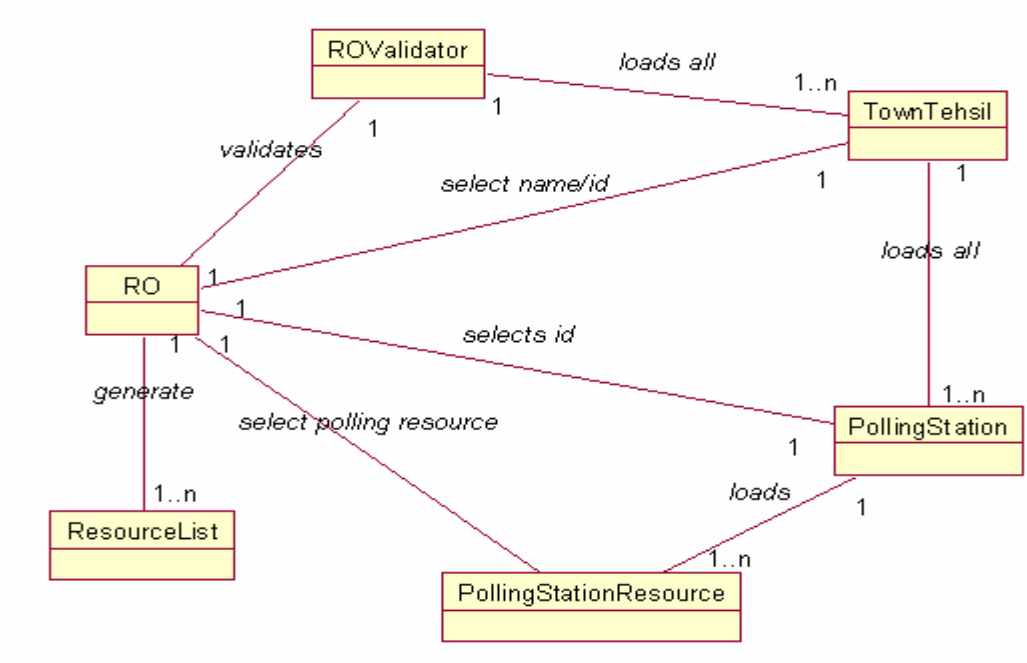


Figure 2.11.21

2.11.22 Domain Model of UC_Approve_PO_APO_PollingStaffList (UC_22)

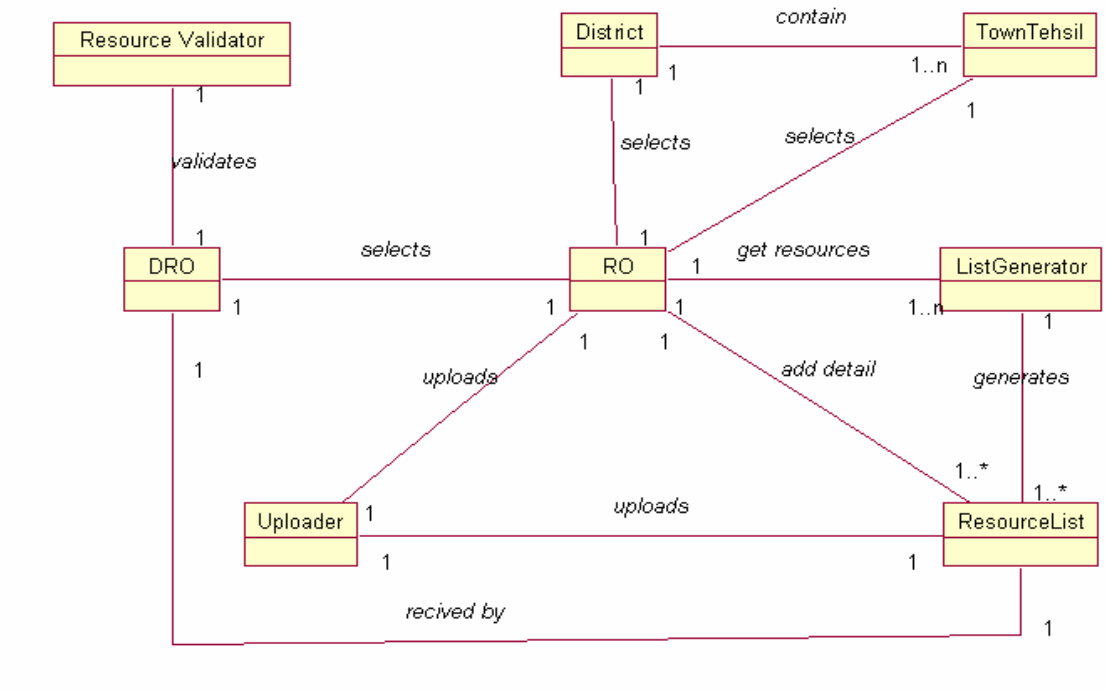


Figure 2.11.22

2.11.23 Domain Model of UC_UpdateResultPOToRO (UC_23)

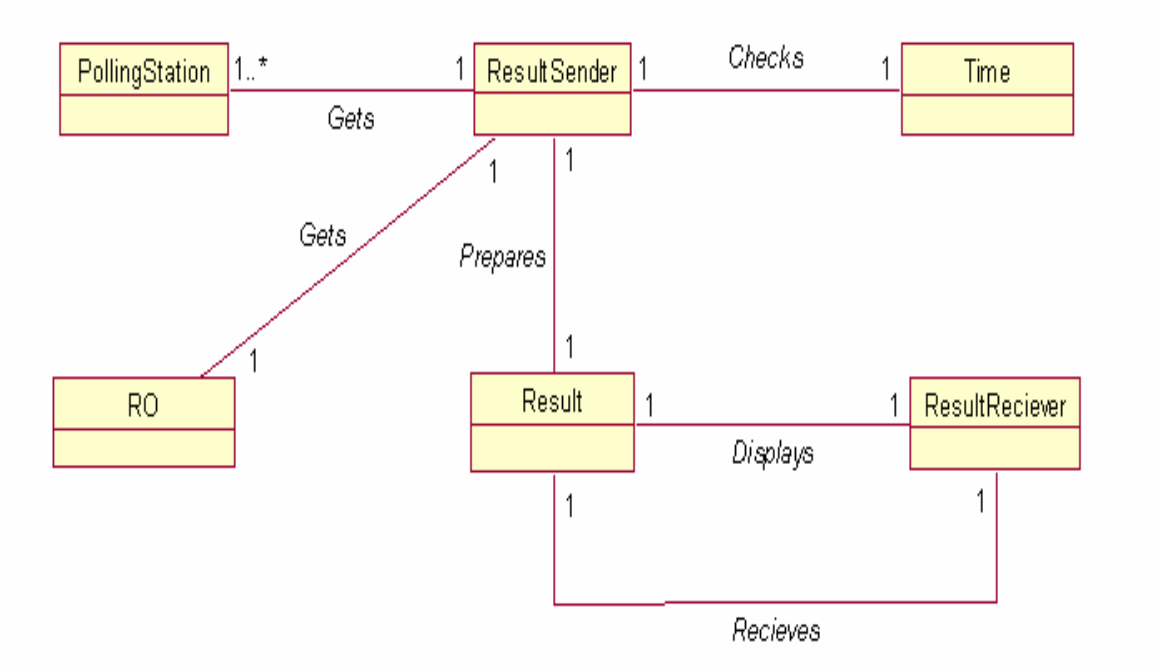


Figure 2.11.23

2.11.24 Domain Model of UC_UpdateResultROToDROCEC (UC_24)

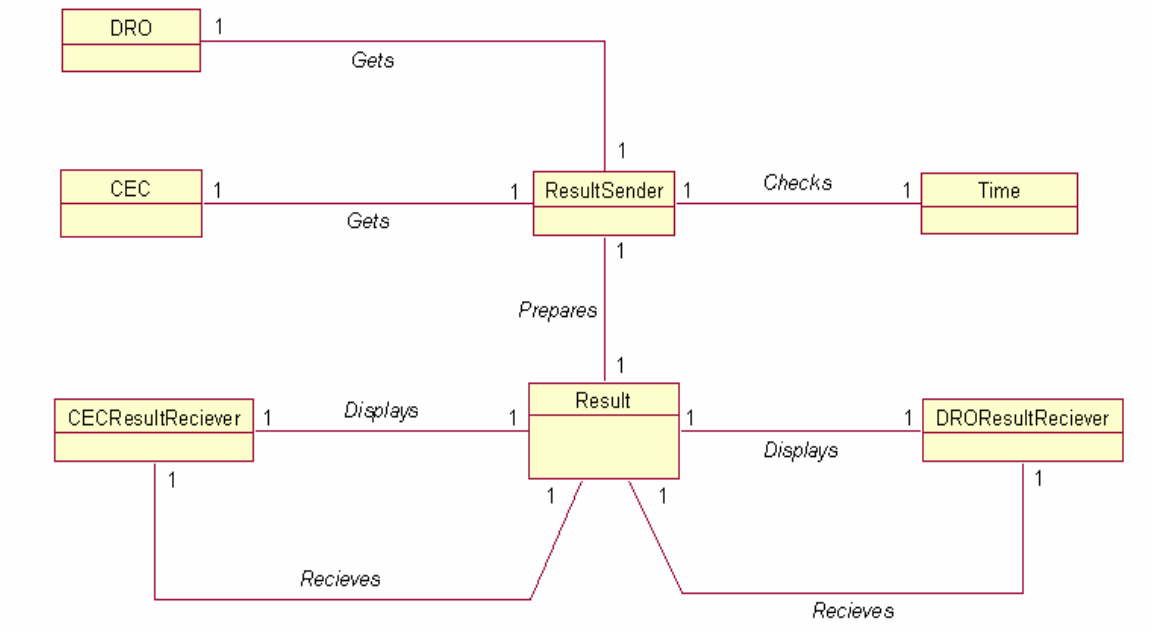


Figure 2.11.24

CHAPTER 3

DESIGNING

3.1 Object-Oriented Design

Design phase is used to managing complexity in the developing system; designing technique helps us to divide large projects into manageable portions that can fit in our brain. Design methods provide a information. Which permits us to store (e.g. on paper or on computer) and be in touch with design decisions.

Object-Oriented design for projects is the method which go ahead to software architectures based on the objects every system or subsystem manipulates (rather than the function it is meant to ensure)

The design of Object-Oriented software requires the definition of multi layered software architecture the specifications and subsystems that perform required functions and provide infrastructure support, a description of objects (classes) that form the building blocks of the system, and a description of the communications mechanisms that allow data to flow between layers, subsystems and objects. Object-oriented designs accomplish all these things.

3.2 Reason for Object-Oriented Design

Object-oriented program has become the leading programming style in the software industry over past few years. The reason for this has to do with the growing size and scale of software project. It becomes extremely to understand a procedural program once it gets above a certain size. Object-oriented program scale up batter, meaning that they are easier to write, understand and maintain than procedural programs of same size. There are basically three reasons for this:

1. Object-oriented programs tent to be written in terms of real-world objects, not internal data structures. This makes them somewhat easier to understand by maintainer & peoples who had to read code – but it makes it harder for the initial designer. Identify objects in a problem is a challenge
2. Object-oriented programs encourage encapsulation – details of objects implementation are hidden from other objects. This keeps a change in one part

of the program from affecting other parts, making the program easier to debug and maintain.

3. Object-oriented programs encourage modularity. This means that pieces of the program do not depend on the other pieces of the program. Those pieces can be reused in future projects, making the new projects easier to build.

The most important purpose of the OOD phase is to convert OOA into something that could actually implement. In this phase, we decide what information each class knows, and what other classes it needs to know about. In particular, the OOD is concerned with how information flows through the system.

3.3 Software Design Process

In the Software design process it develops the architecture details required to build system or product. The software design process take in the following activities:

- Partition & Analysis model into subsystems
- Identify the concurrency that is dictated by the problem
- Allocate the subsystems to processors and tasks
- Develop a design for user interface
- Chose a basic strategy for implementing data management
- Identify global recourses and the control mechanism required to access them
- Design appropriate control mechanisms for the system, including task management.
- Consider how boundary condition should be handled
- Review and consider tradeoffs

3.4 Characteristics for the evaluation of good Design

Following characteristics serves as a guideline for the evaluation of a good design.

- The design must implement all the clear requirements included in the analysis model as well must contain the contained requirements required by the consumer.

- The design must me readable & understandable for those who generate code and for those who test and subsequently support the software.
- The design should provide complete picture of software, addressing the date, functional behavioral domains from an implementation perspective.

3.5 Partitioning the Analysis Model

The analysis model is partitioned into cohesive collection of classes, relationships and behavior. The design elements are packed as a subsystem. In subsystem all the elements which constitute the subsystem are characterized by their responsibilities i.e. a subsystem can be identified by the services that it provides. A service is collection of operations that perform a specific function.

When a system is partitioned into subsystems, another design activity called "Layering" occurs. Each layer of system contains one or more subsystems and represents a different level of abstraction of the functionality required to accomplish system functions.

Class Diagrams

Class Diagrams explain the types of objects in the system and the various kind of relationship that exist among them. There are two principals' kinds of static relationships:

- **Associations**

  Represent relationship between instances of classes

- **Subtypes**

  Like (a doctor is a kind of person)

3.6 Architecture Diagram

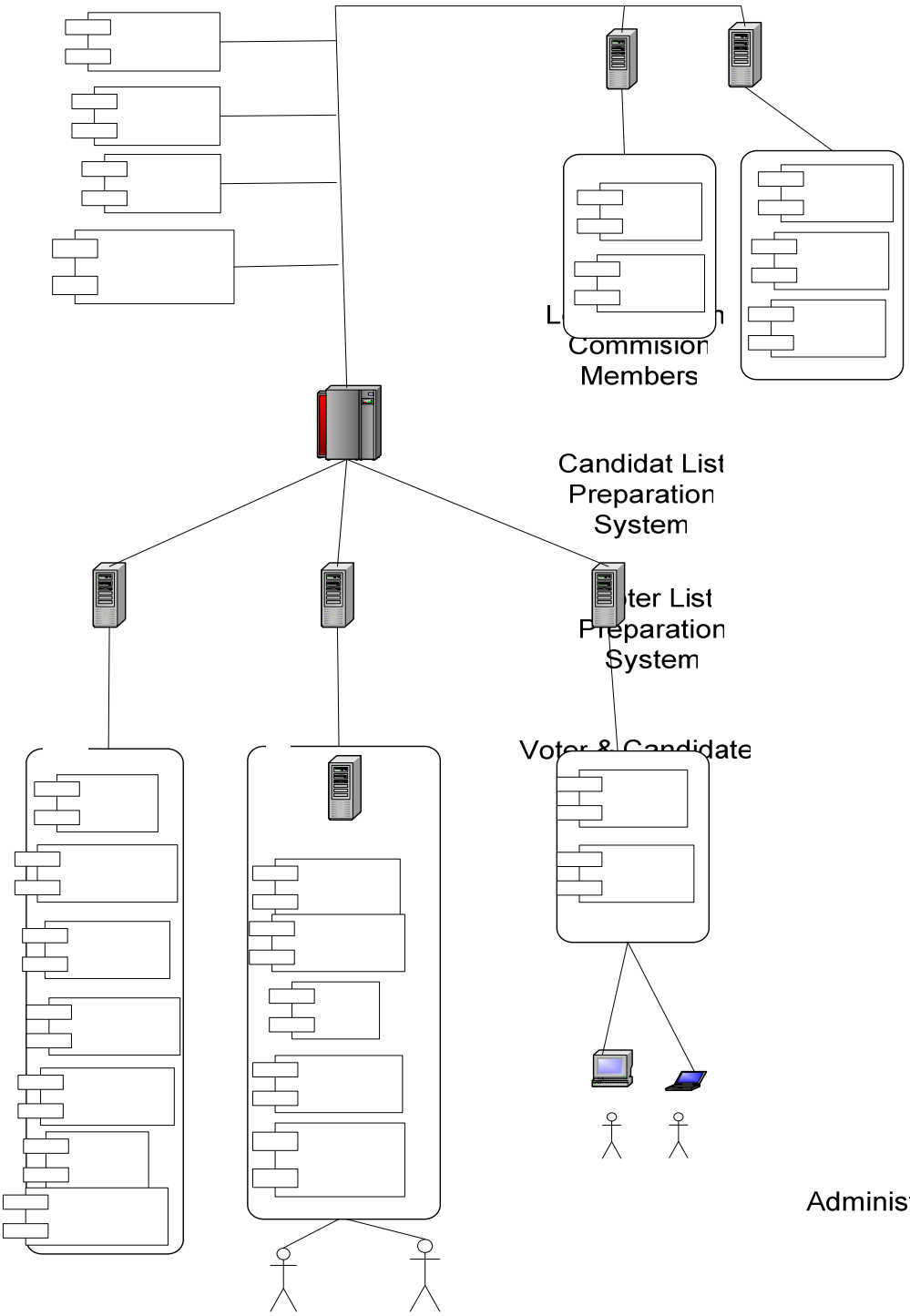


Figure 3.6

3.7 System Sequence Diagram

3.7.1 System Sequence Diagram of UC_Prepare_ElectorlRoll (UC_1)

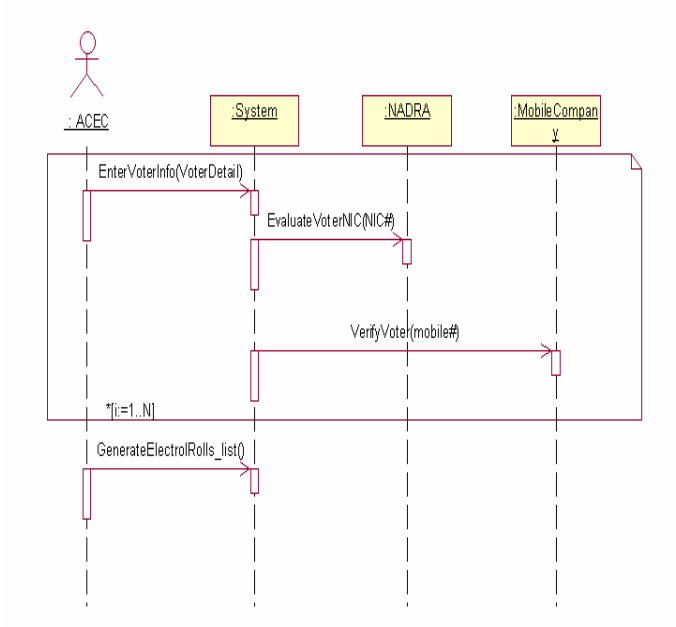


Figure 3.7.1

3.7.2 System Sequence Diagram of UC_Candidate_Nomination (UC_2)

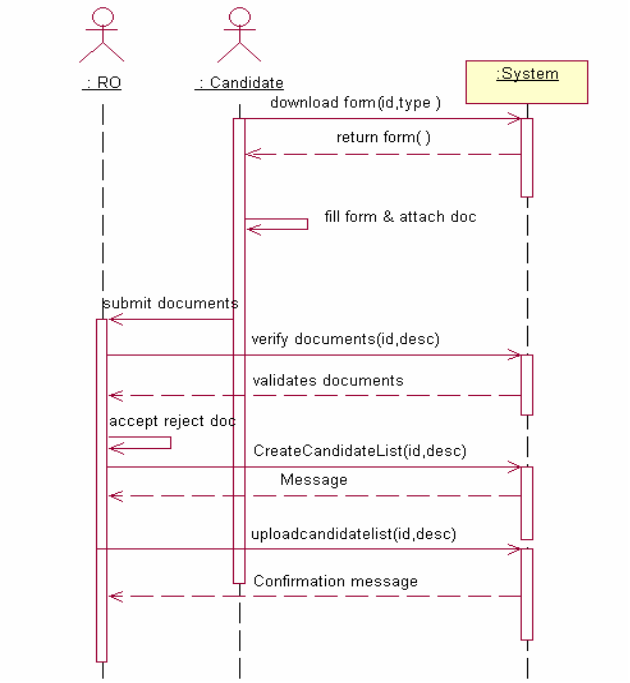


Figure 3.7.2

3.7.3 System Sequence Diagram of UC_Prepare_Symbol _CandidateList (UC_3)

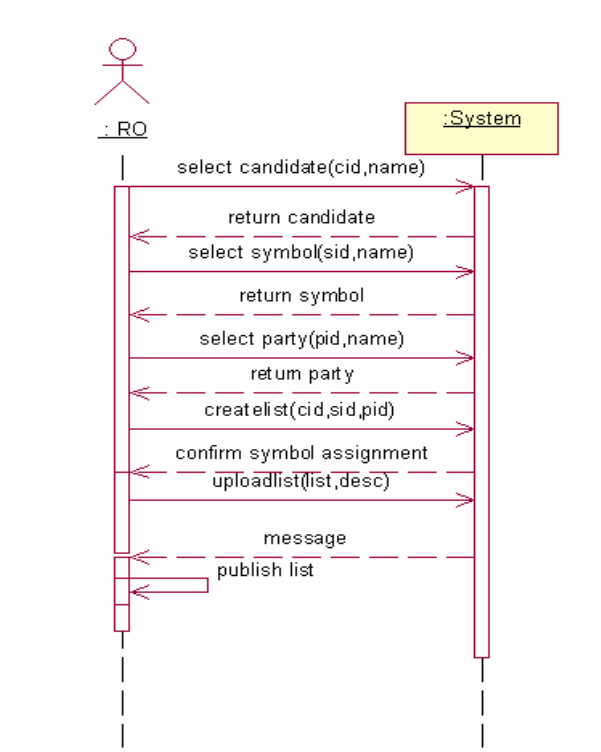


Figure 3.7.3

3.7.4 System Sequence Diagram of UC_VoterList_DRODistribution (UC_4)

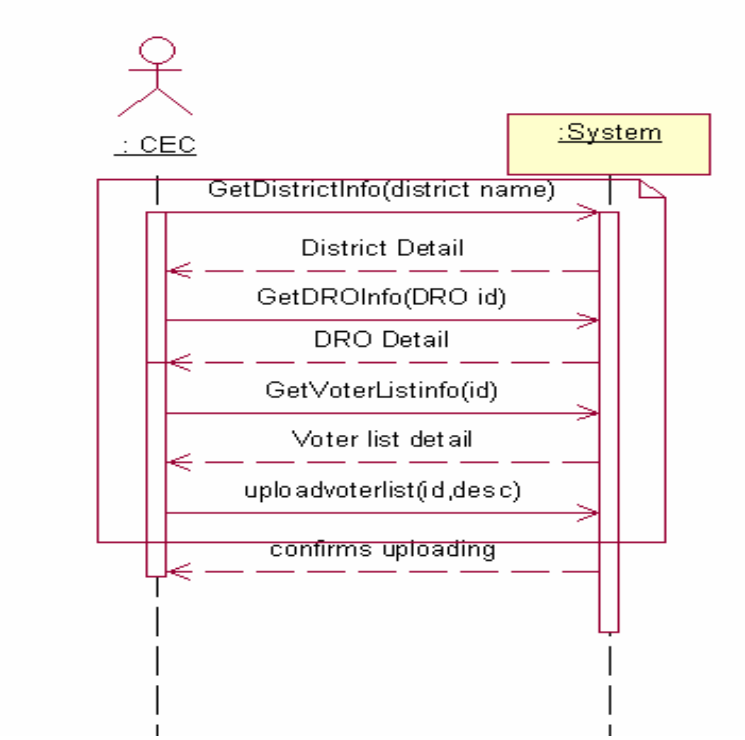


Figure 3.7.4

3.7.5 System Sequence Diagram of UC_VoterList_RODistribution (UC_5)

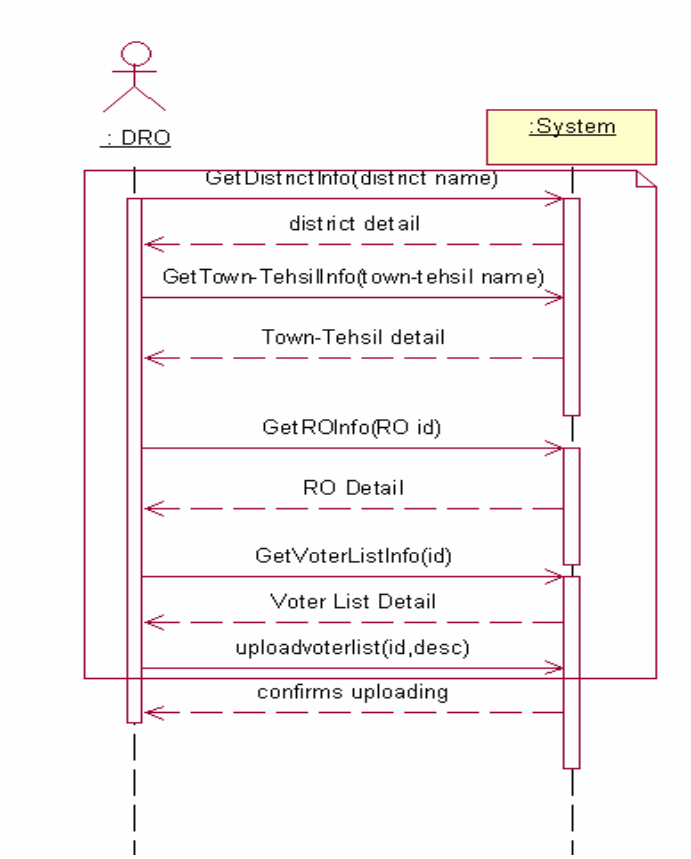


Figure 3.7.5

3.7.6 System Sequence Diagram of UC_VoterList_PODistribution (UC_6)

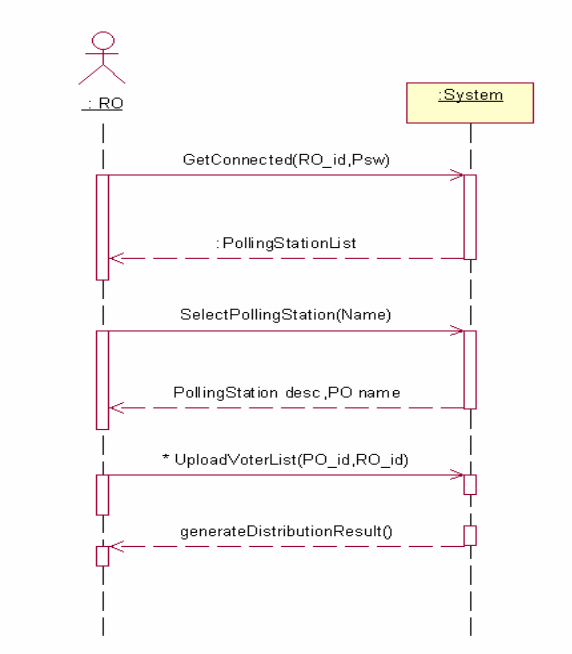


Figure 3.7.6

3.7.7 System Sequence Diagram of UC_Validate_Voter (UC_7)

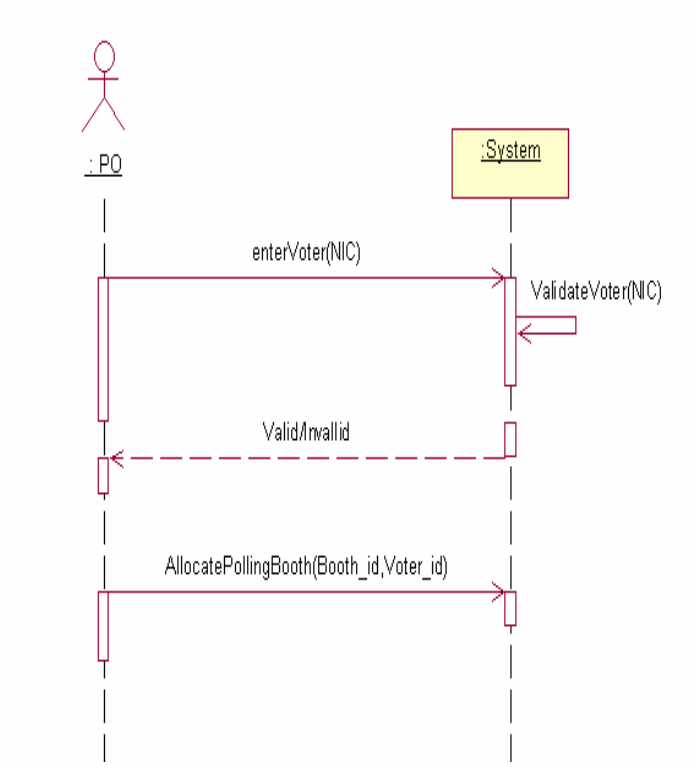


Figure 3.7.7

3.7.8 System Sequence Diagram of UC_Cast_Vote (UC_8)

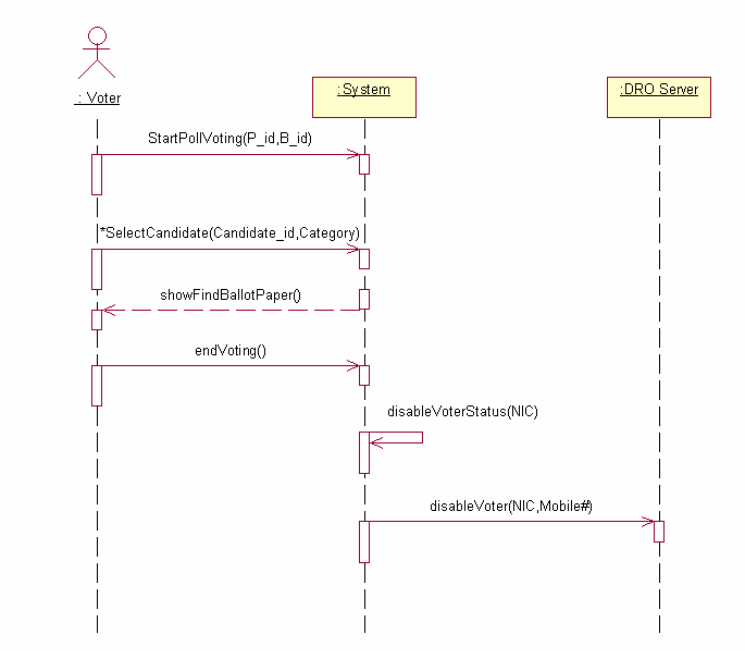


Figure 3.7.8

3.7.9 System Sequence Diagram of UC_Compile_Submit_POResult (UC_9)

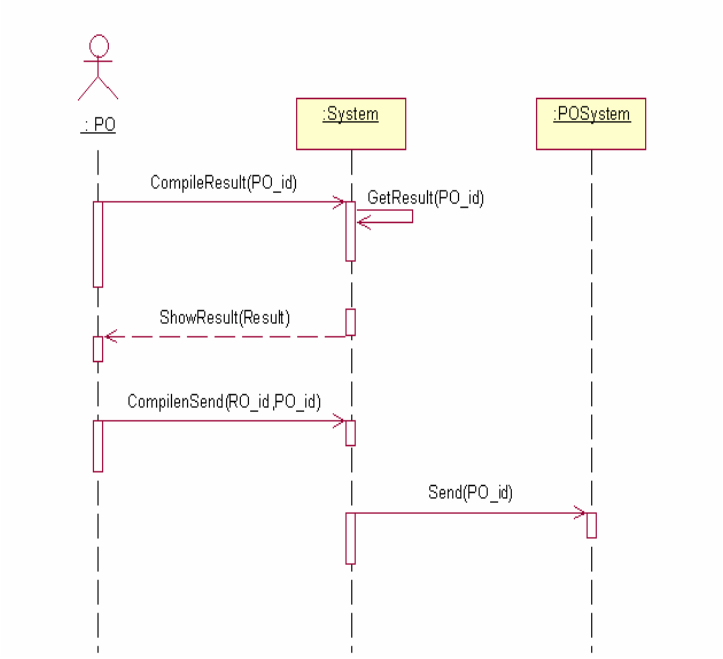


Figure 3.7.9

3.7.10 System Sequence Diagram of UC_Compile_ROResult (UC_10)

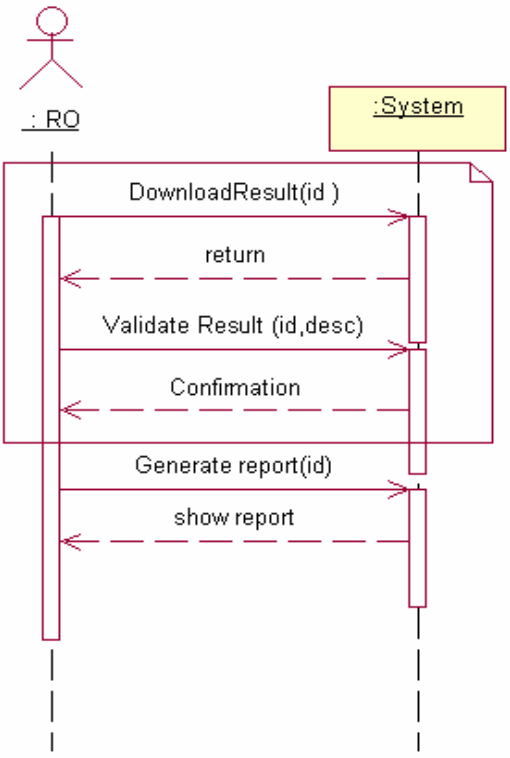


Figure 3.7.10

3.7.11 System Sequence Diagram of UC_Submit_ROResult (UC_11)

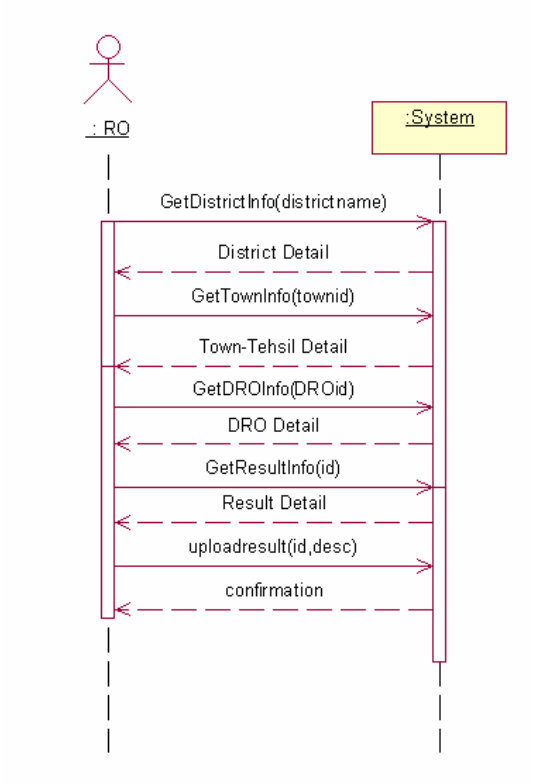


Figure 3.7.11

3.7.12 System Sequence Diagram of UC_Compile_DROResult (UC_12)

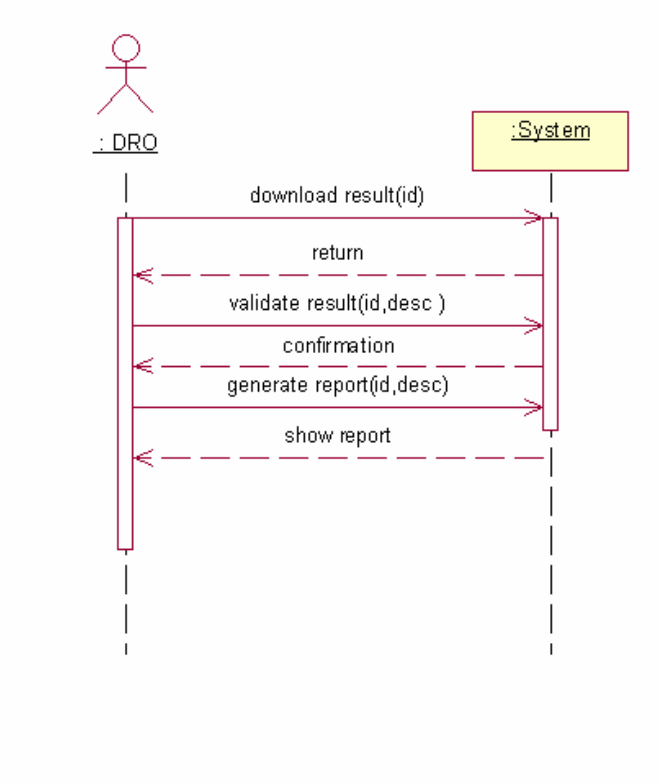


Figure 3.7.12

3.7.13 System Sequence Diagram of UC_Submit_DROResult (UC_13)

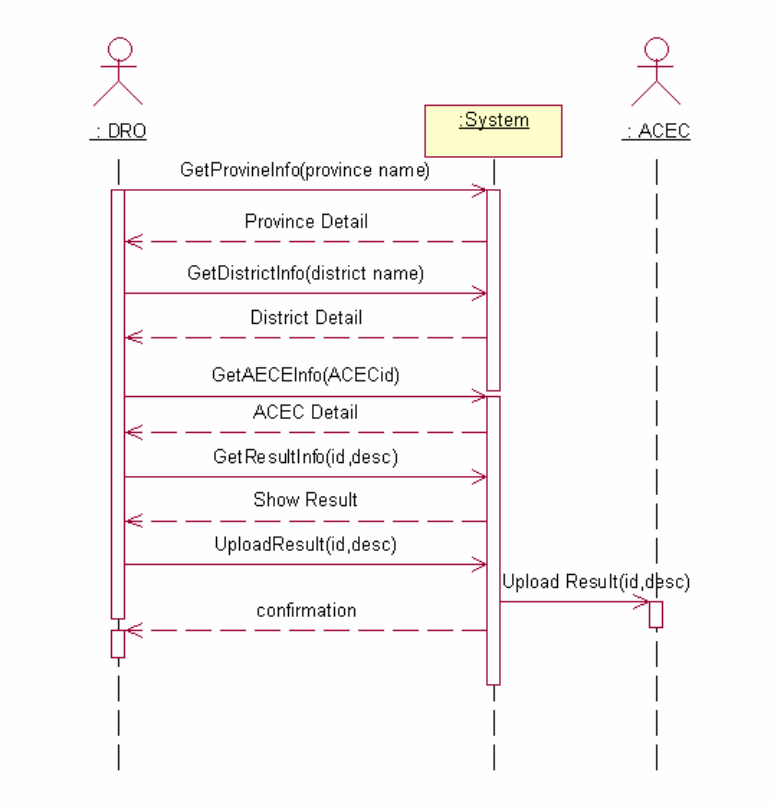


Figure 3.7.13

3.7.14 System Sequence Diagram of UC_Compile_FinalResult (UC_14)

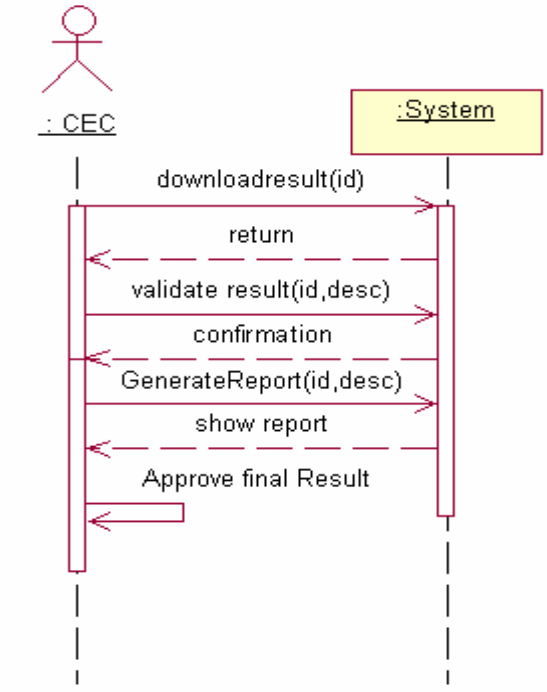


Figure 3.7.14

3.7.15 System Sequence Diagram of UC_Prepare_DROList (UC_15)

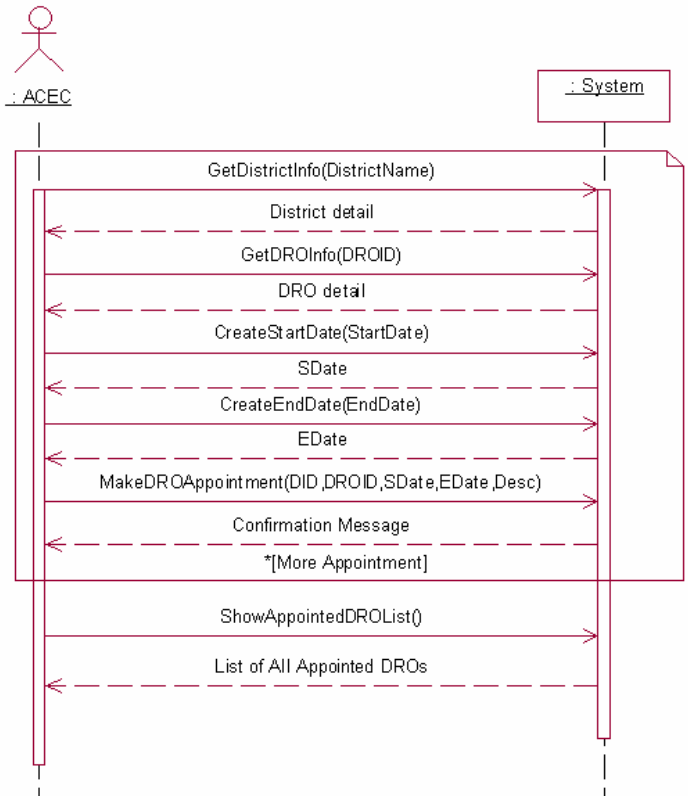


Figure 3.7.15

3.7.16 System Sequence Diagram of UC_Approve_DROList (UC_16)

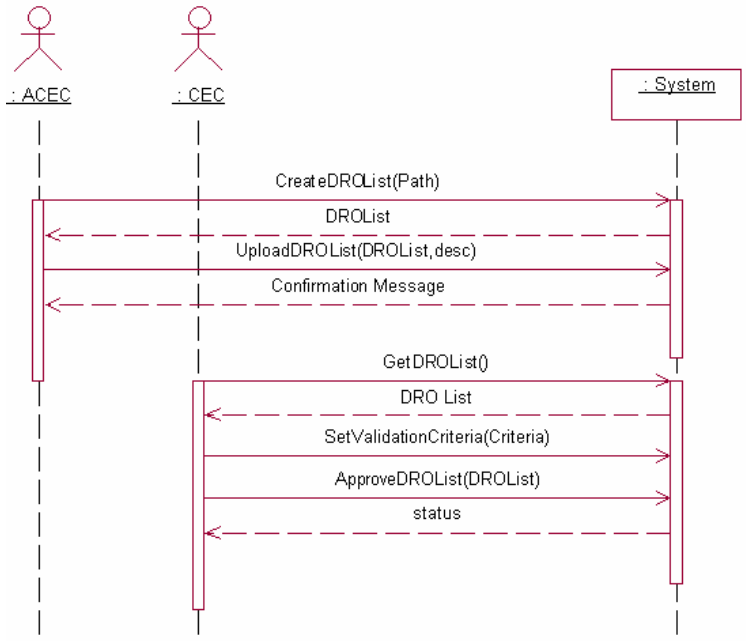


Figure 3.7.16

3.7.17 System Sequence Diagram of UC_Prepare_ROList (UC_17)

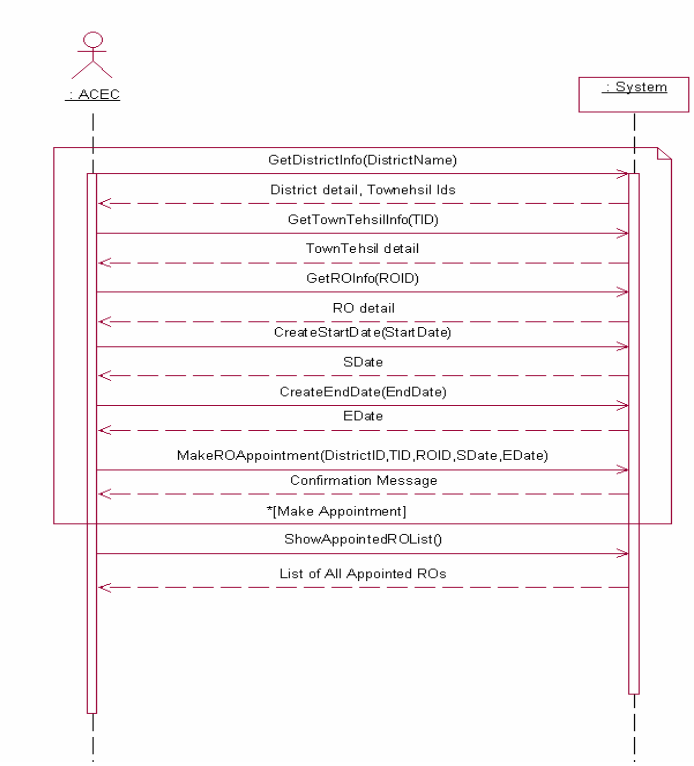


Figure 3.7.17

3.7.18 System Sequence Diagram of UC_Approve_ROList (UC_18)

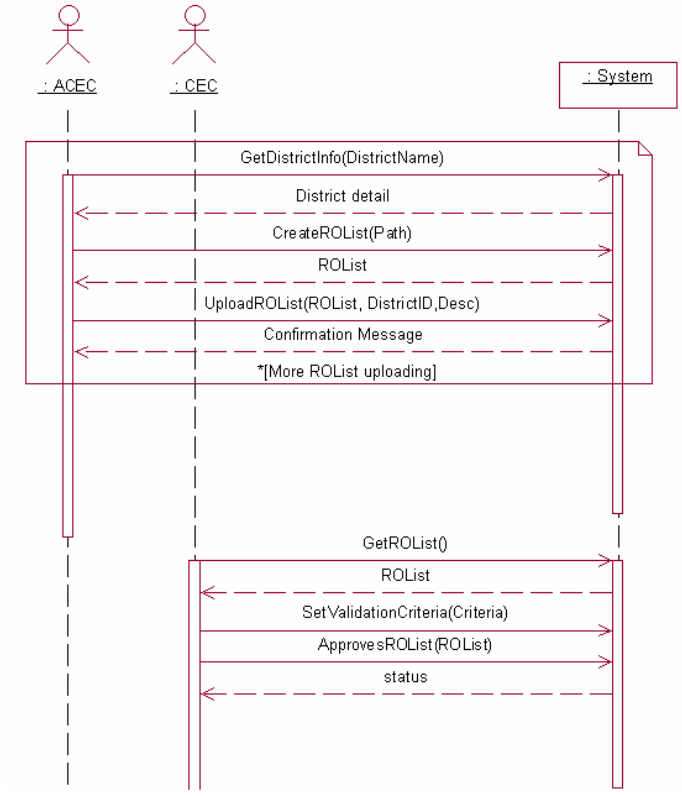


Figure 3.7.18

3.7.19 System Sequence Diagram of UC_Prepare_PollingStationList (UC_19)

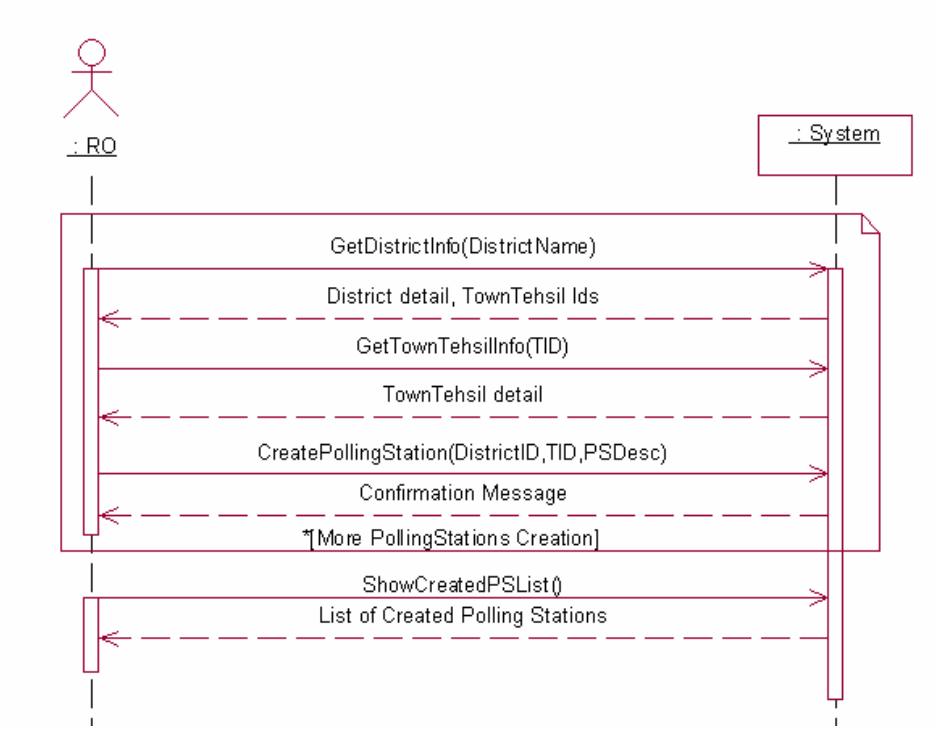


Figure 3.7.19

3.7.20 System Sequence Diagram of UC_Approve_PollingStationList (UC_20)

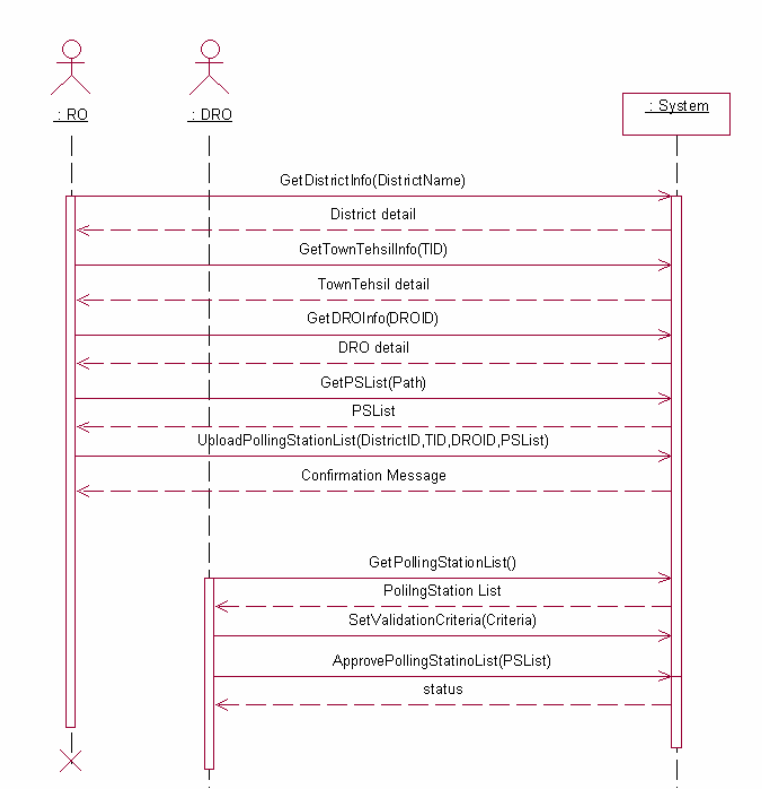


Figure 3.7.20

3.7.21 System Sequence Diagram of UC_Prepare_PO_APO_PollingStaffList (UC_21)

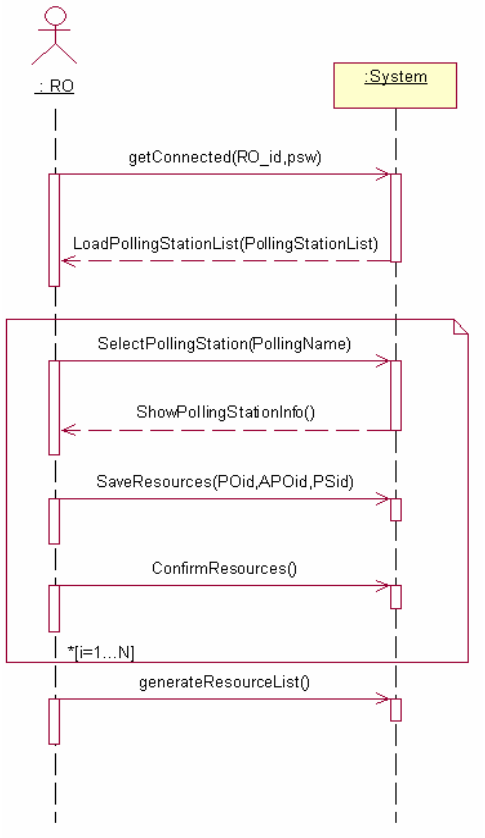


Figure 3.7.21

3.7.22 System Sequence Diagram of UC_Approve_PO_APO_PollingStaffList (UC_22)

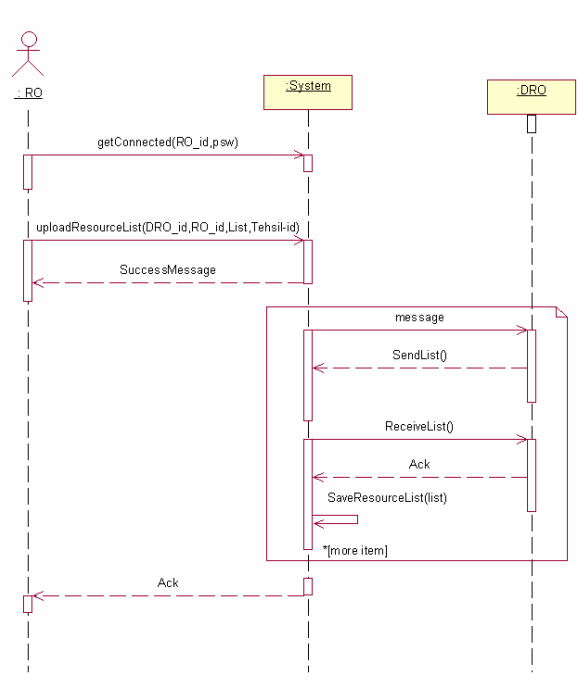


Figure 3.7.22

3.7.23 System Sequence Diagram of UC_UpdateResultPOToRO (UC_23)

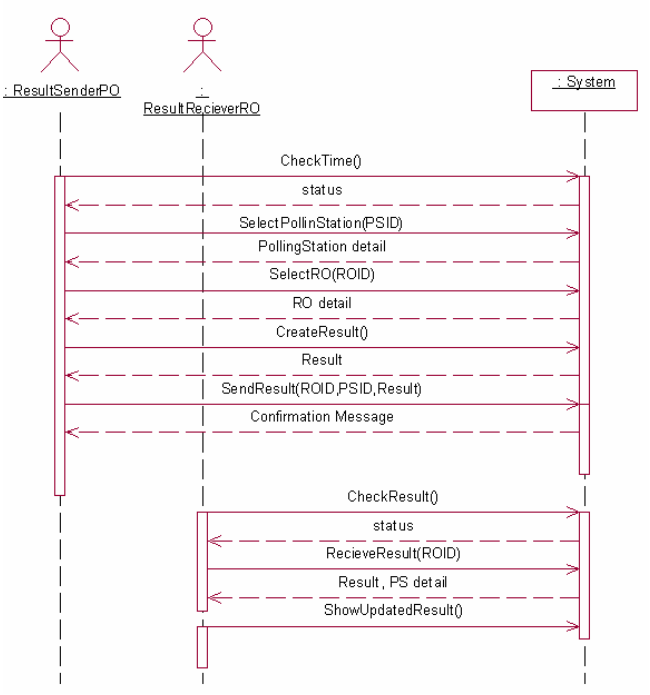


Figure 3.7.23

3.7.24 System Sequence Diagram of UC_UpdateResultROToDROCEC (UC_24)

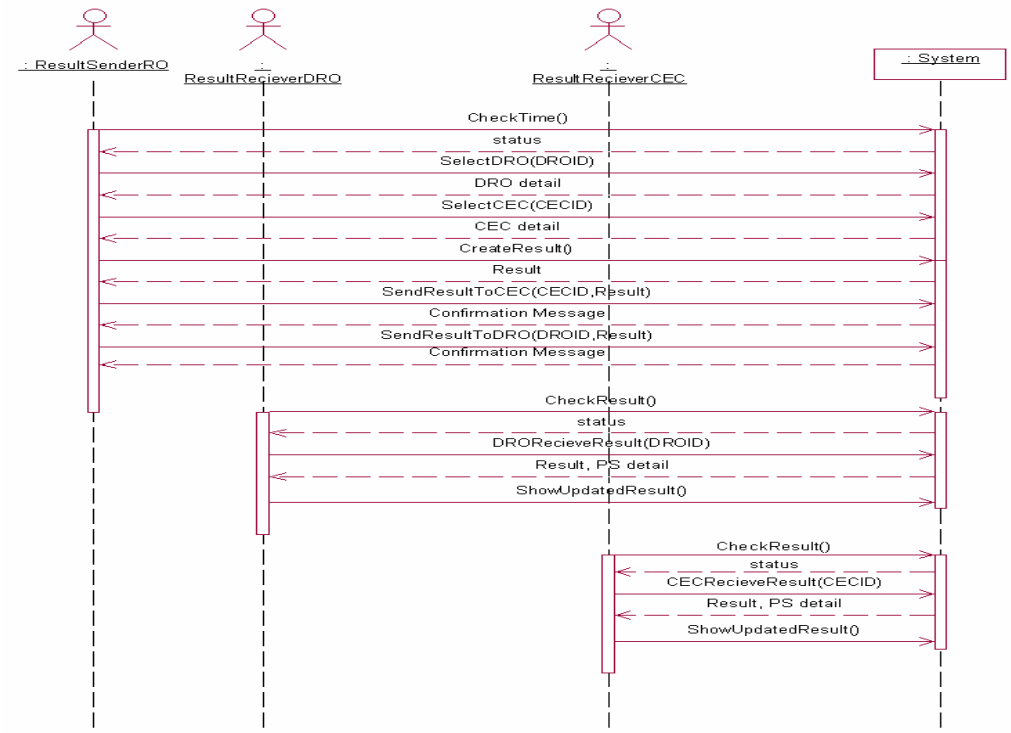


Figure 3.7.24

3.8 Sequence Diagrams

3.8.1 Sequence Diagram of UC_Prepare_ElectorlRoll (UC_1)

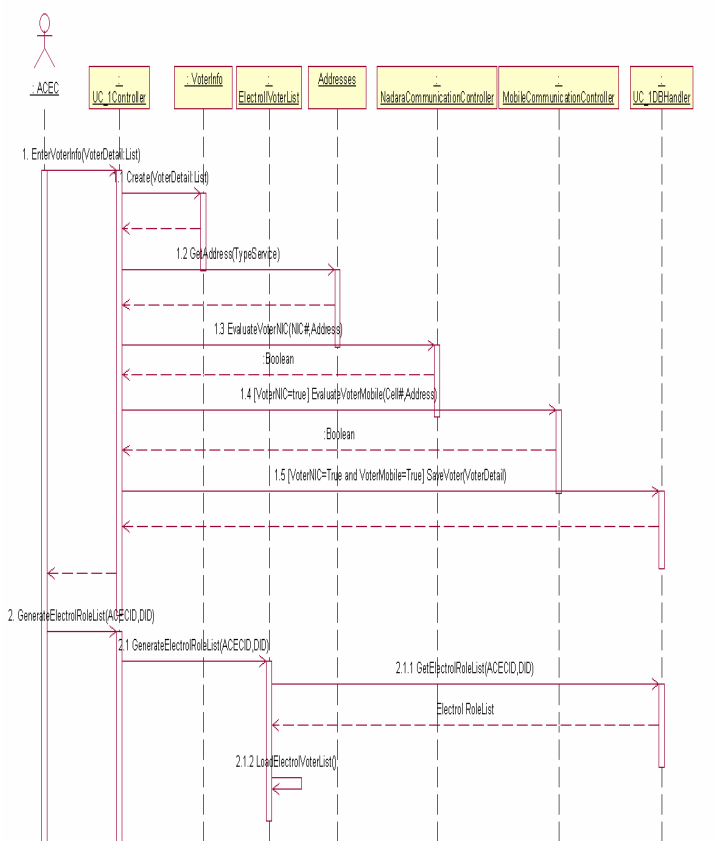


Figure 3.8.1

3.8.2 Sequence Diagram of UC_Candidate_Nomination (UC_2)

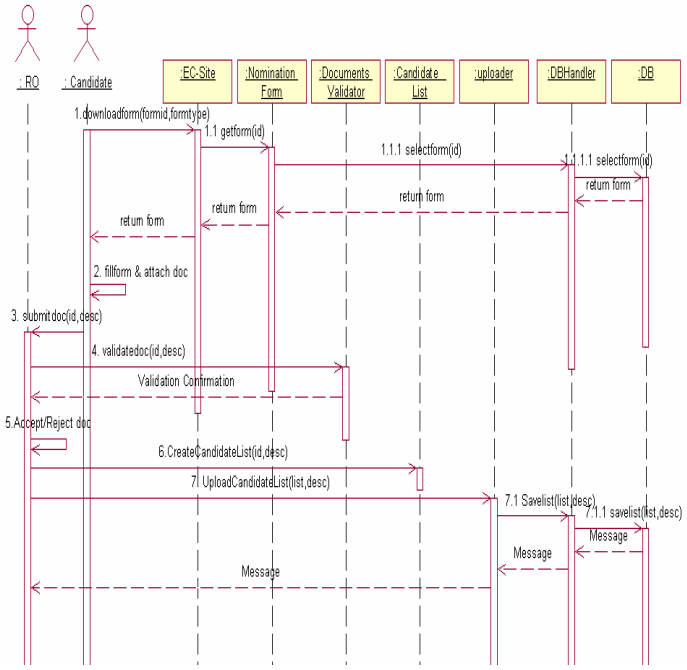


Figure 3.8.2

3.8.3 Sequence Diagram of UC_Prepare_Symbol _CandidateList (UC_3)

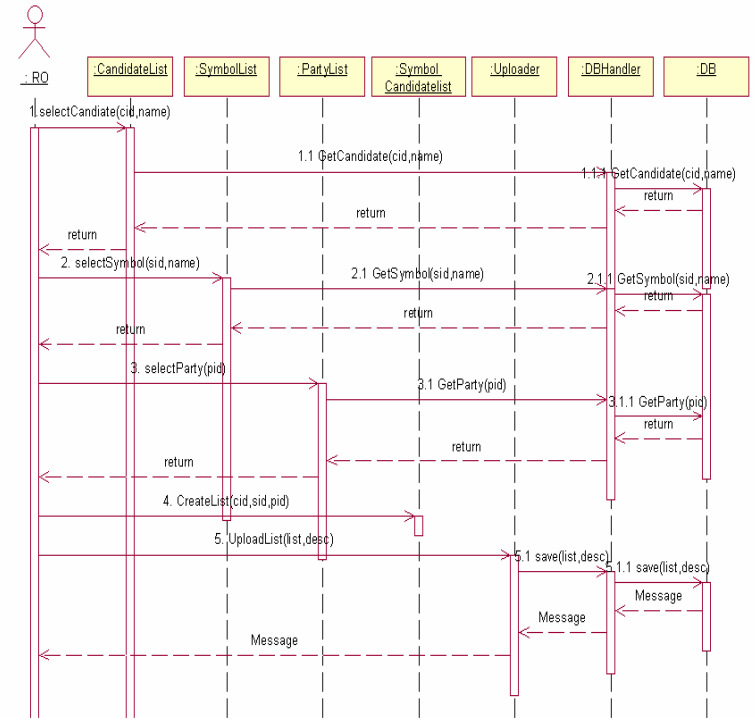


Figure 3.8.3

3.8.4 Sequence Diagram of UC_VoterList_DRODistribution (UC_4)

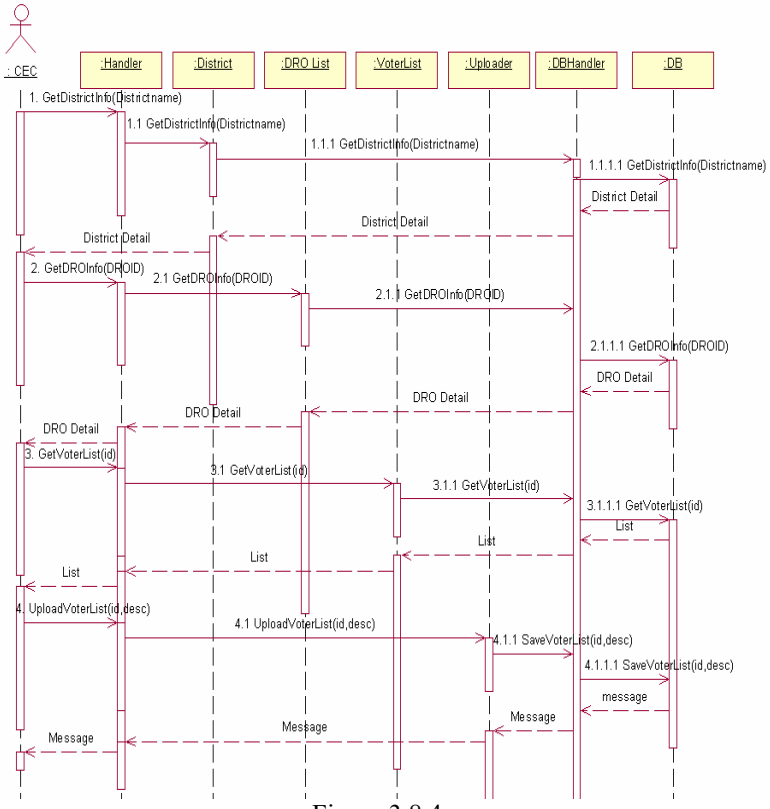


Figure 3.8.4

3.8.5 Sequence Diagram of UC_VoterList_RODistribution (UC_5)

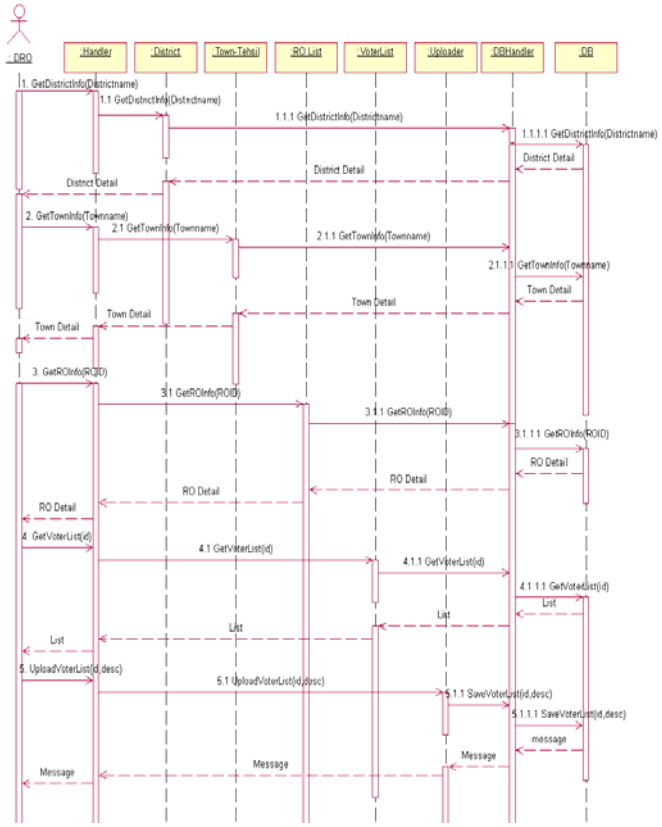


Figure 3.8.5

3.8.6 Sequence Diagram of UC_VoterList_PODistribution (UC_6)

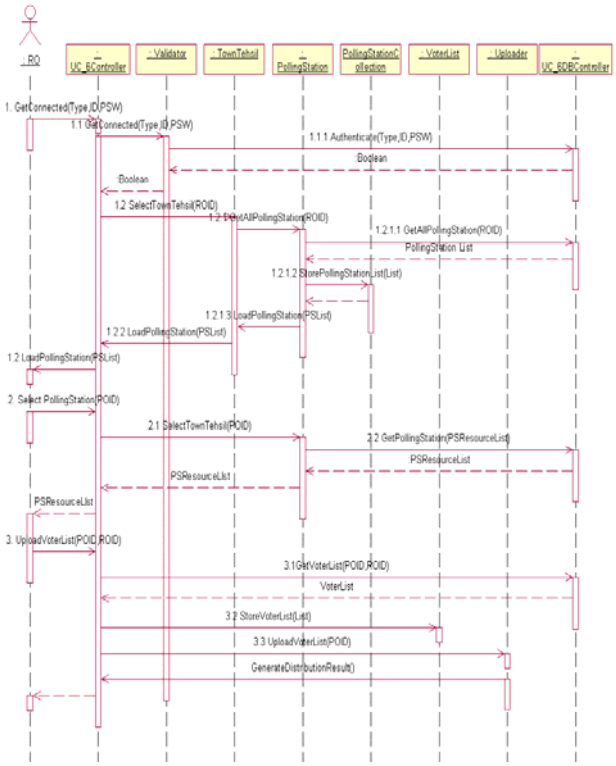


Figure 3.8.6

3.8.7 Sequence Diagram of UC_Validate_Voter (UC_7)

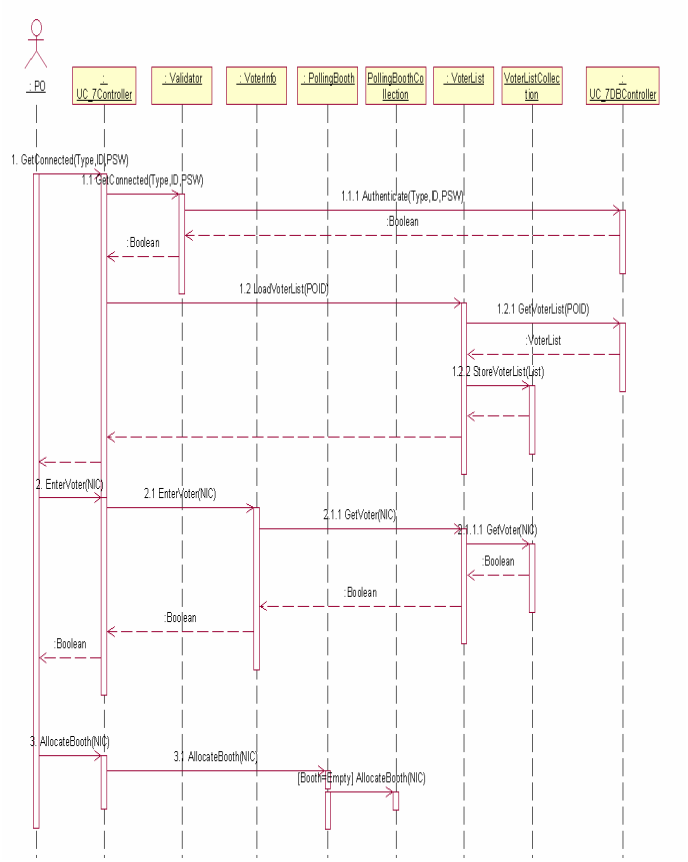


Figure 3.8.7

3.8.8 Sequence Diagram of UC_Cast_Vote (UC_8)

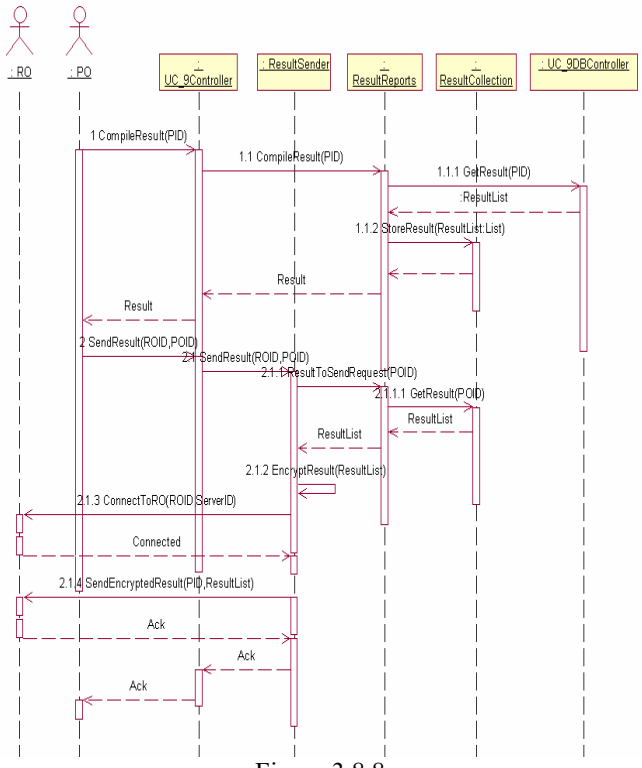


Figure 3.8.8

3.8.9 Sequence Diagram of UC_Compile_Submit_POResult (UC_9)

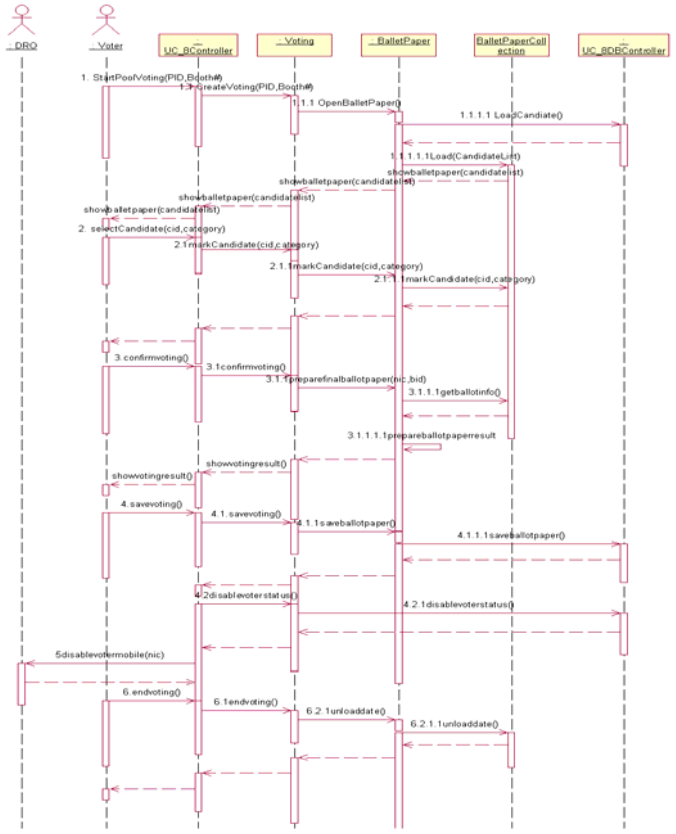


Figure 3.8.9

3.8.10 Sequence Diagram of UC_Compile_ROResult (UC_10)

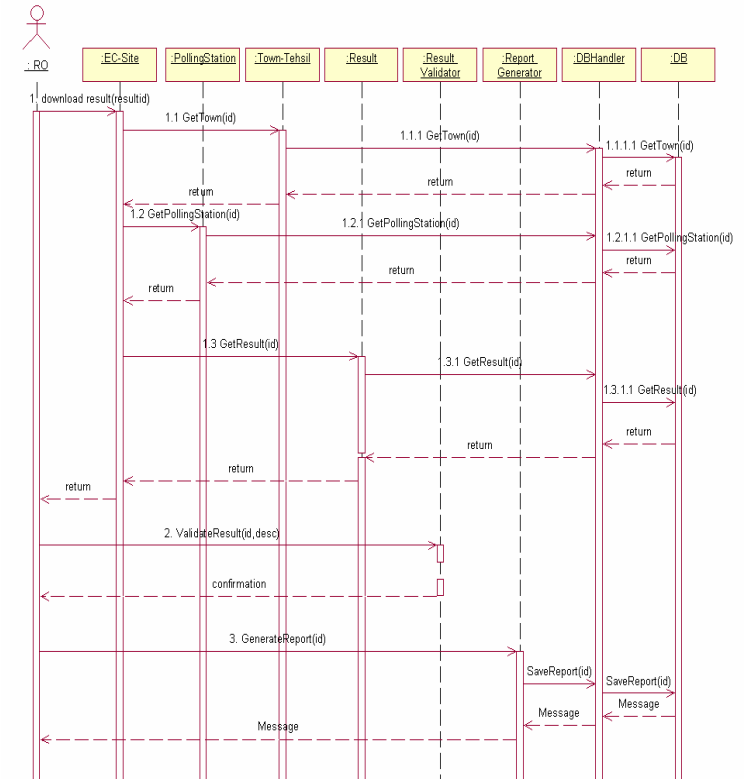


Figure 3.8.10

3.8.11 Sequence Diagram of UC_Submit_ROResult (UC_11)

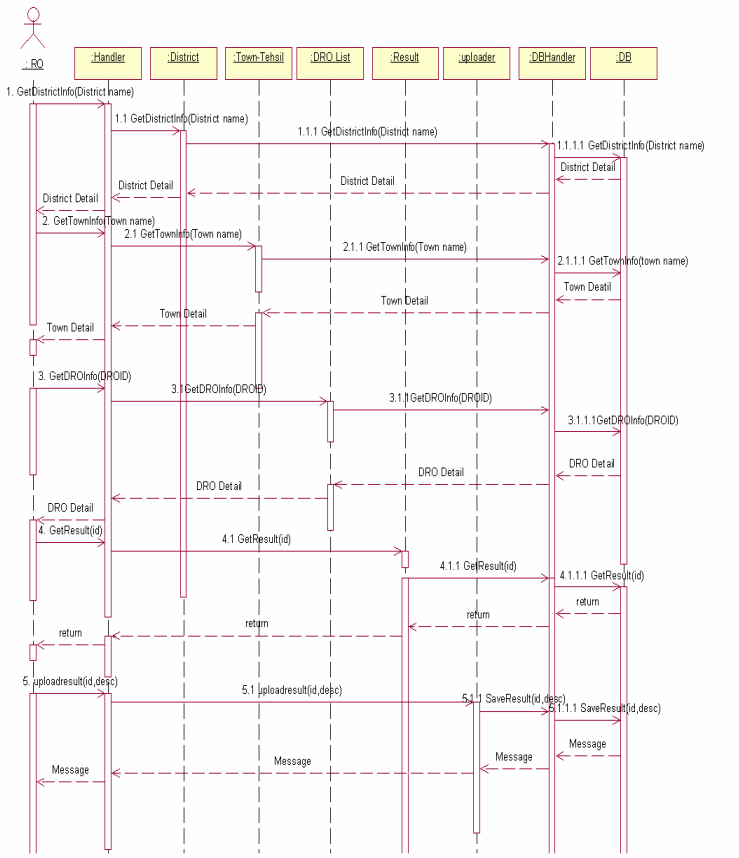


Figure 3.8.11

3.8.12 Sequence Diagram of UC_Compile_DROResult (UC_12)

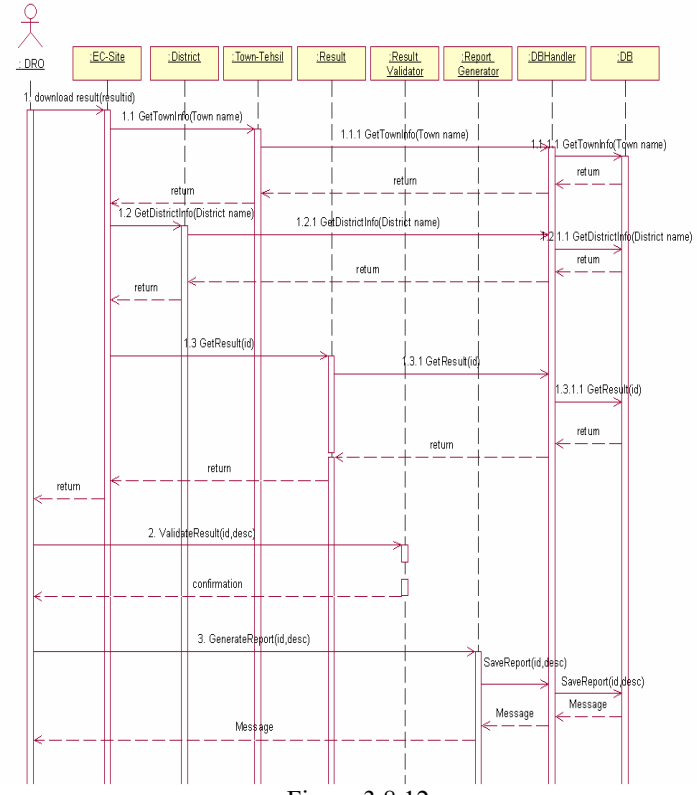


Figure 3.8.12

3.8.13 Sequence Diagram of UC_Submit_DROResult (UC_13)

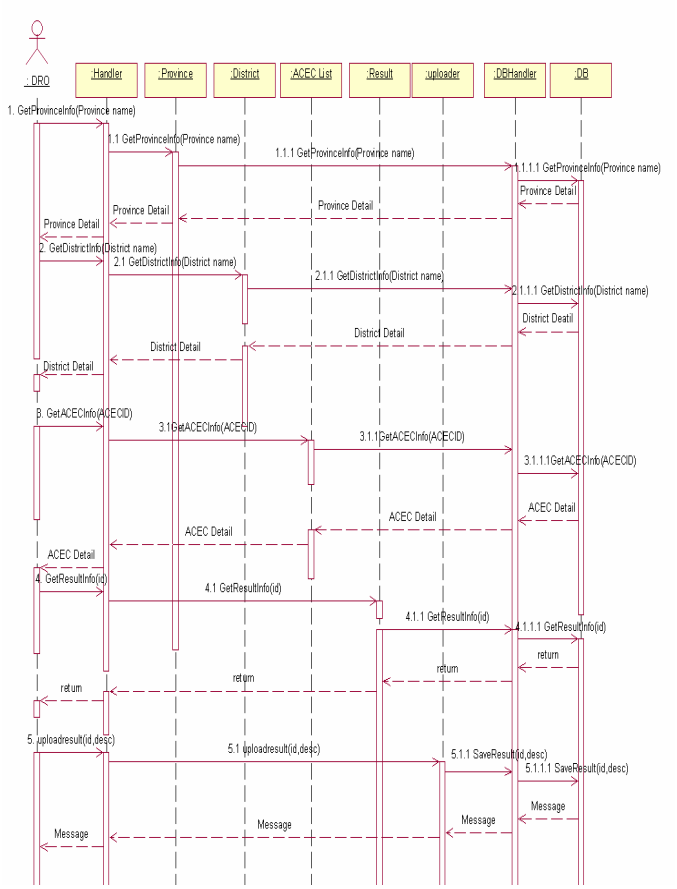


Figure 3.8.13

3.8.14 Sequence Diagram of UC_Compile_FinalResult (UC_14)

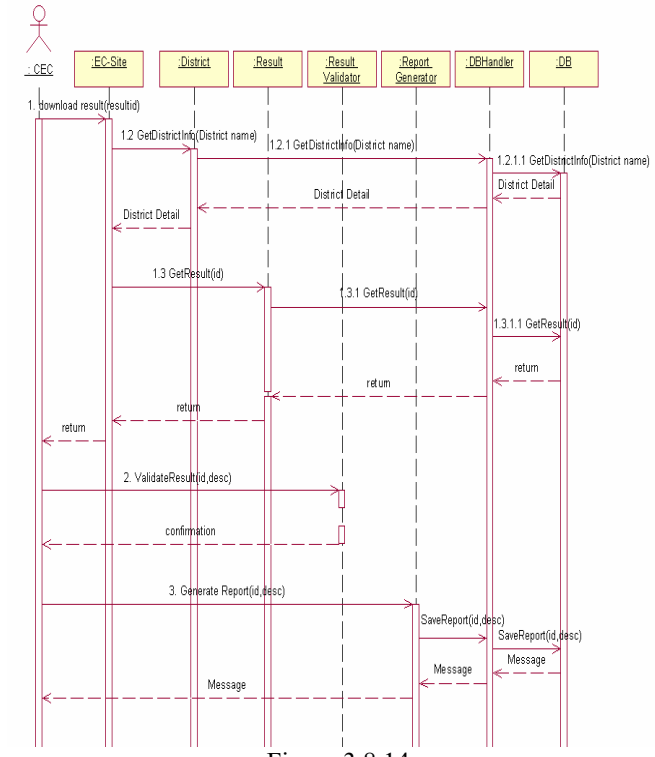


Figure 3.8.14

3.8.15 Sequence Diagram of UC_Prepare_DROList (UC_15)

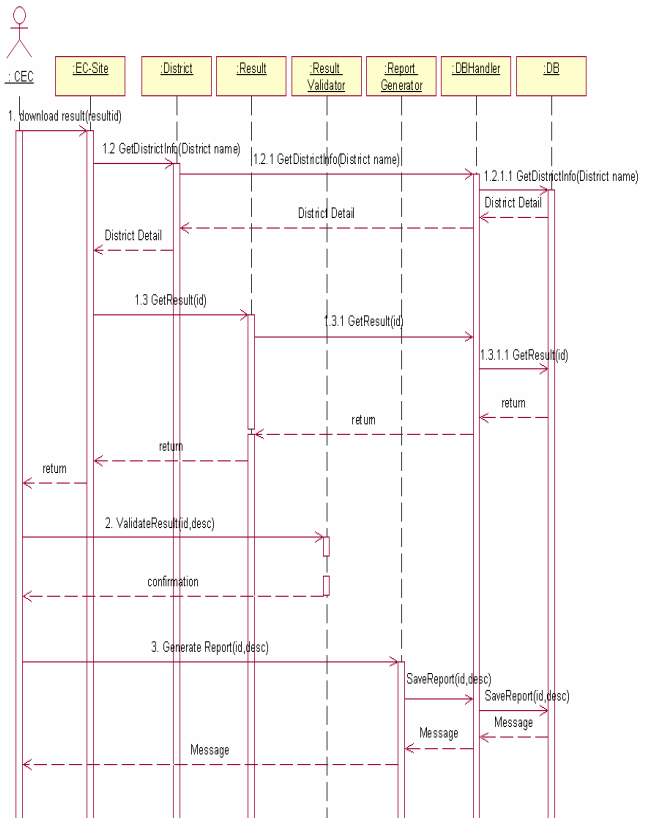


Figure 3.8.15

3.8.16 Sequence Diagram of UC_Approve_DROList (UC_16)

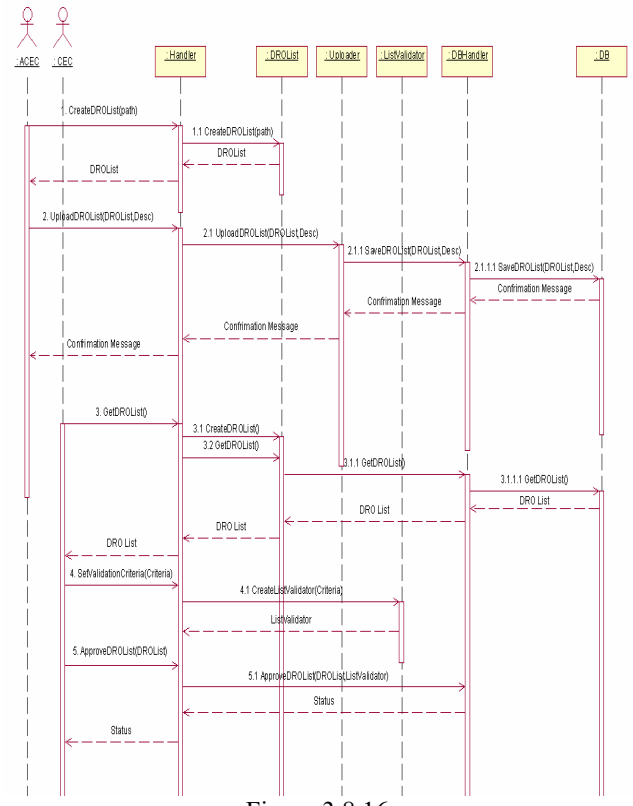


Figure 3.8.16

3.8.17 Sequence Diagram of UC_Prepare_ROList (UC_17)

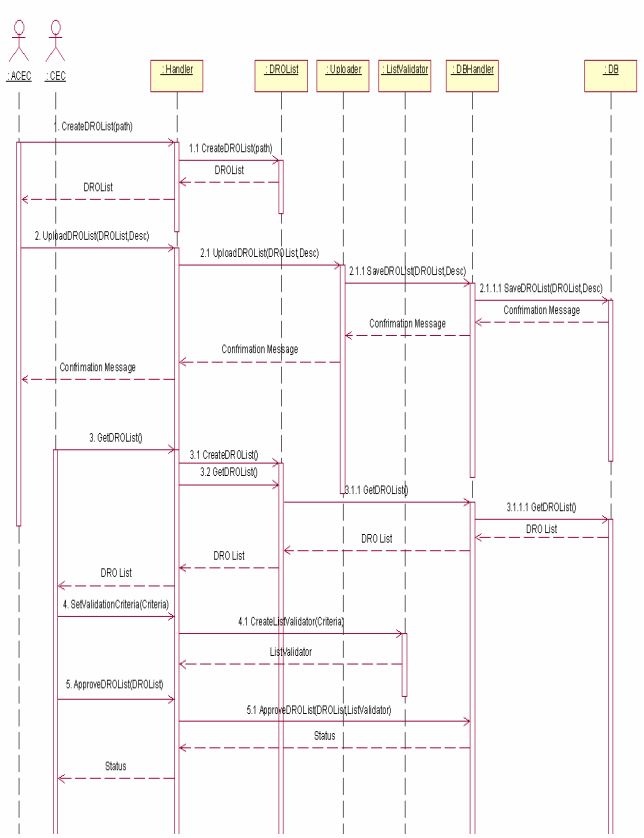


Figure 3.8.17

3.8.18 Sequence Diagram of UC_Approve_ROList (UC_18)

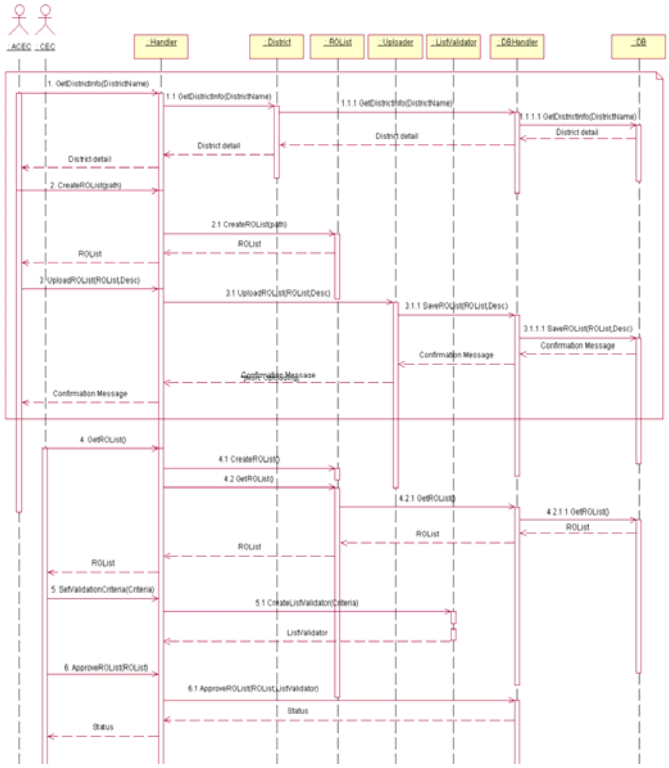


Figure 3.8.18

3.8.19 Sequence Diagram of UC_Prepare_PollingStationList (UC_19)

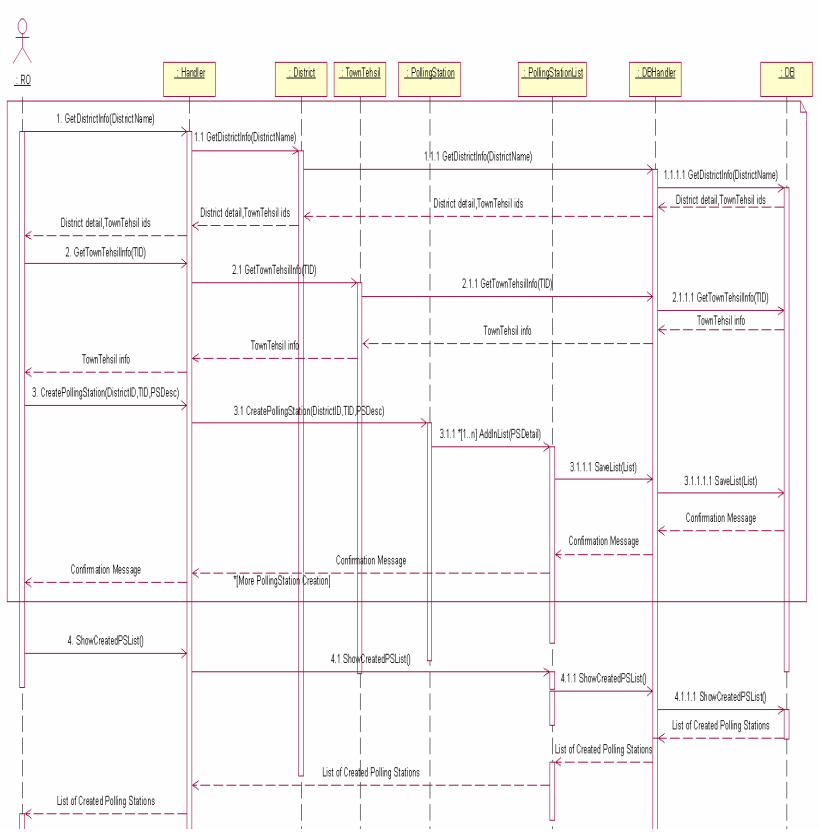


Figure 3.8.19

3.8.20 Sequence Diagram of UC_Approve_PollingStationList (UC_20)

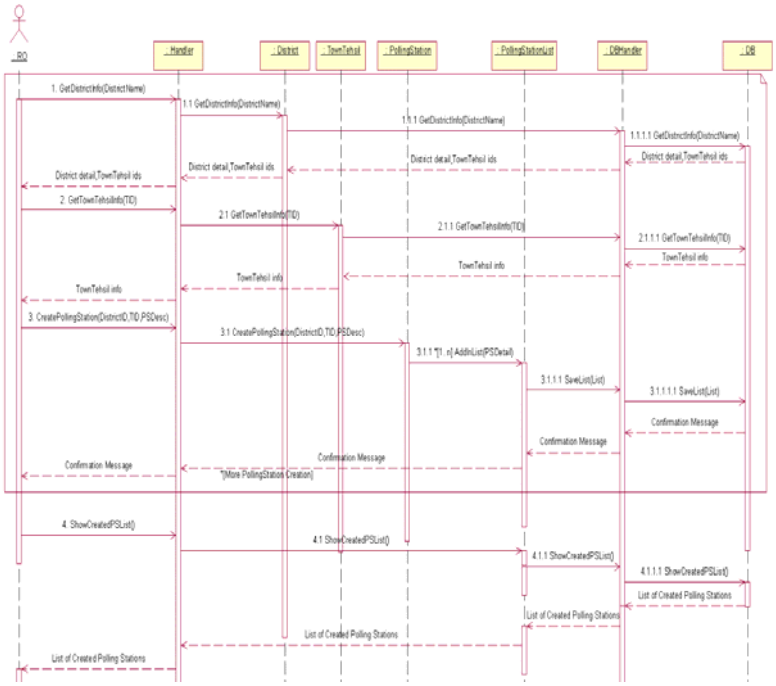


Figure 3.8.20

3.8.21 Sequence Diagram of UC_Prepare_PO_APO_PollingStaffList (UC_21)

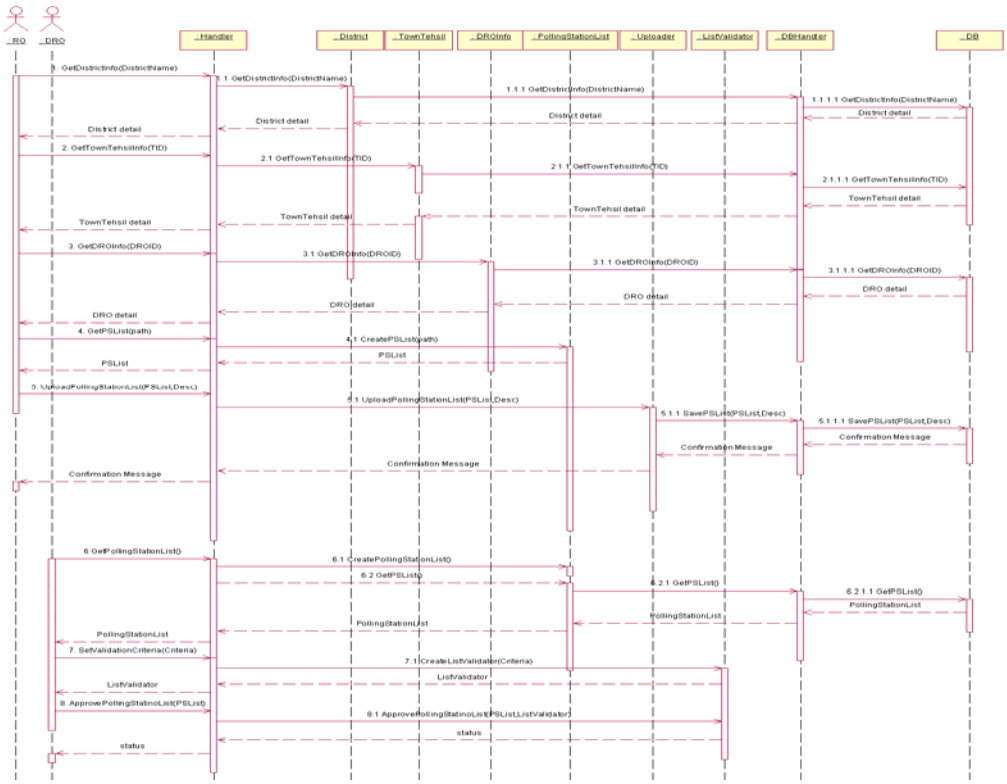


Figure 3.8.21

3.8.22 Sequence Diagram of UC_Approve_PO_APO_PollingStaffList (UC_22)

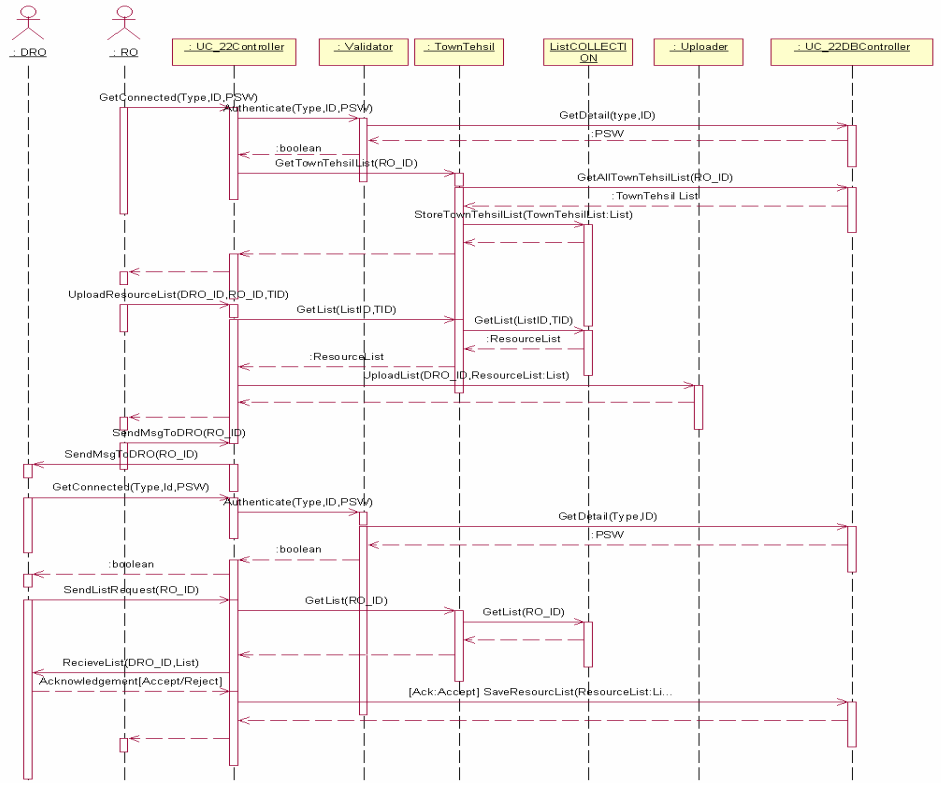


Figure 3.8.22

3.8.23 Sequence Diagram of UC_UpdateResultPOToRO (UC_23)

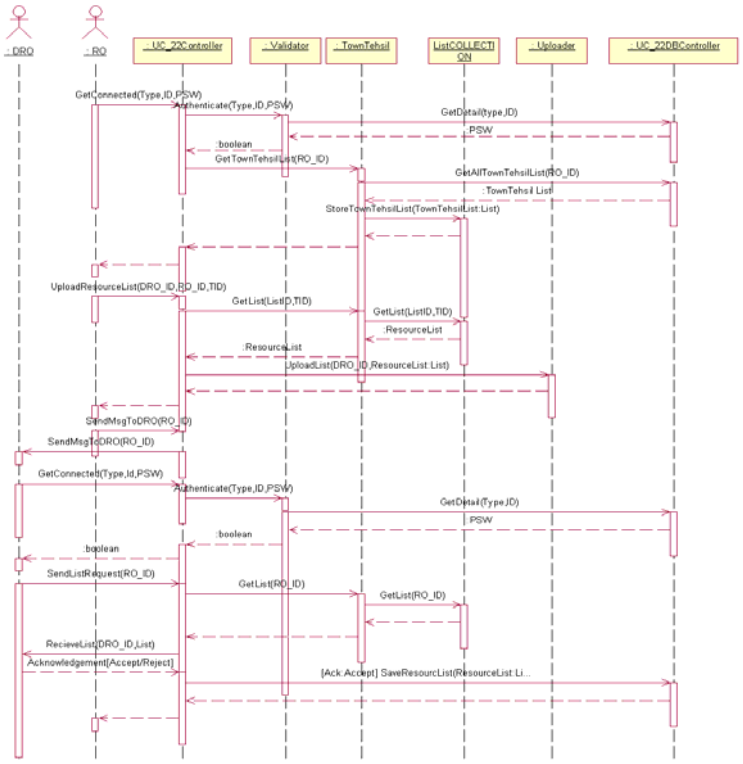


Figure 3.8.23

3.8.24 Sequence Diagram of UC_UpdateResultROToDROCEC (UC_24)

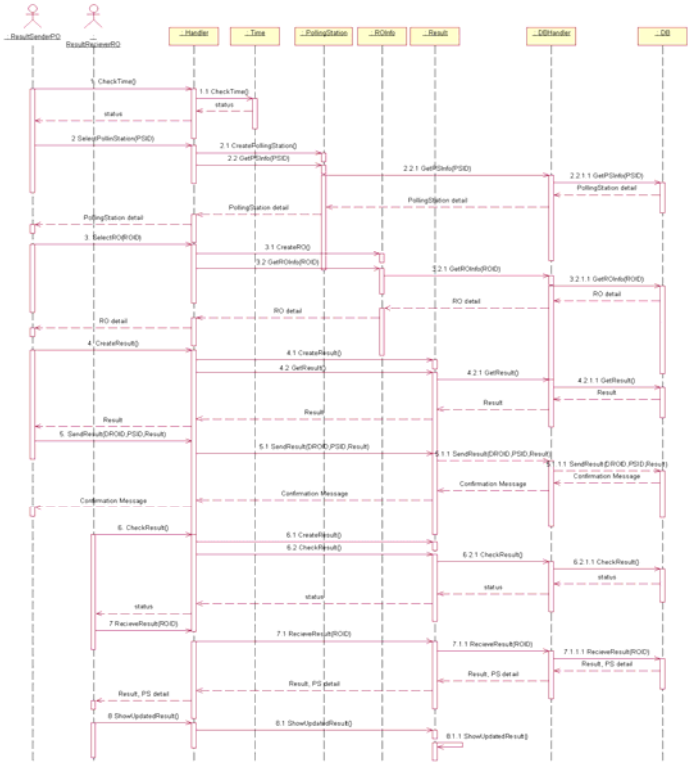


Figure 3.8.24

3.9 Collaboration Diagrams

3.9.1 Collaboration Diagram of UC_Prepare_ElectorlRoll (UC_1)

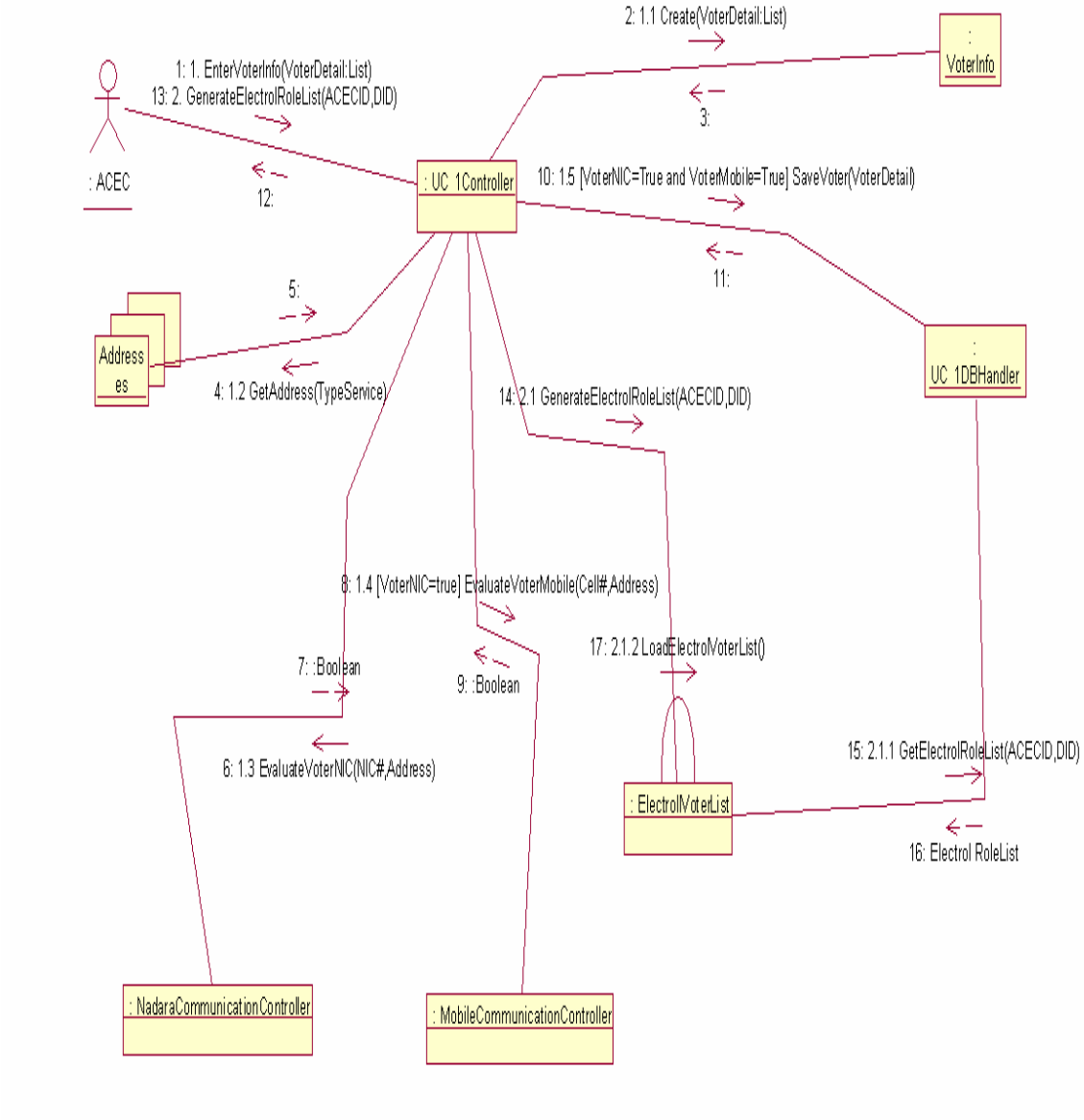


Figure 3.9.1

3.9.2 Collaboration Diagram of UC_Candidate_Nomination (UC_2)

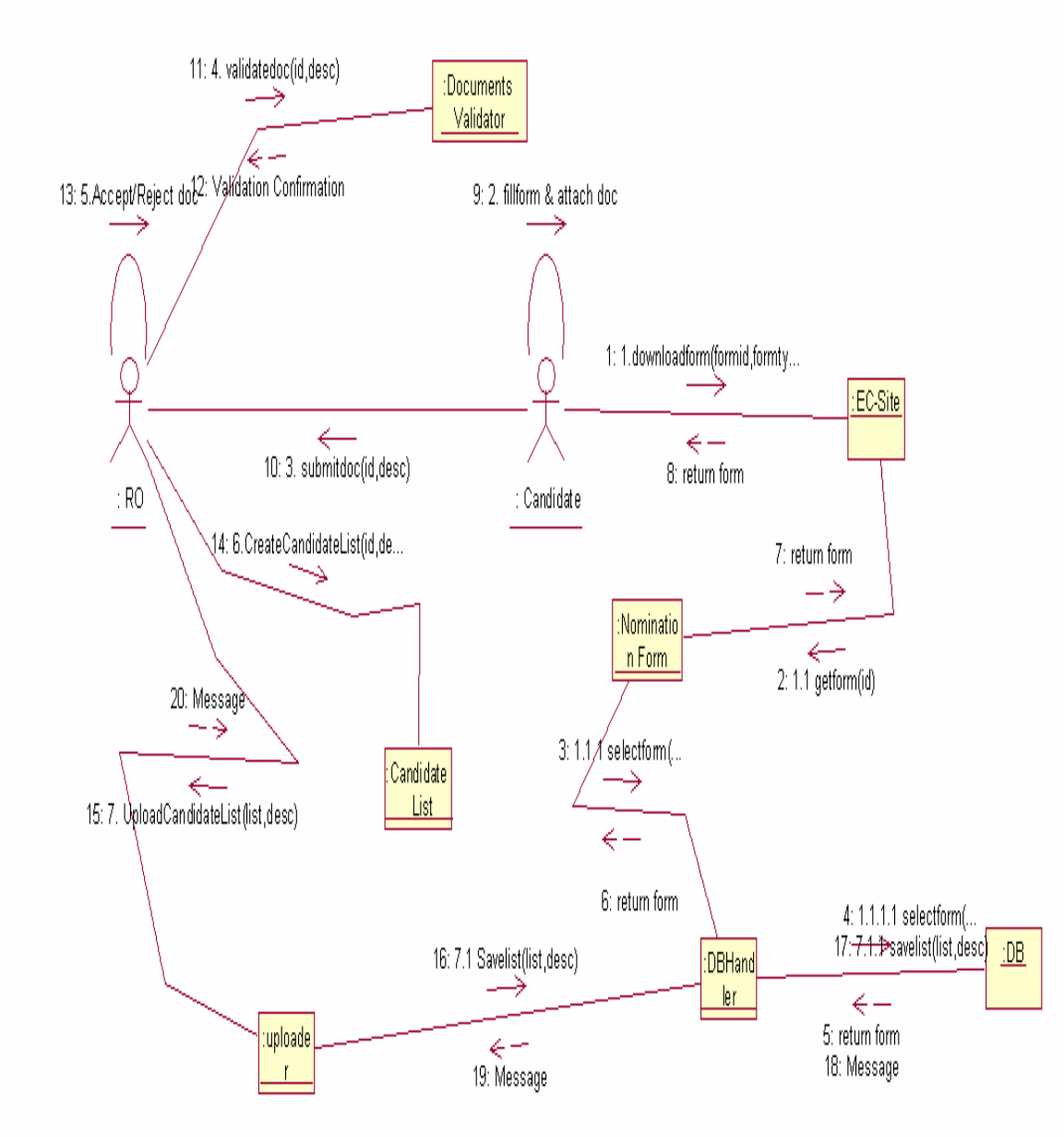


Figure 3.9.2

3.9.3 Collaboration Diagram of UC_Prepare_Symbol _CandidateList (UC_3)

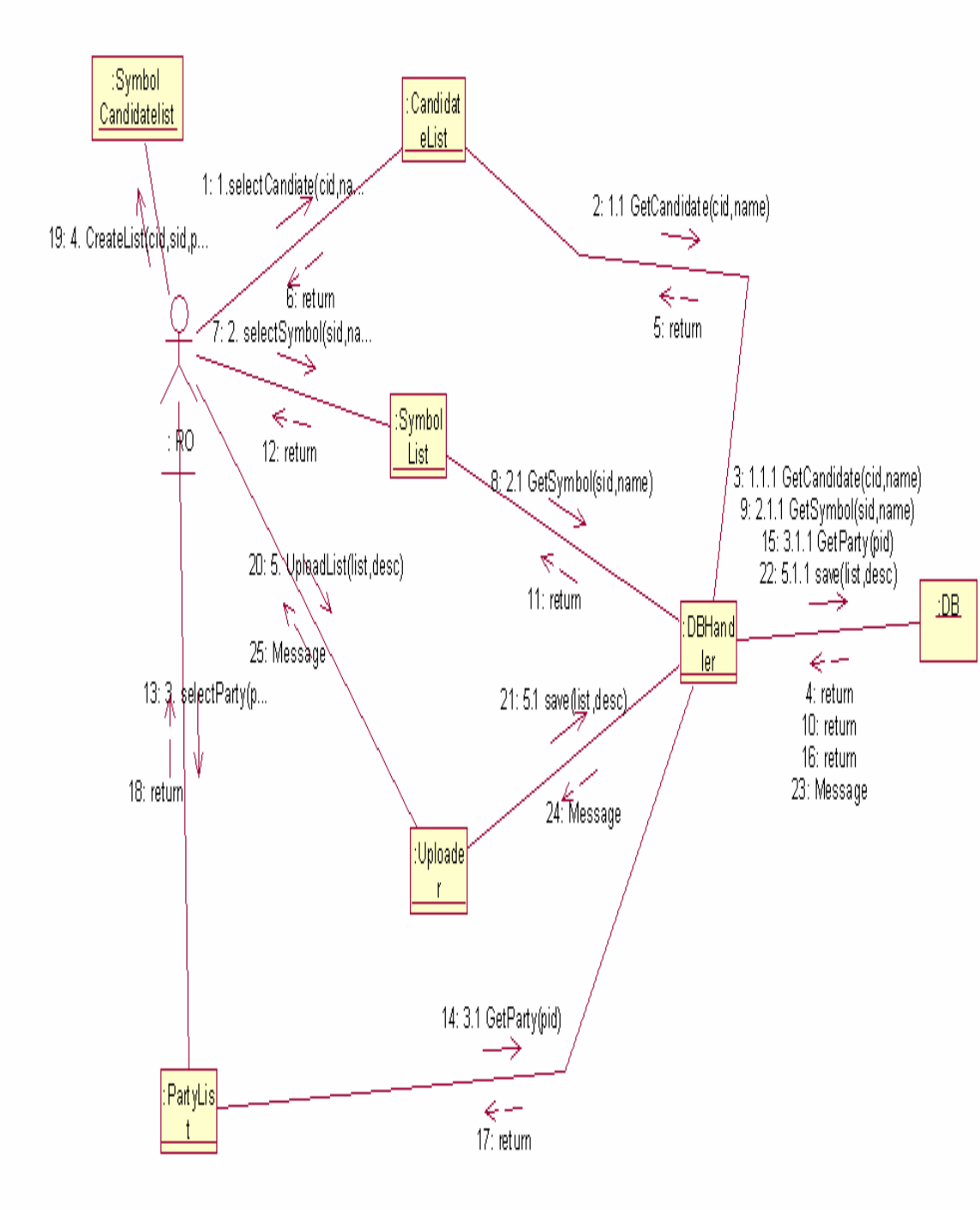


Figure 3.9.3

3.9.4 Collaboration Diagram of UC_VoterList_DRODistribution (UC_4)

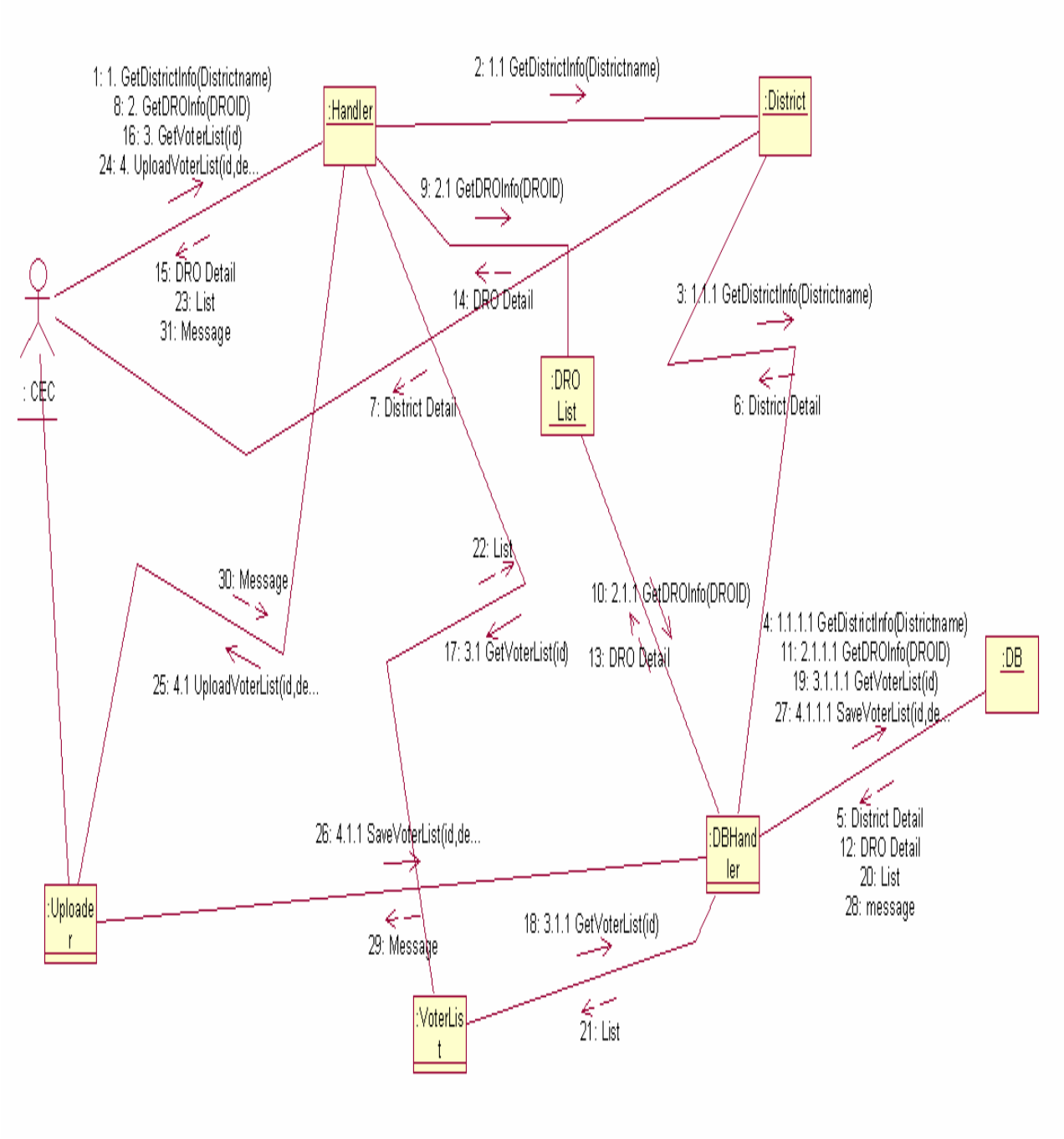


Figure 3.9.4

3.9.5 Collaboration Diagram of UC_VoterList_RODistribution (UC_5)

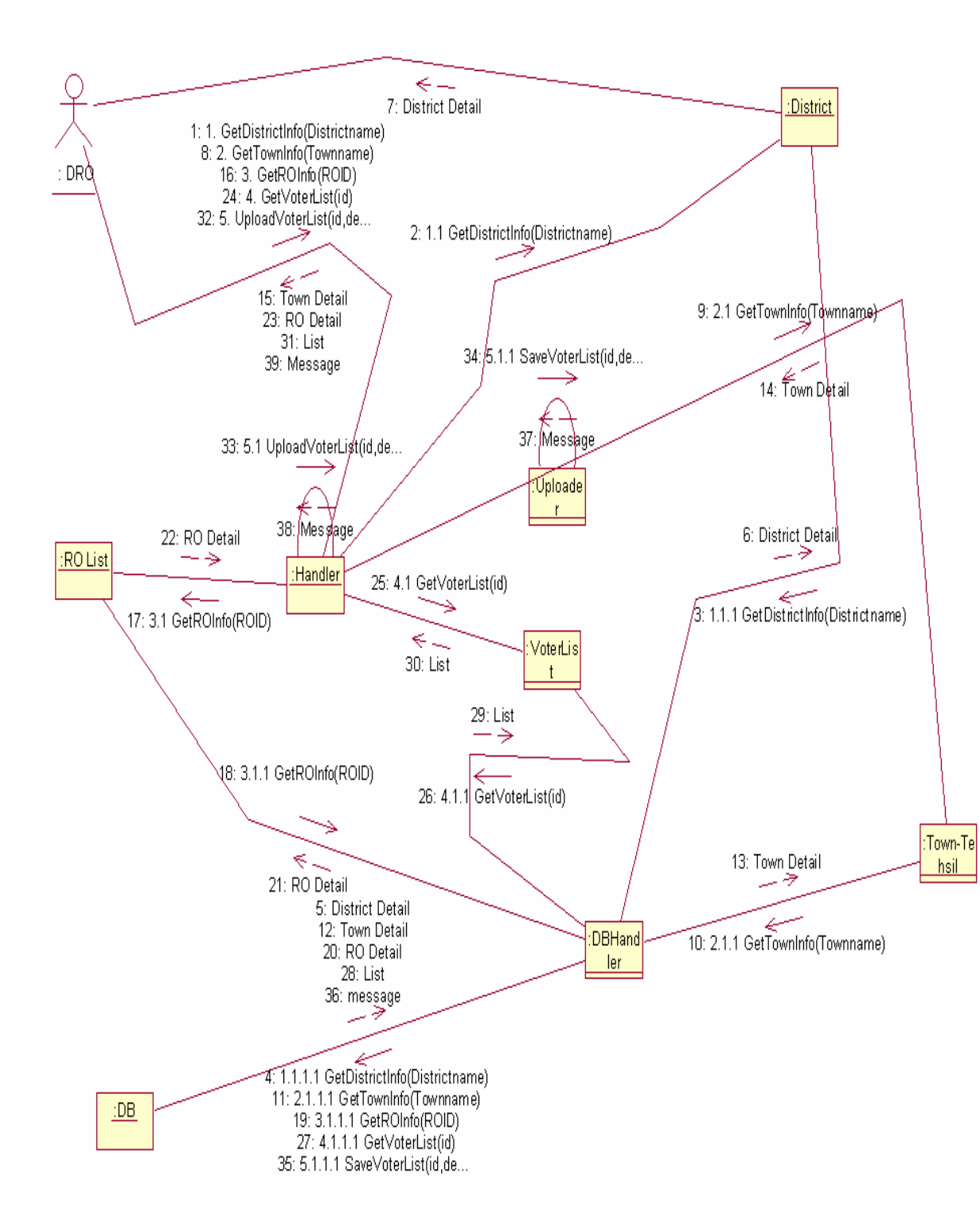


Figure 3.9.5

3.9.6 Collaboration Diagram of UC_VoterList_PODistribution (UC_6)

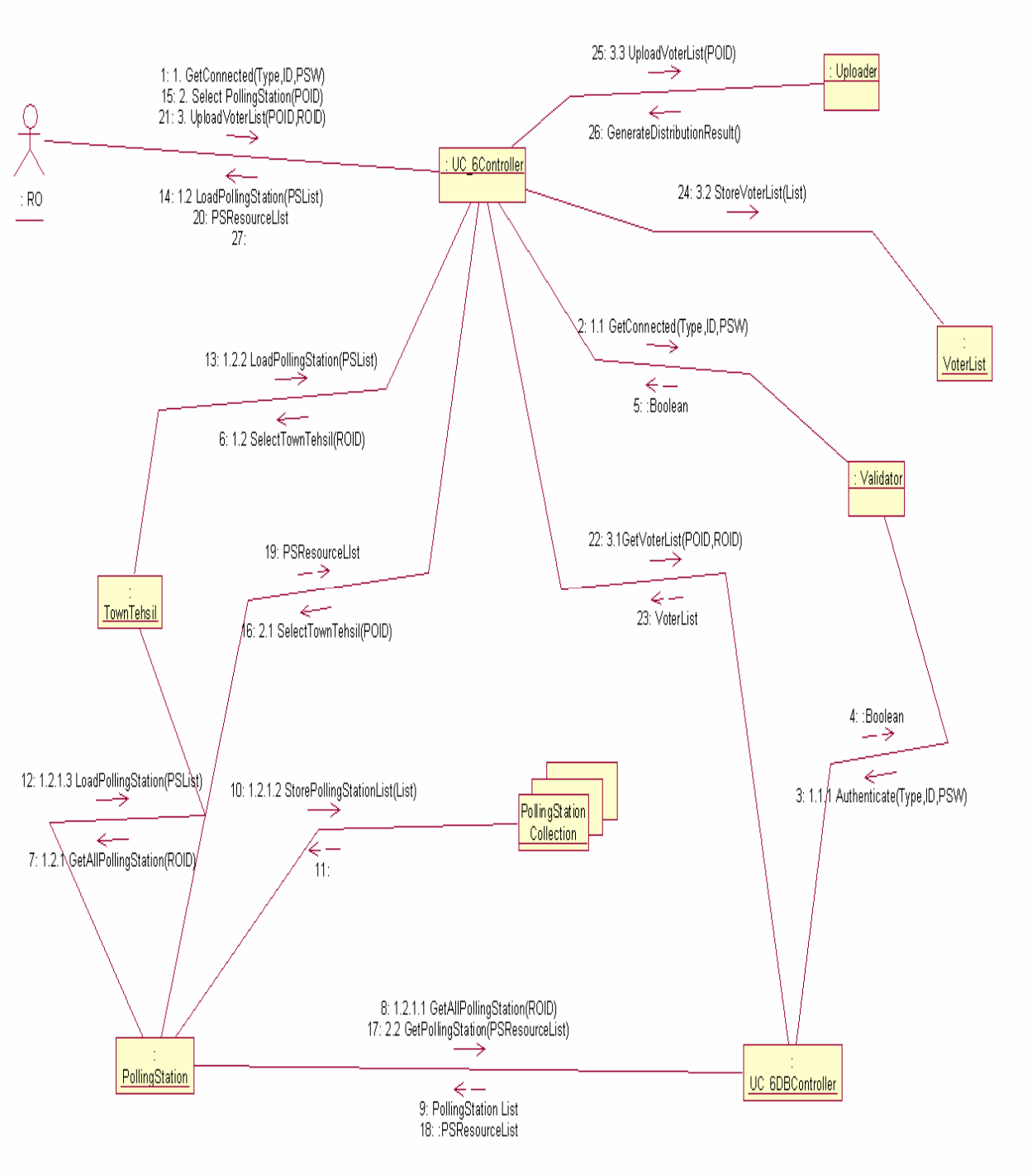


Figure 3.9.6

3.9.7 Collaboration Diagram of UC_Validate_Voter (UC_7)

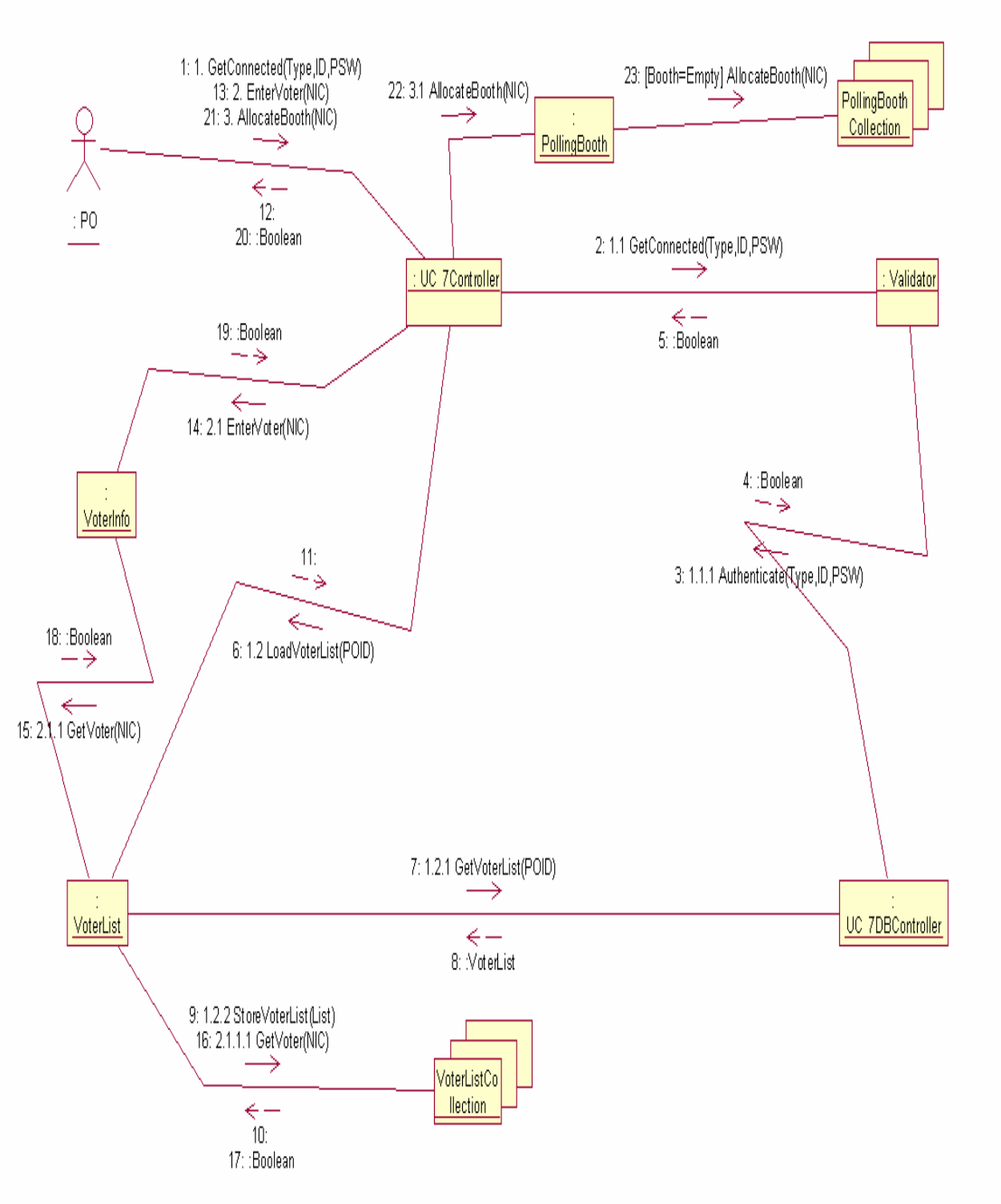


Figure 3.9.7

3.9.8 Collaboration Diagram of UC_Cast_Vote (UC_8)

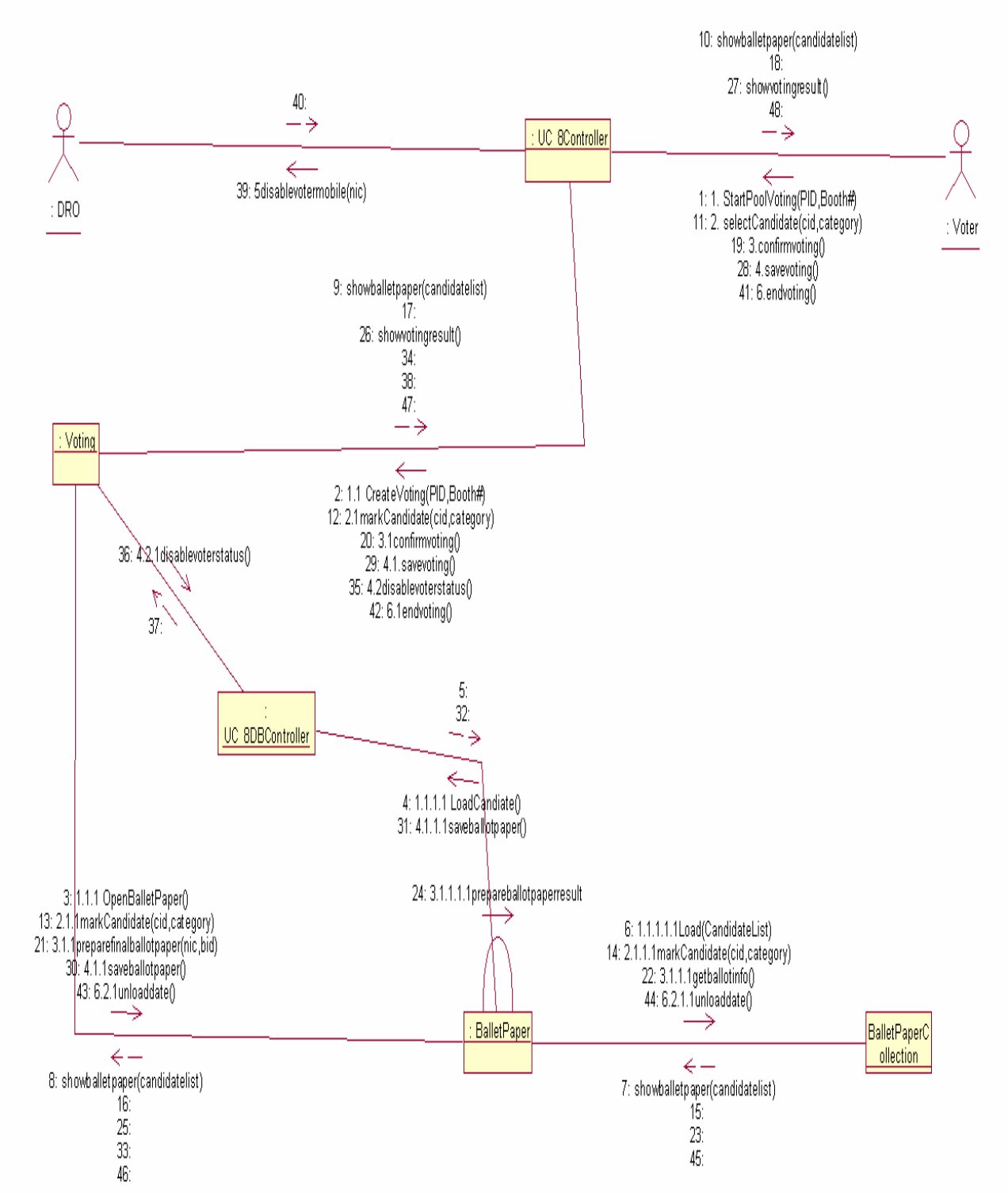


Figure 3.9.8

3.9.9 Collaboration Diagram of UC_Compile_Submit_POResult (UC_9)

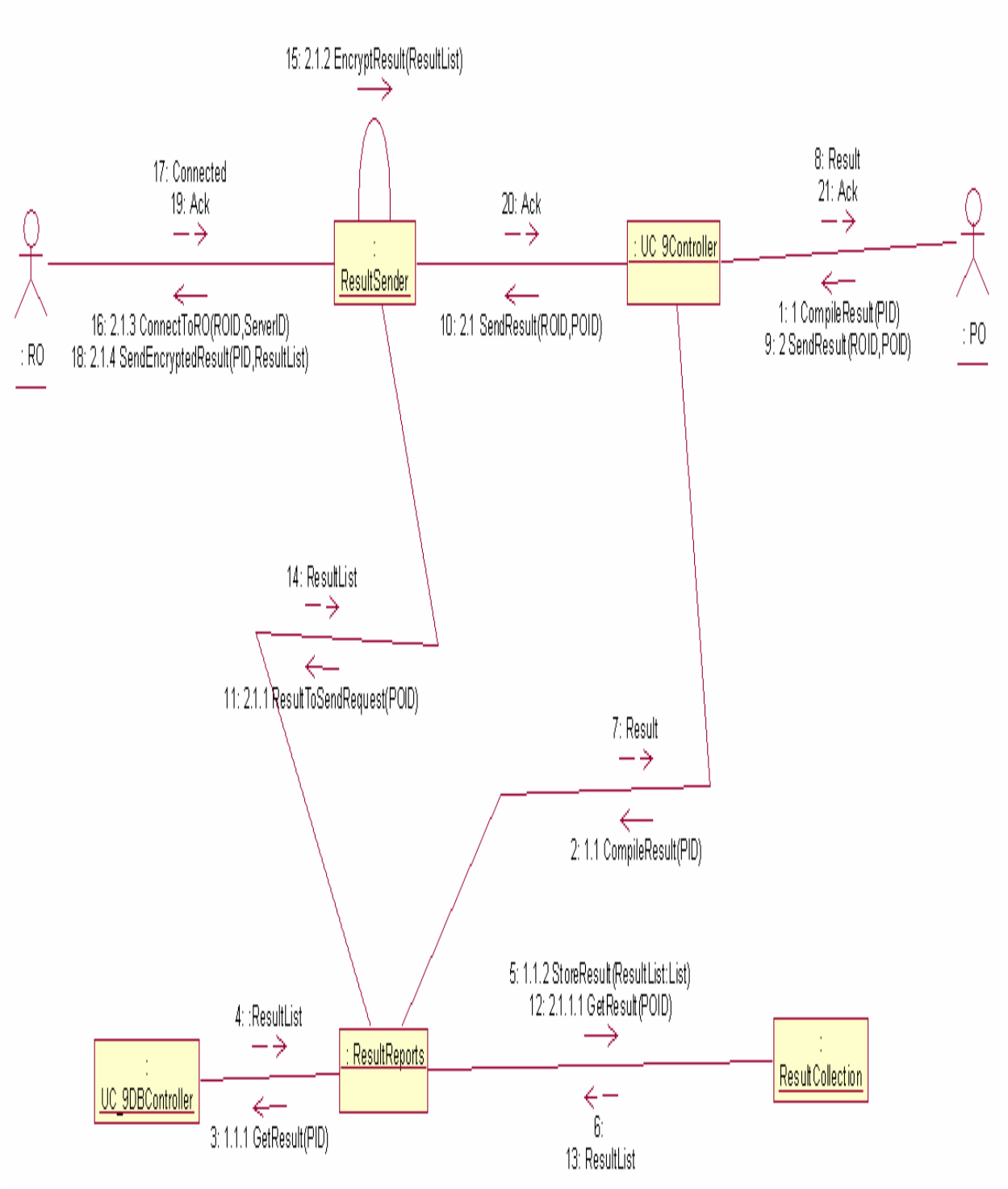


Figure 3.9.9

3.9.10 Collaboration Diagram of UC_Compile_ROResult (UC_10)

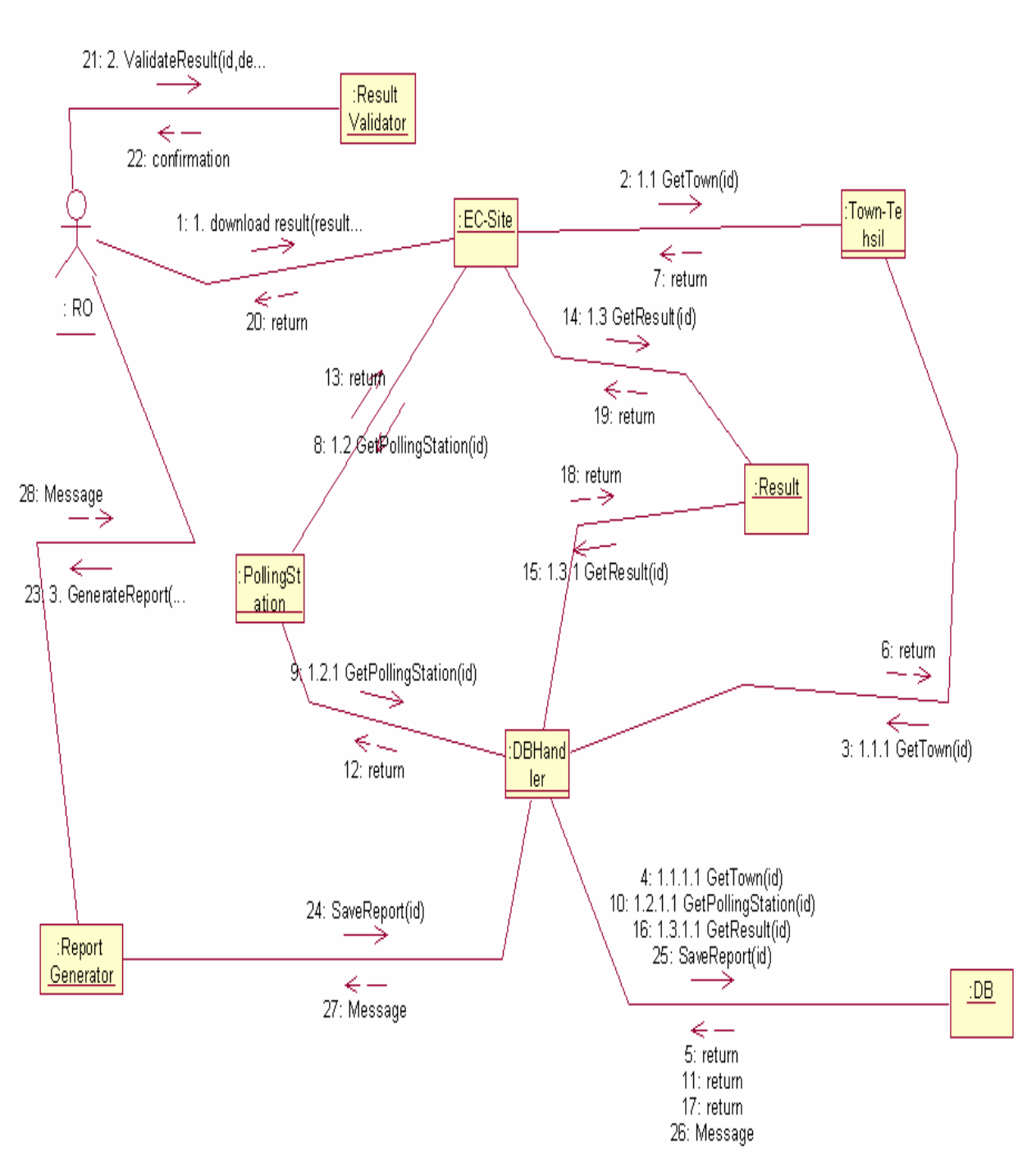


Figure 3.9.10

3.9.11 Collaboration Diagram of UC_Submit_ROResult (UC_11)

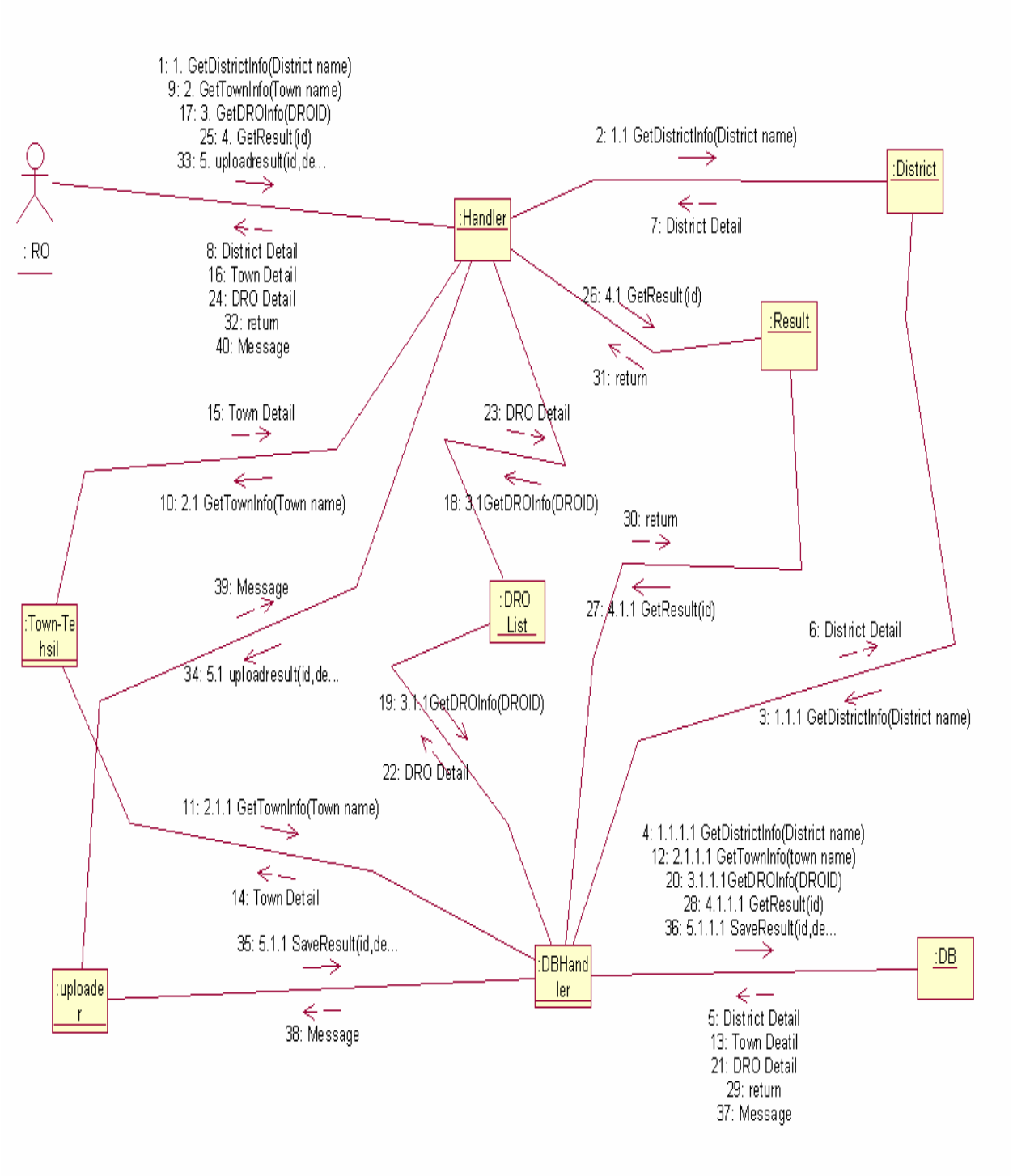


Figure 3.9.11

3.9.12 Collaboration Diagram of UC_Compile_DROResult (UC_12)

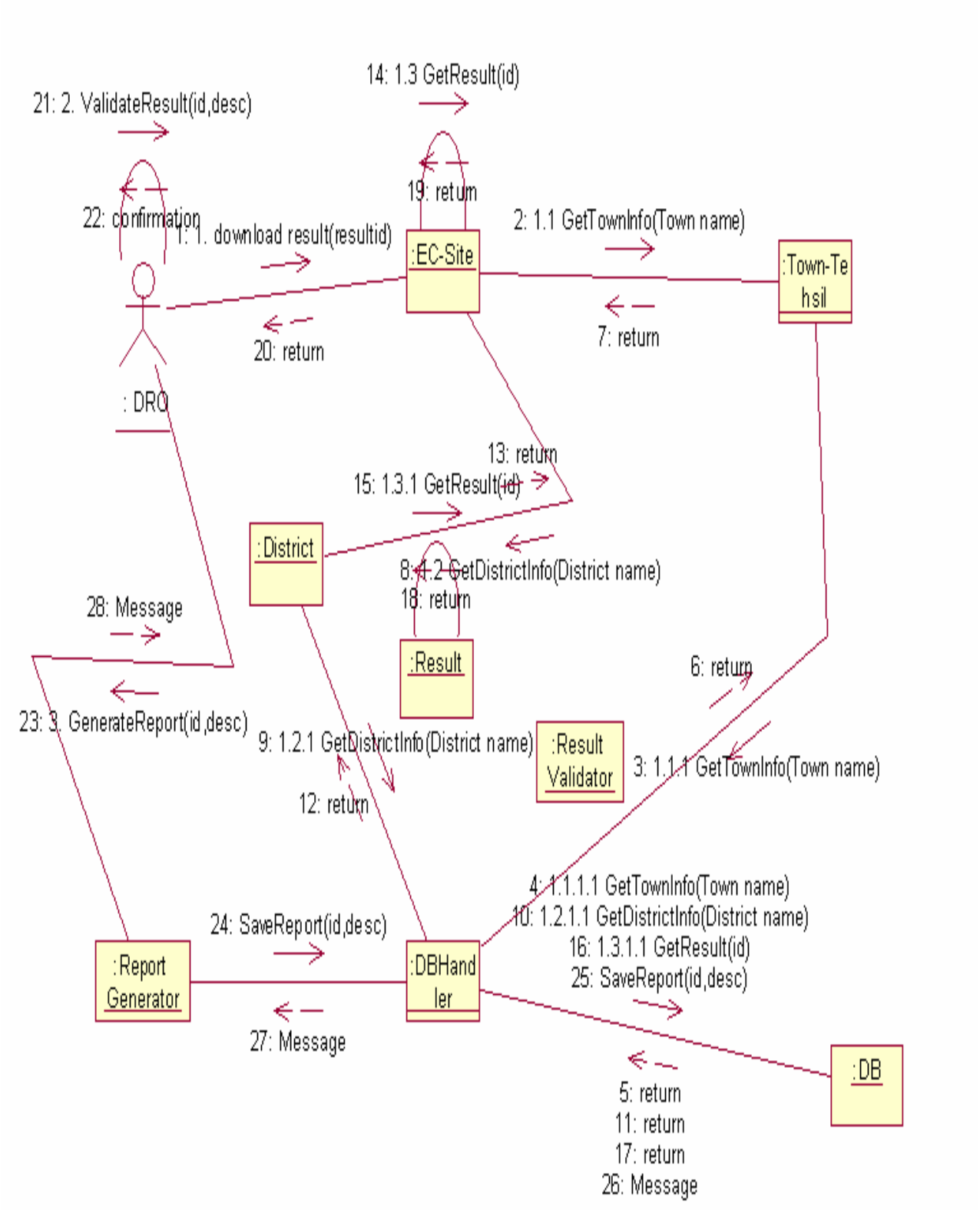


Figure 3.9.12

3.9.13 Collaboration Diagram of UC_Submit_DROResult (UC_13)

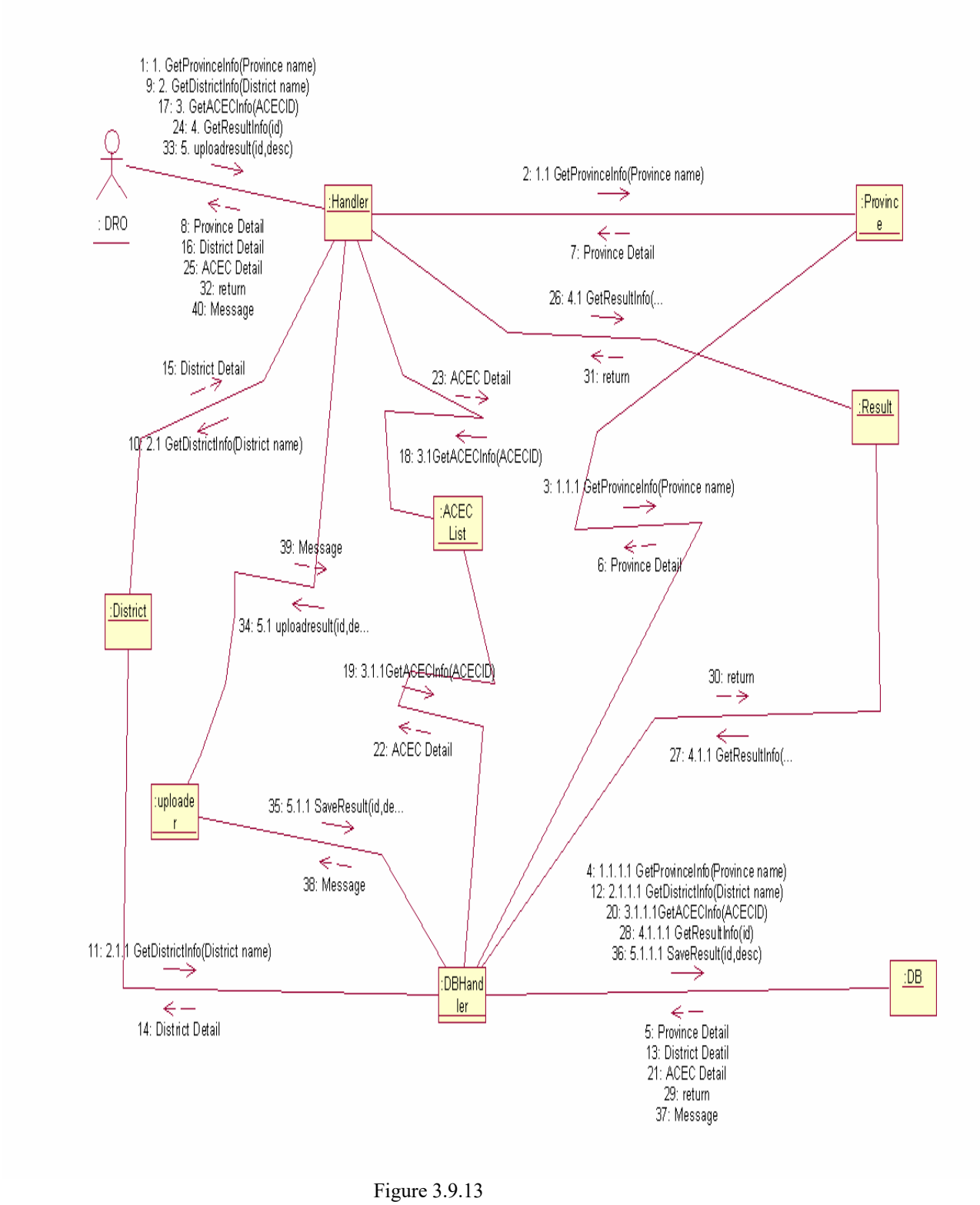


Figure 3.9.13

3.9.14 Collaboration Diagram of UC_Compile_FinalResult (UC_14)

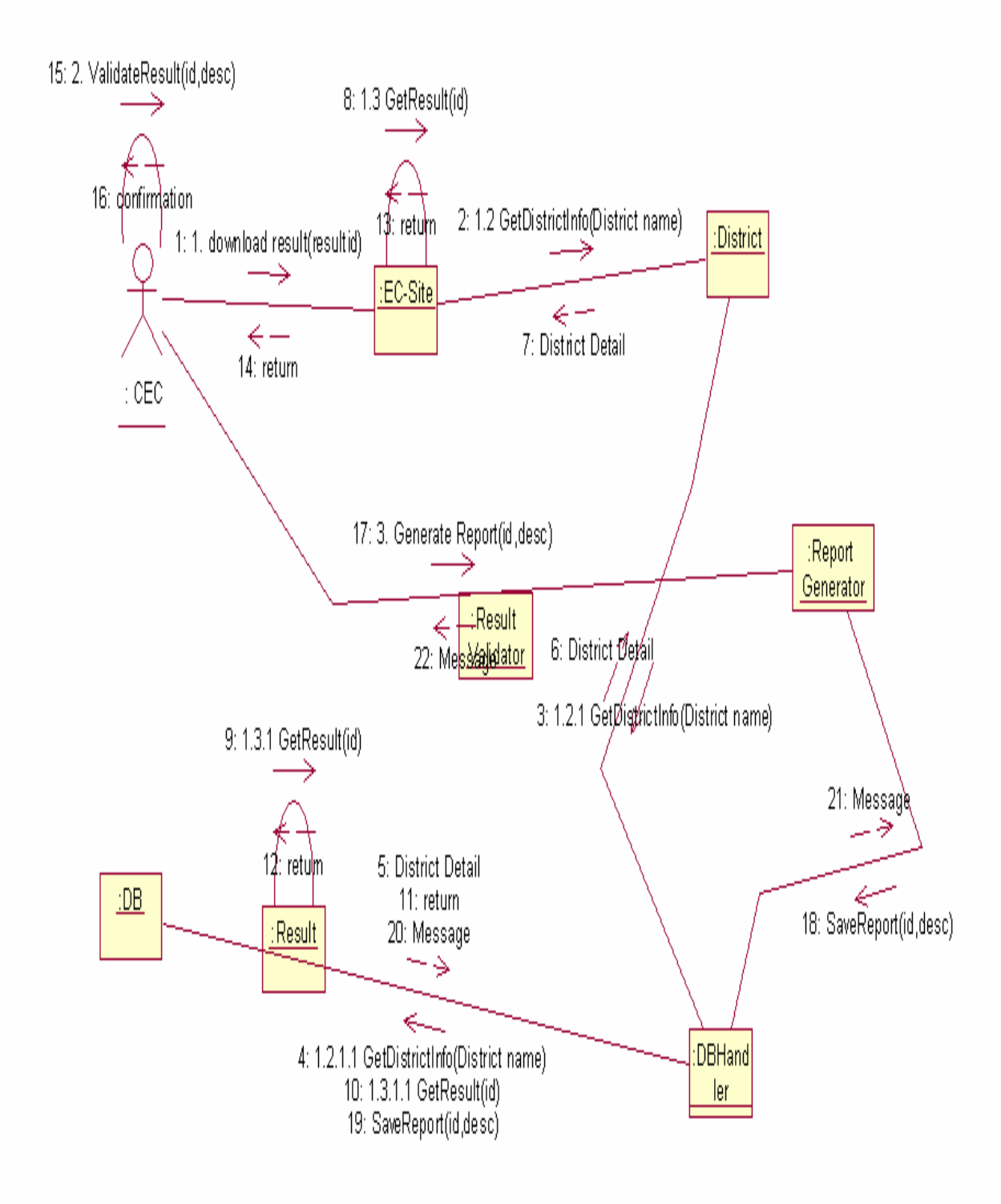


Figure 3.9.14

3.9.15 Collaboration Diagram of UC_Prepare_DROList (UC_15)

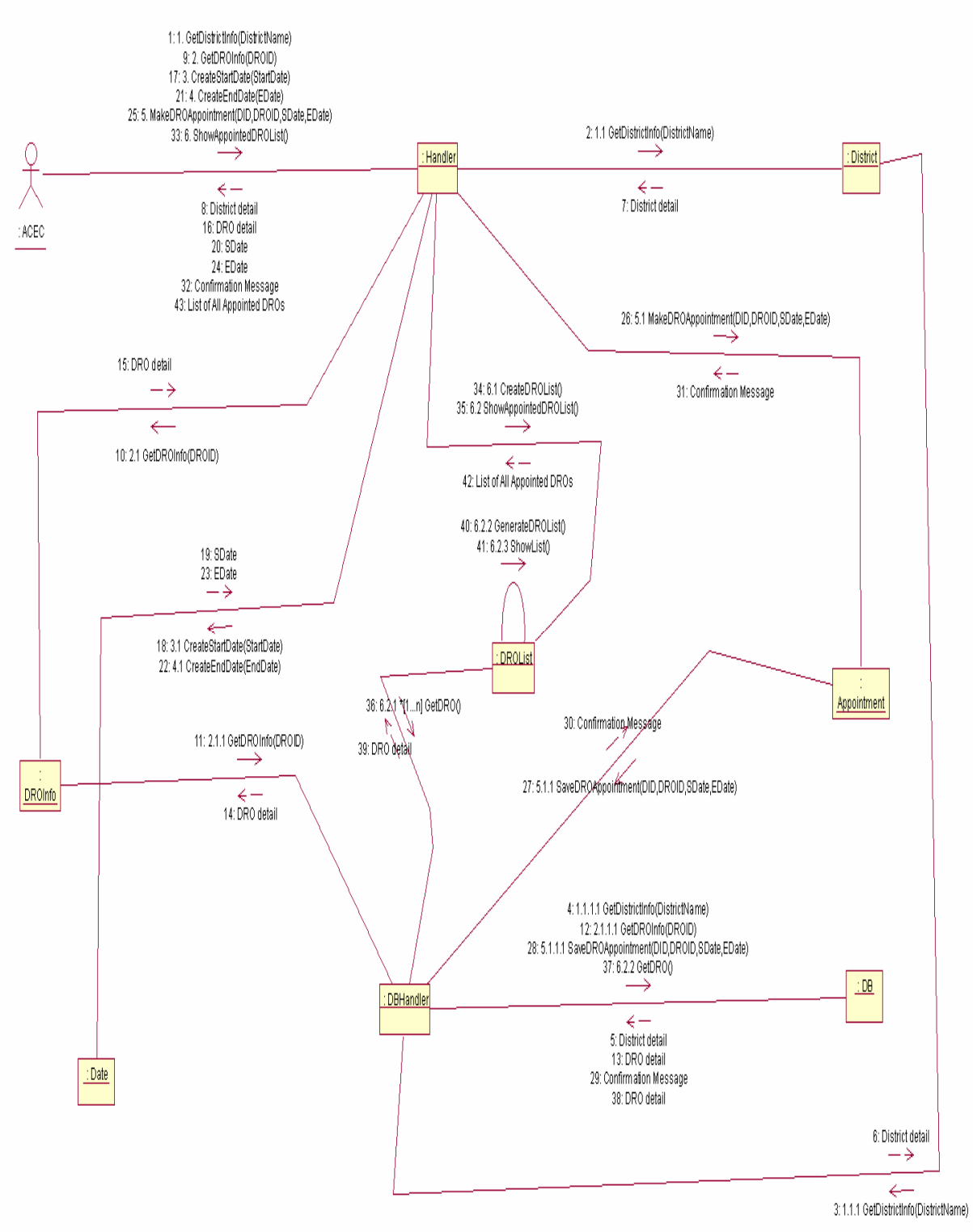


Figure 3.9.15

3.9.16 Collaboration Diagram of UC_Approve_DROList (UC_16)

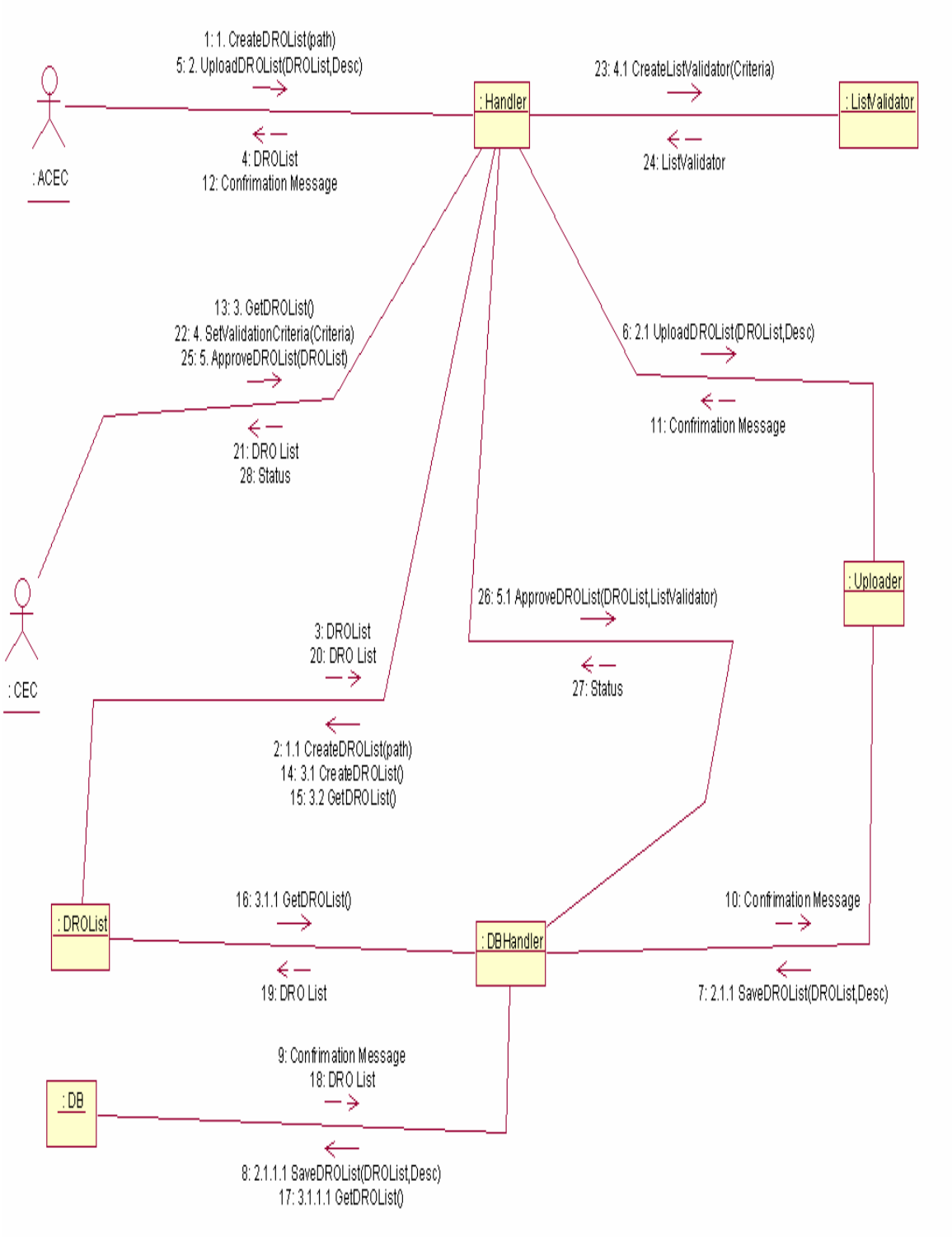


Figure 3.9.16

3.9.17 Collaboration Diagram of UC_Prepare_ROList (UC_17)

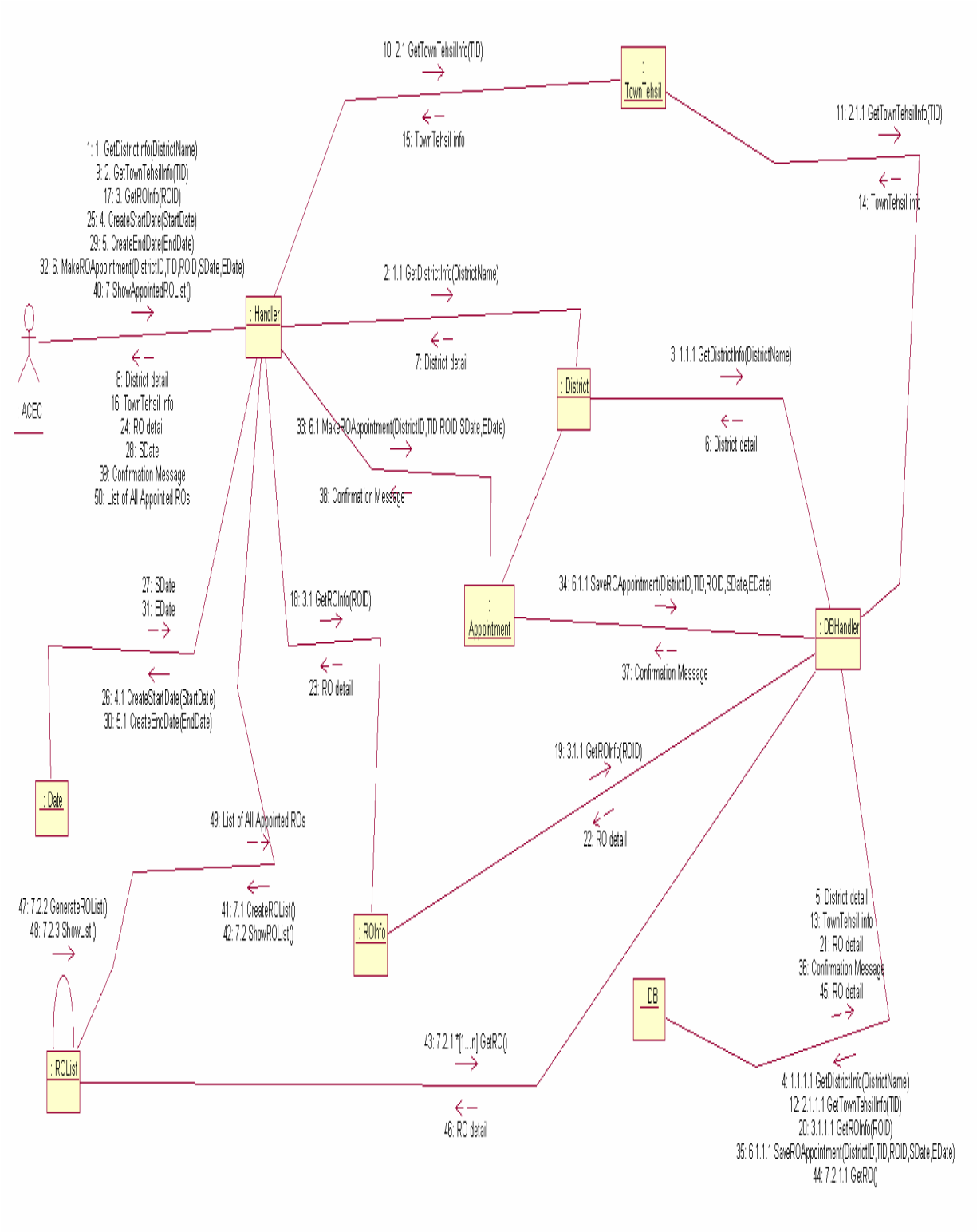


Figure 3.9.17

3.9.18 Collaboration Diagram of UC_Approve_ROList (UC_18)

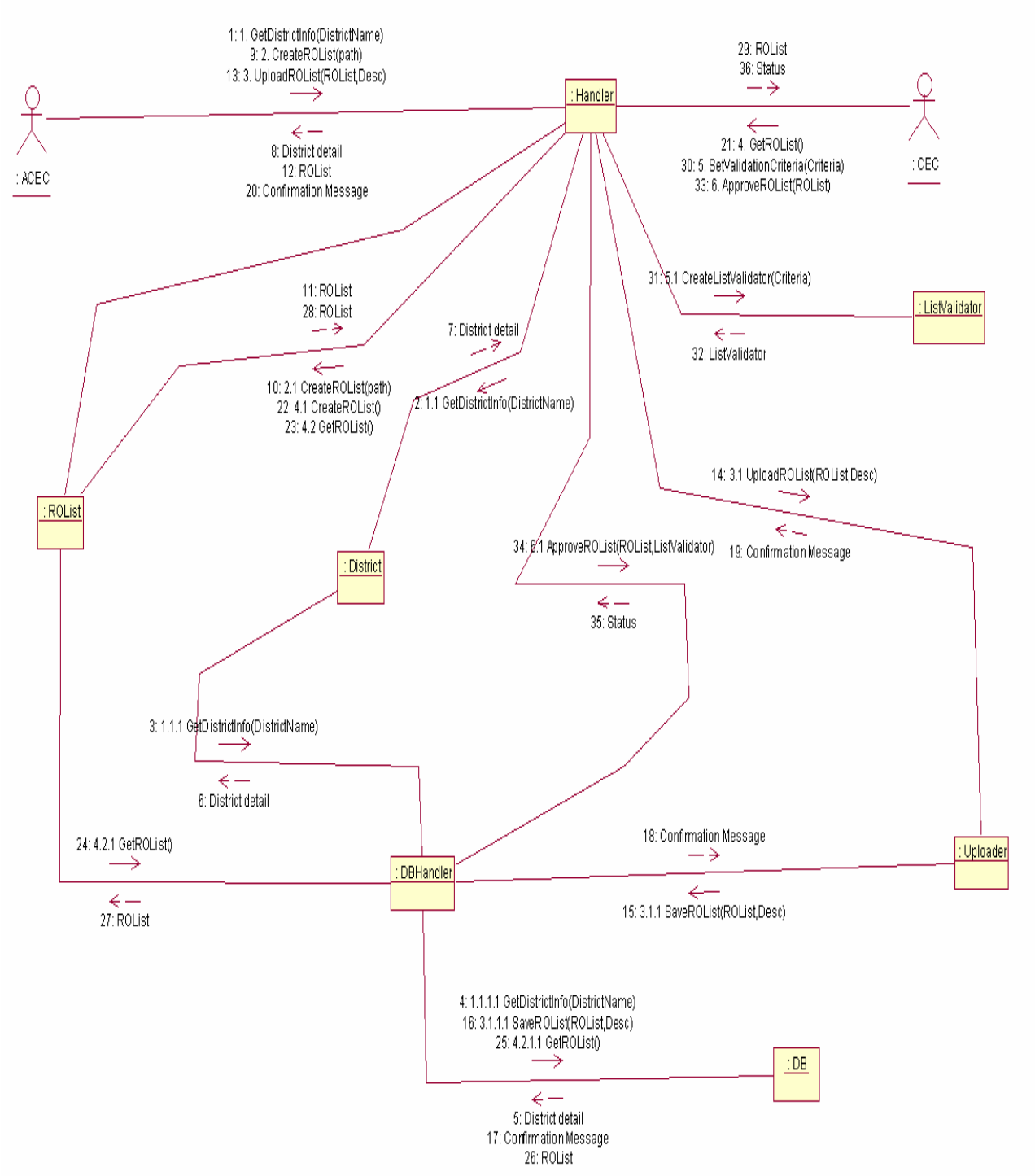


Figure 3.9.18

3.9.19 Collaboration Diagram of UC_Prepare_PollingStationList (UC_19)

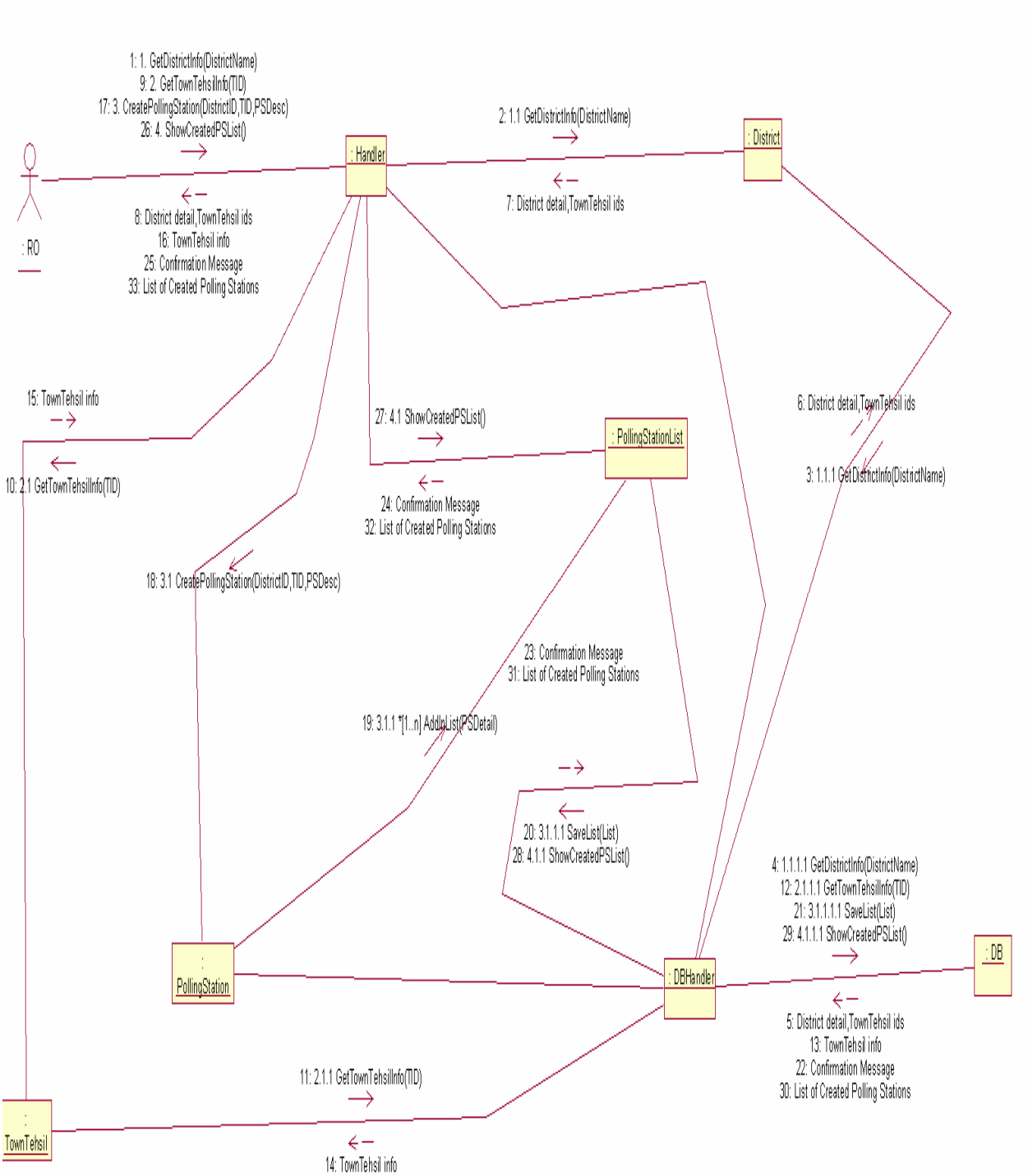


Figure 3.9.19

3.9.20 Collaboration Diagram of UC_Approve_PollingStationList (UC_20)

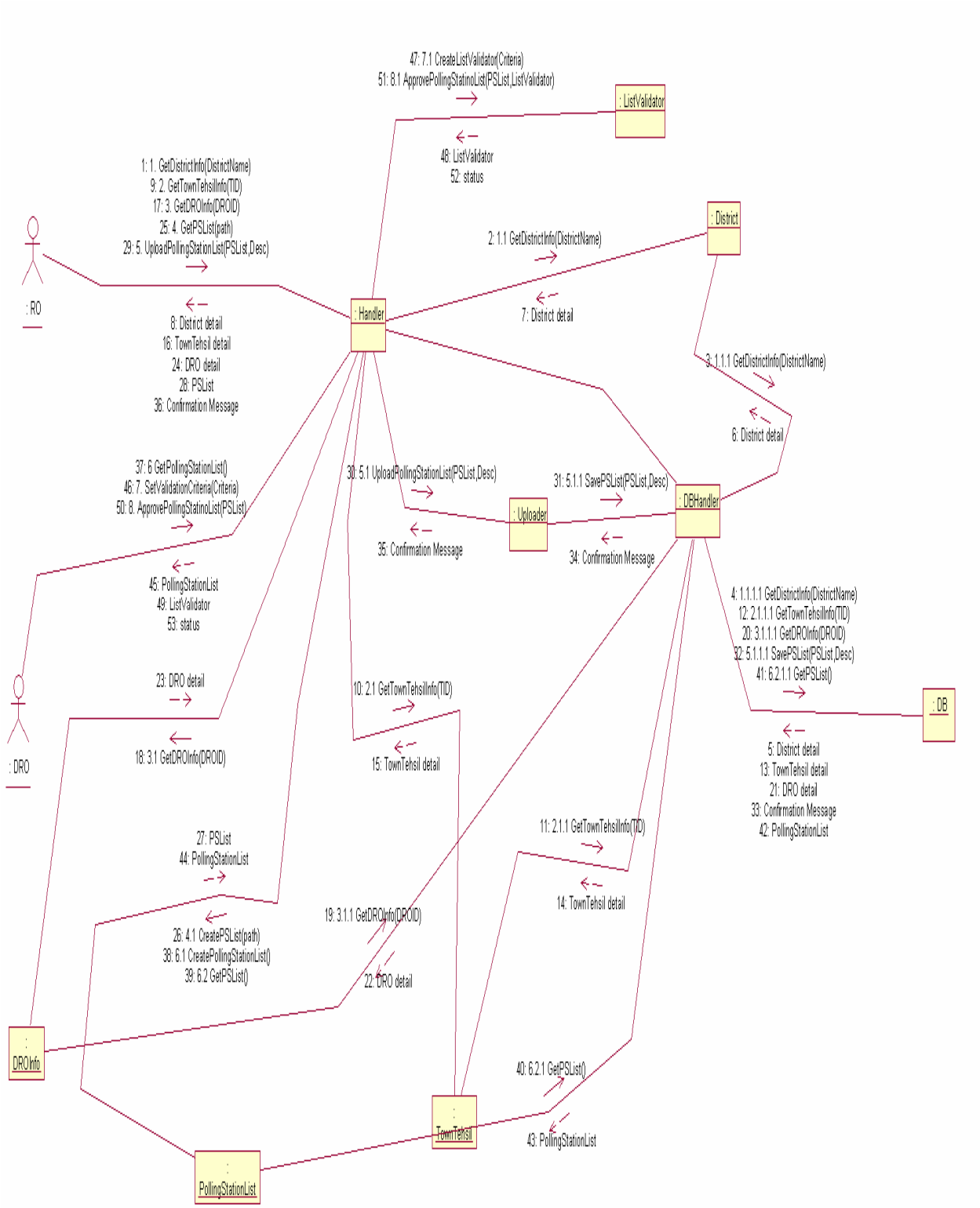


Figure 3.9.20

3.9.21 Collaboration Diagram of UC_Prepare_PO_APO_PollingStaffList (UC_21)

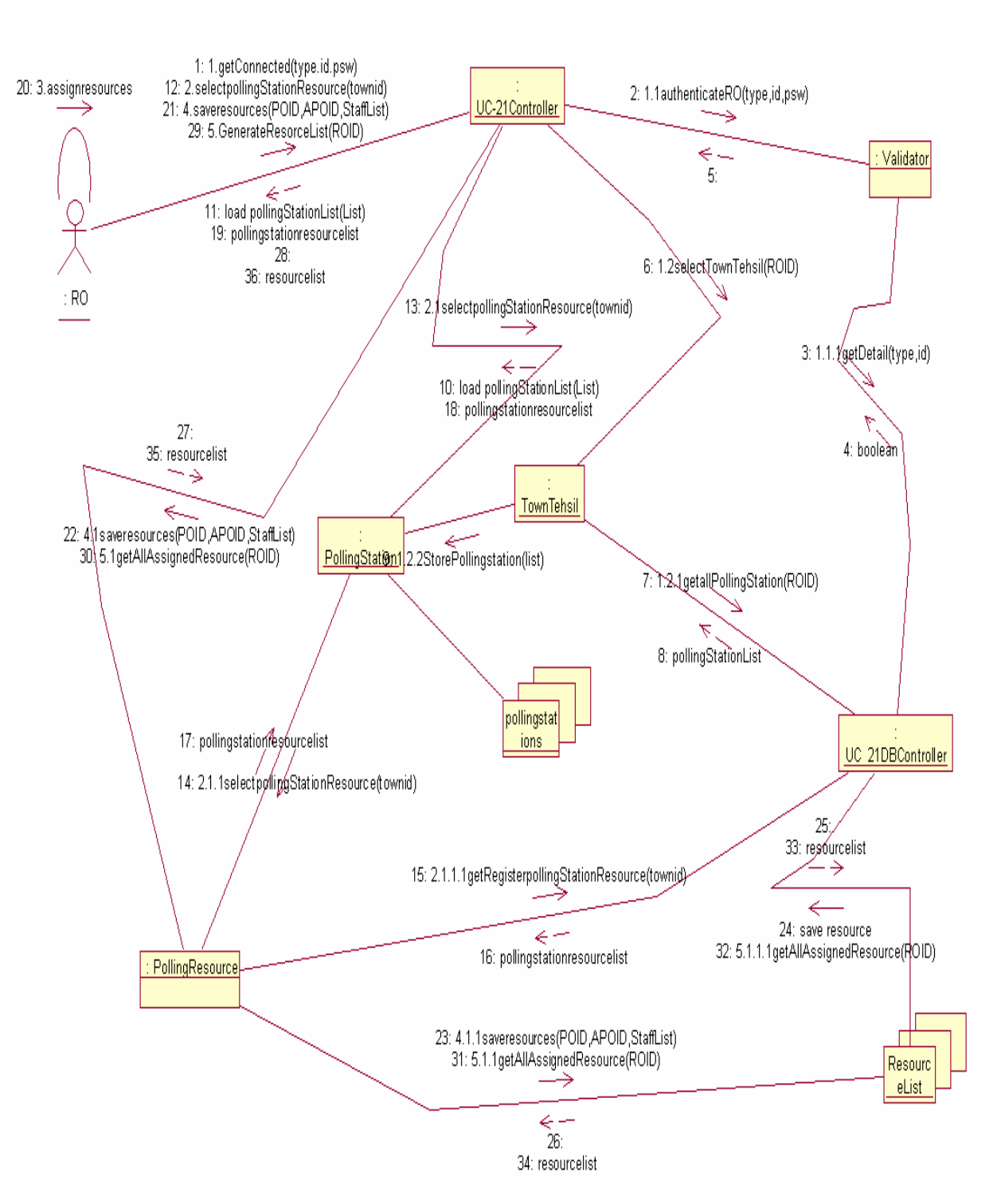


Figure 3.9.21

3.9.22 Collaboration Diagram of UC_Approve_PO_APO_PollingStaffList (UC_22)

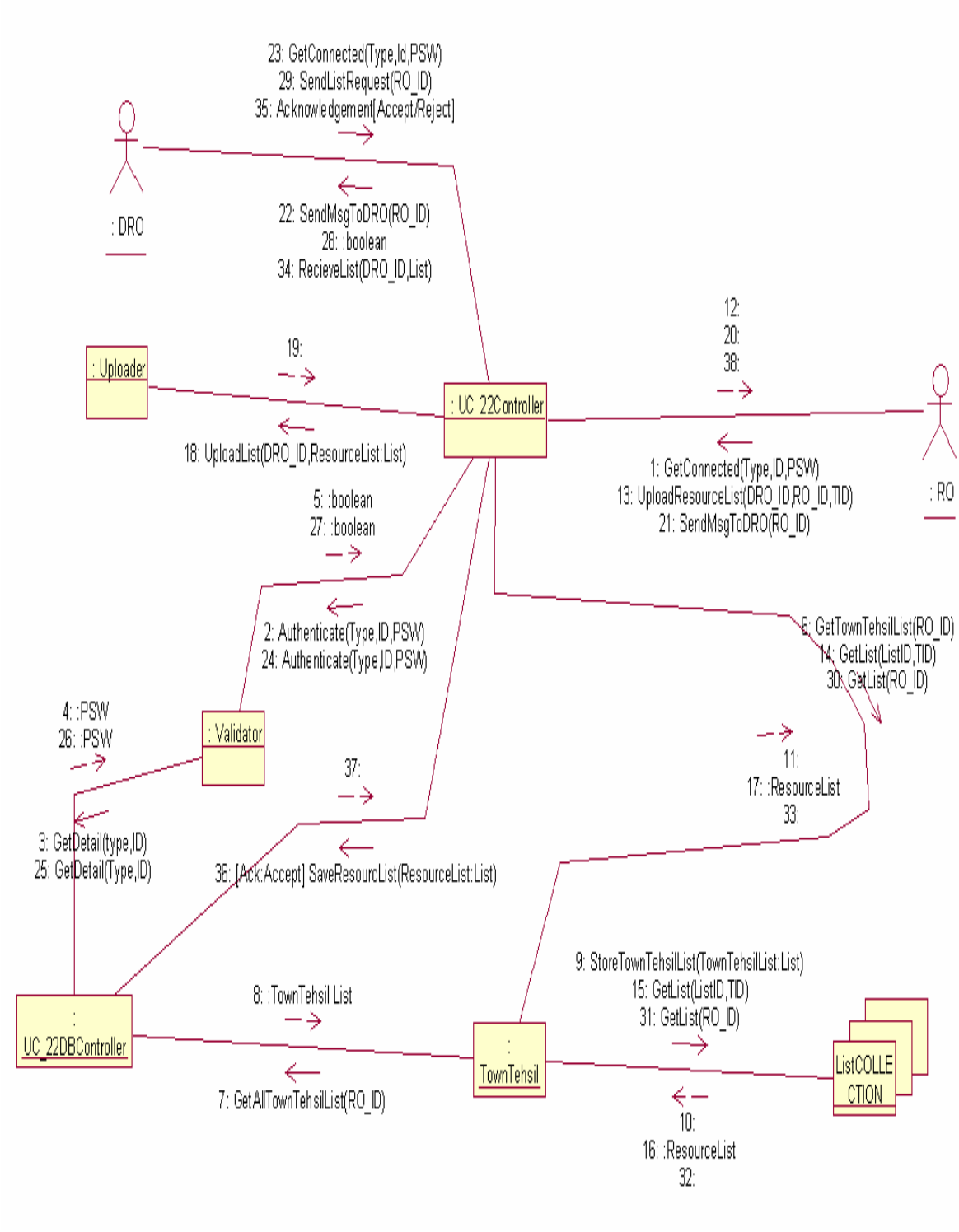


Figure 3.9.22

3.9.23 Collaboration Diagram of UC_UpdateResultPOToRO (UC_23)

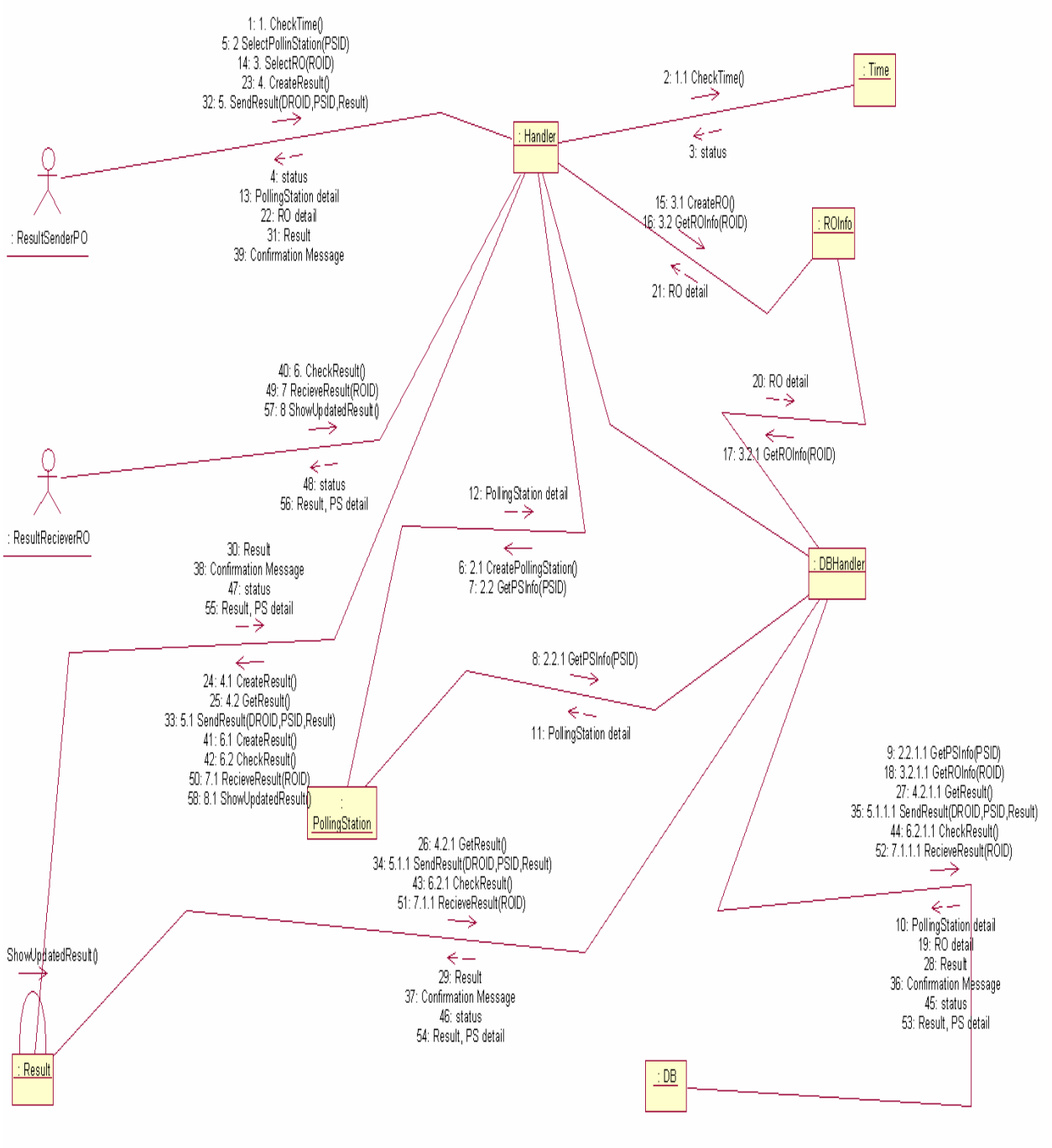


Figure 3.9.23

3.9.24 Collaboration Diagram of UC_UpdateResultROToDROCEC (UC_24)

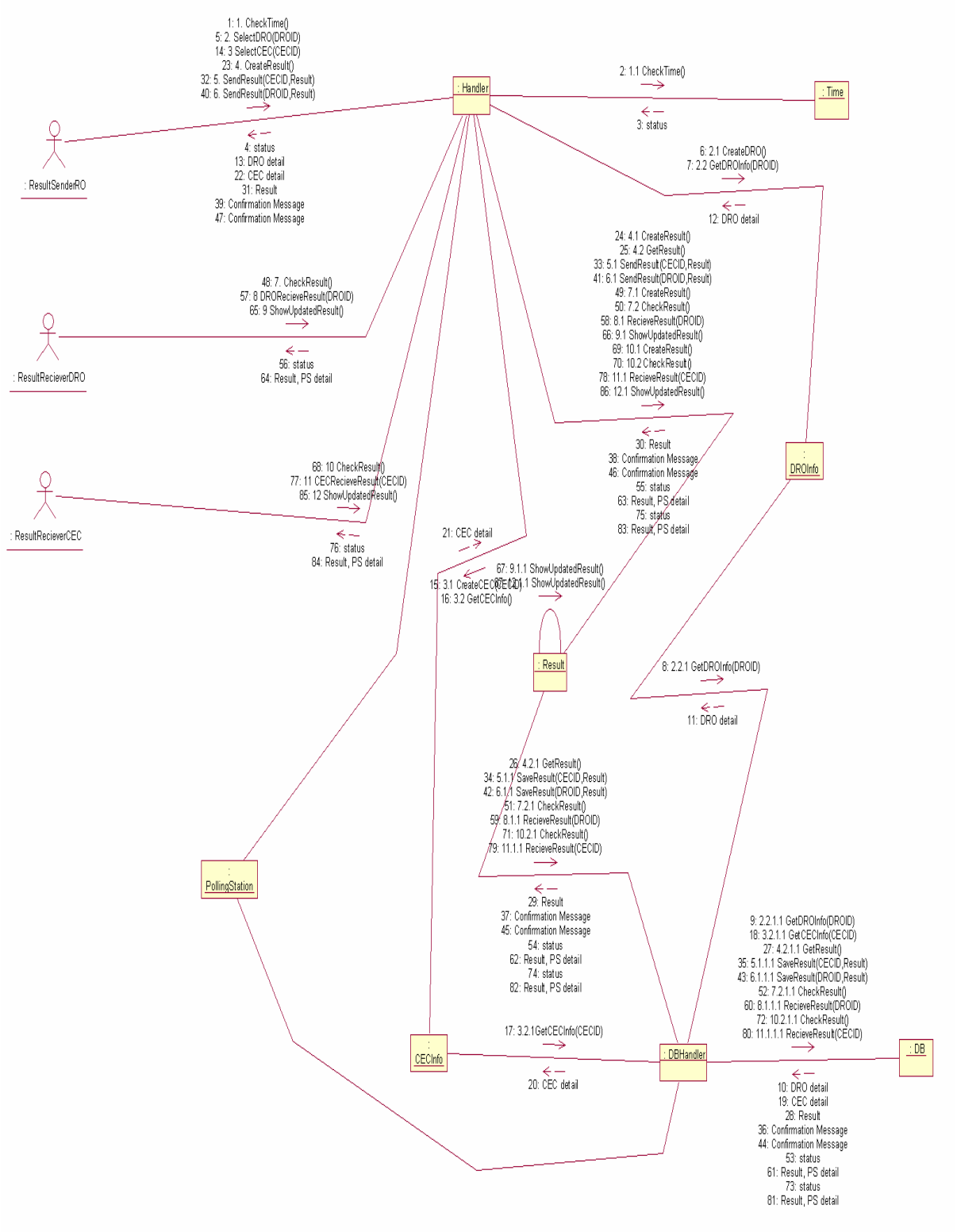


Figure 3.9.24

3.10 Design Class Diagrams

3.10.1 Design Class Diagram of UC_Prepare_ElectorlRoll (UC_1)

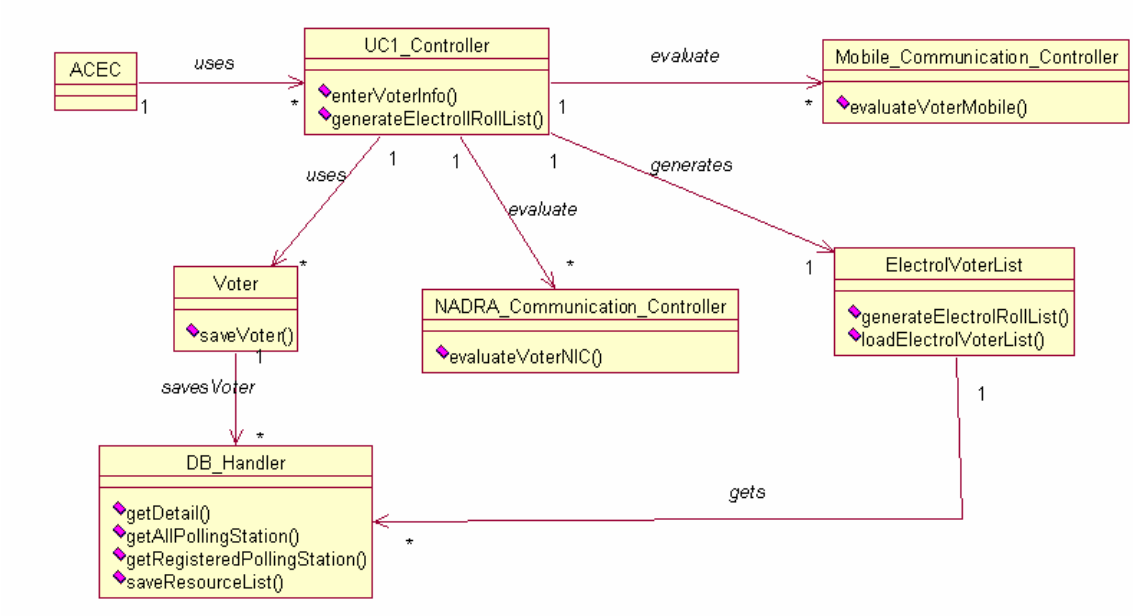


Figure 3.10.1

3.10.2 Design Class Diagram of UC_Candidate_Nomination (UC_2)

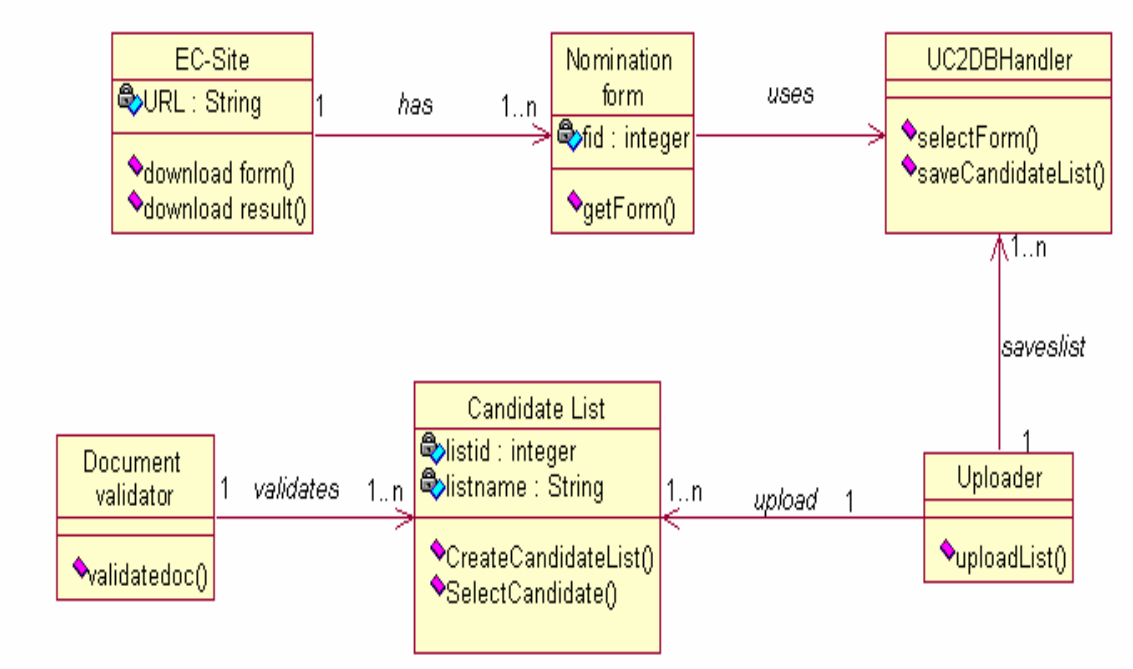


Figure 3.10.2

3.10.3 Design Class Diagram of UC_Prepare_Symbol _CandidateList (UC_3)

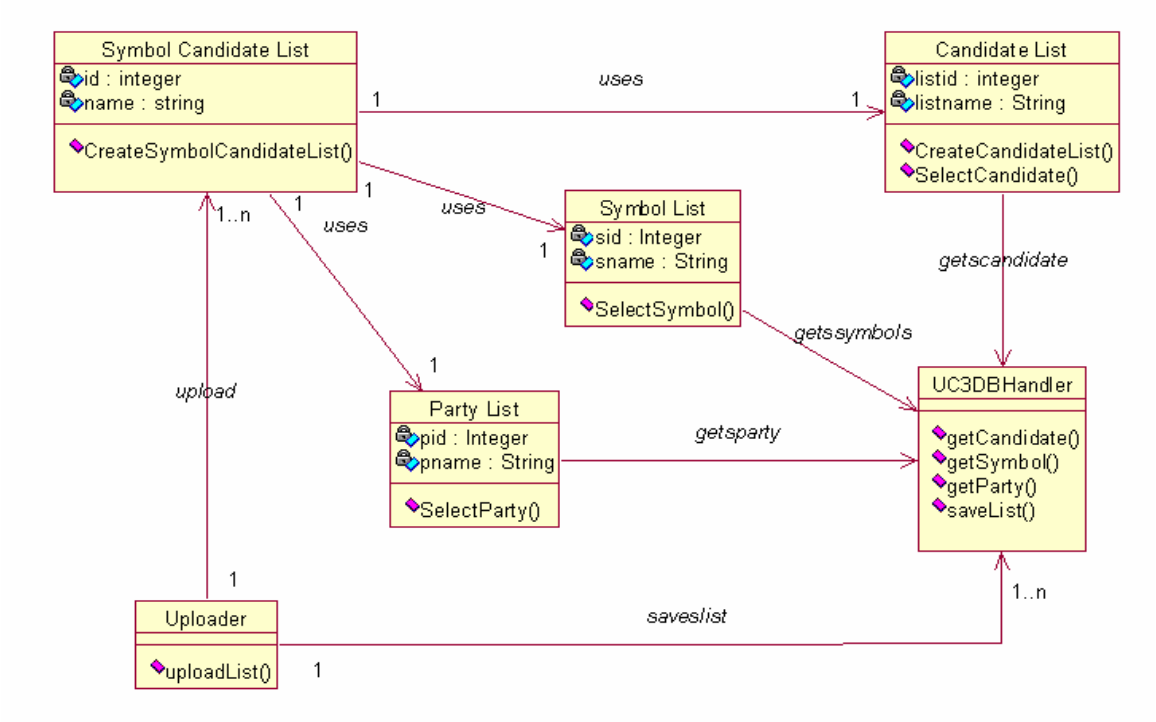


Figure 3.10.3

3.10.4 Design Class Diagram of UC_VoterList_DRODistribution (UC_4)

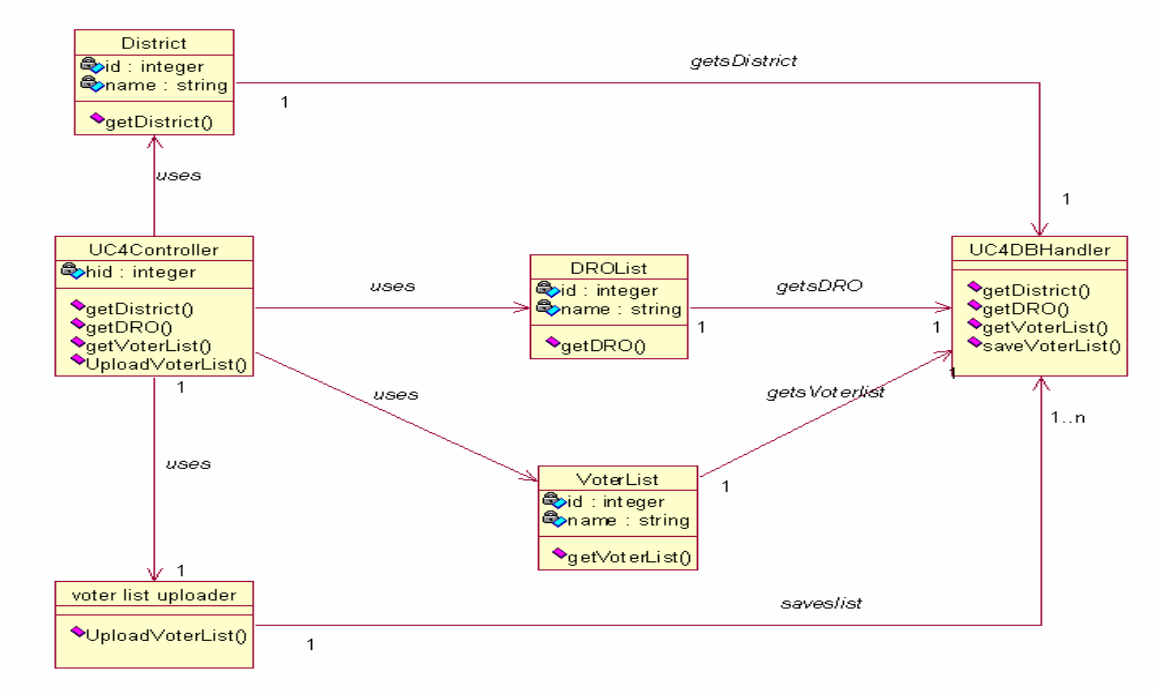


Figure 3.10.4

3.10.5 Design Class Diagram of UC_VoterList_RODistribution (UC_5)

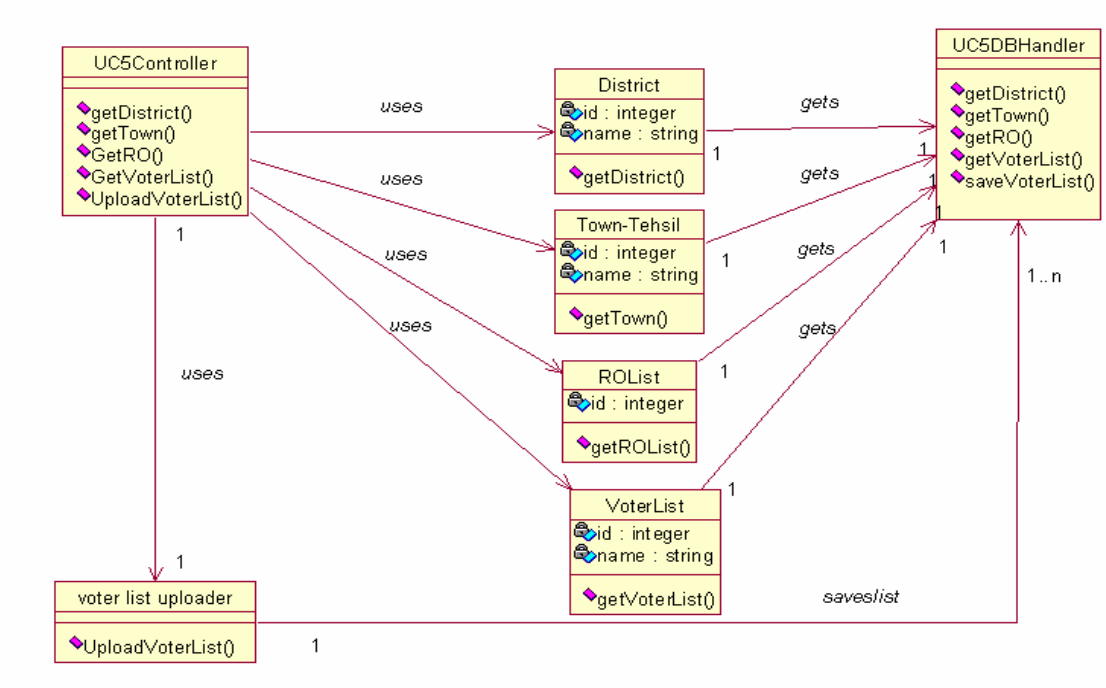


Figure 3.10.5

3.10.6 Design Class Diagram of UC_VoterList_PODistribution (UC_6)

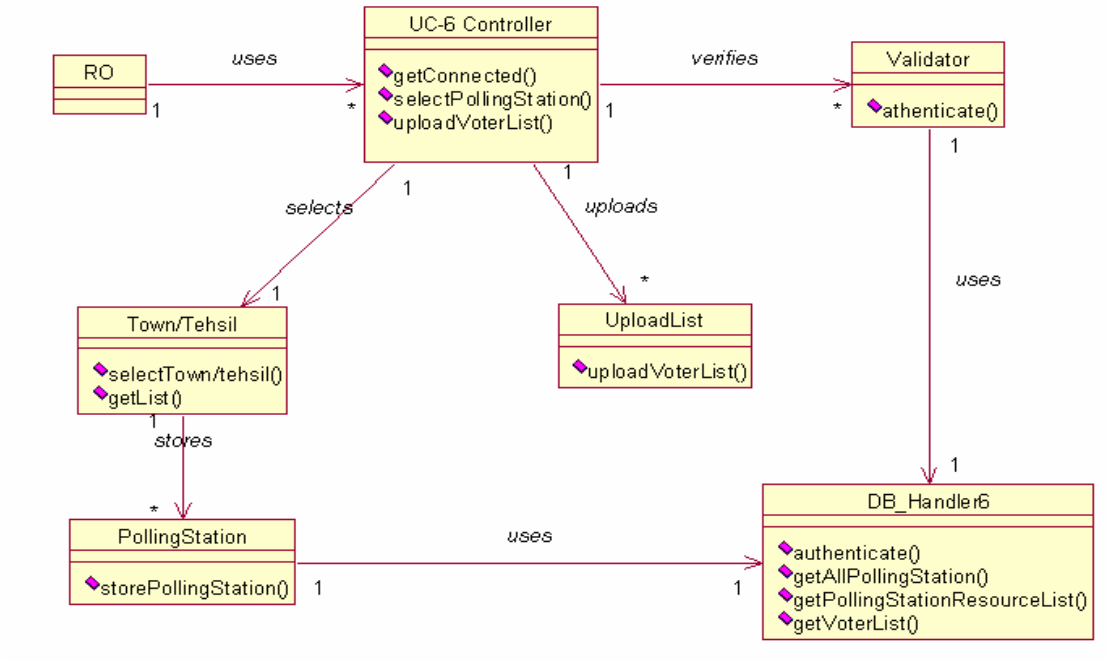


Figure 3.10.6

3.10.7 Design Class Diagram of UC_Validate_Voter (UC_7)

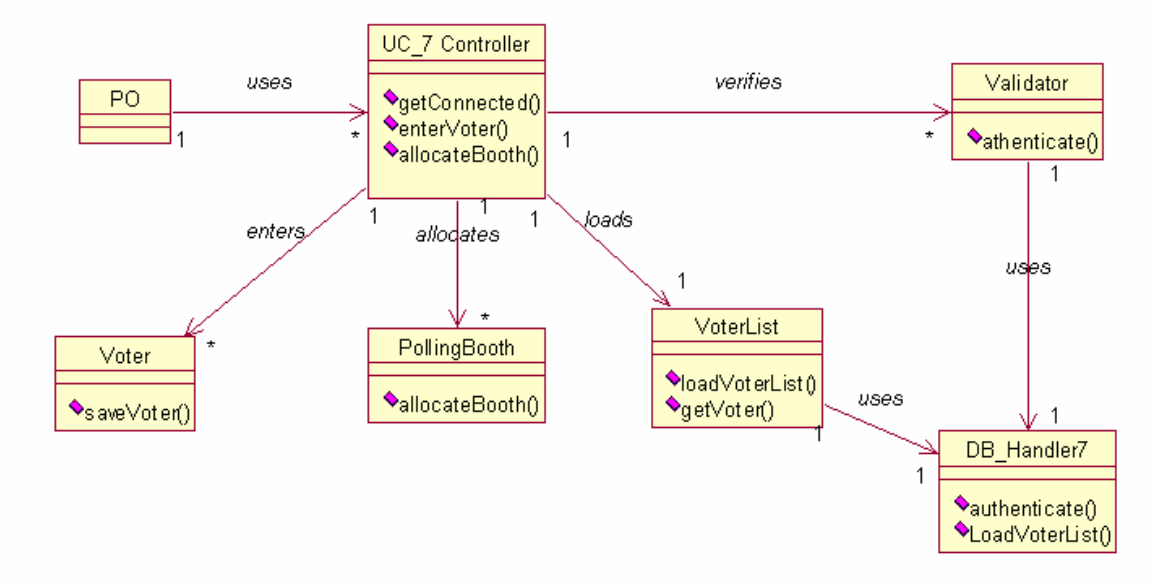


Figure 3.10.7

3.10.8 Design Class Diagram of UC_Cast_Vote (UC_8)

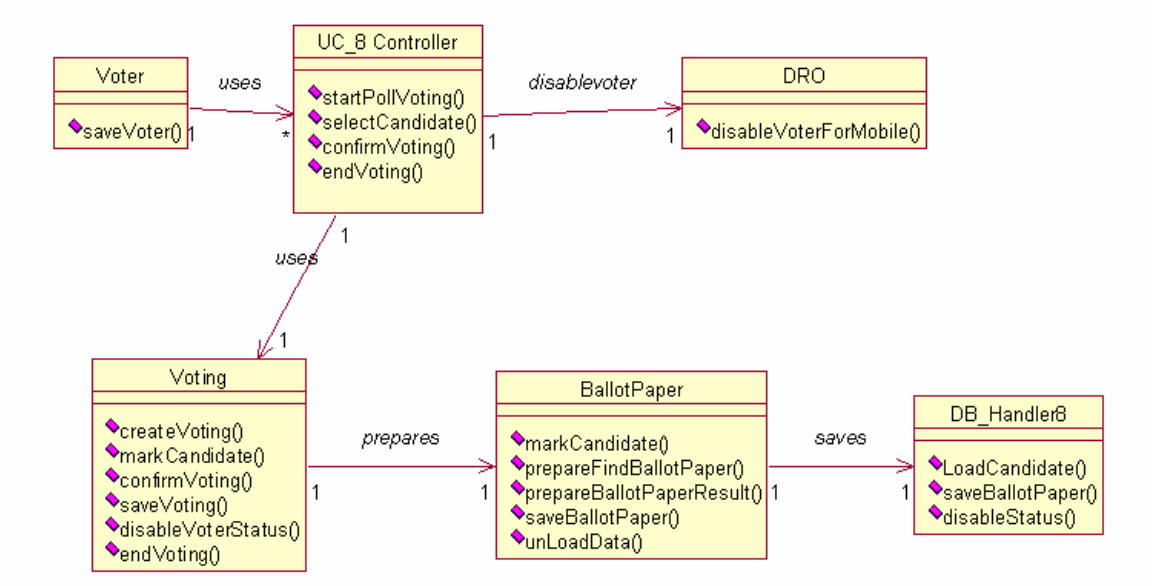


Figure 3.10.8

3.10.9 Design Class Diagram of UC_Compile_Submit_POResult (UC_9)

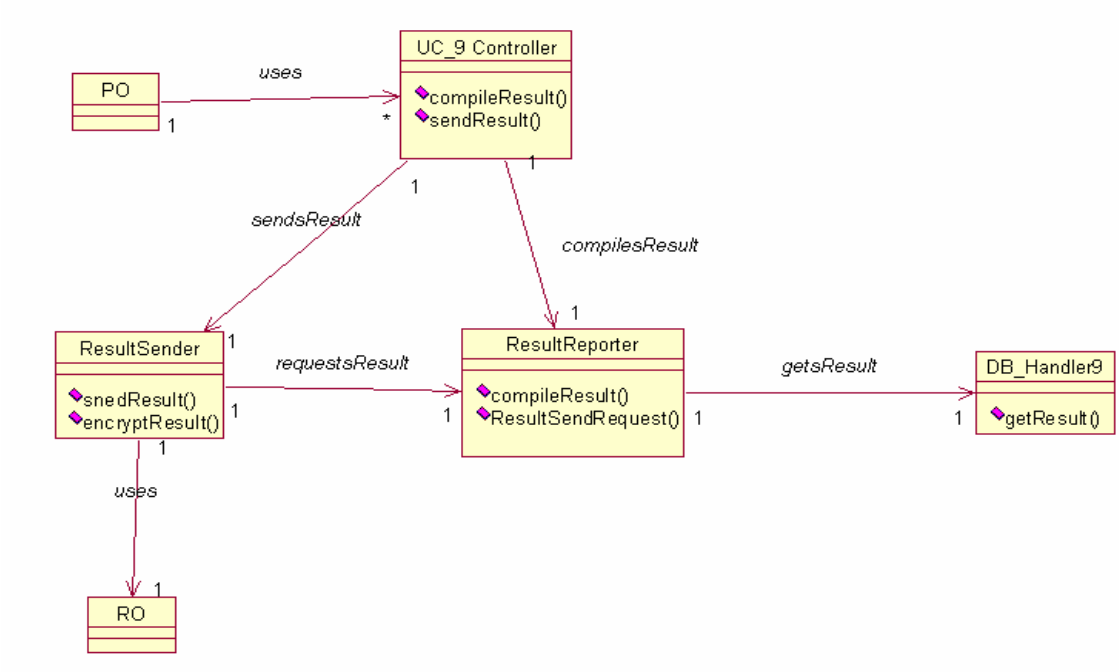


Figure 3.10.9

3.10.10 Design Class Diagram of UC_Compile_ROResult (UC_10)

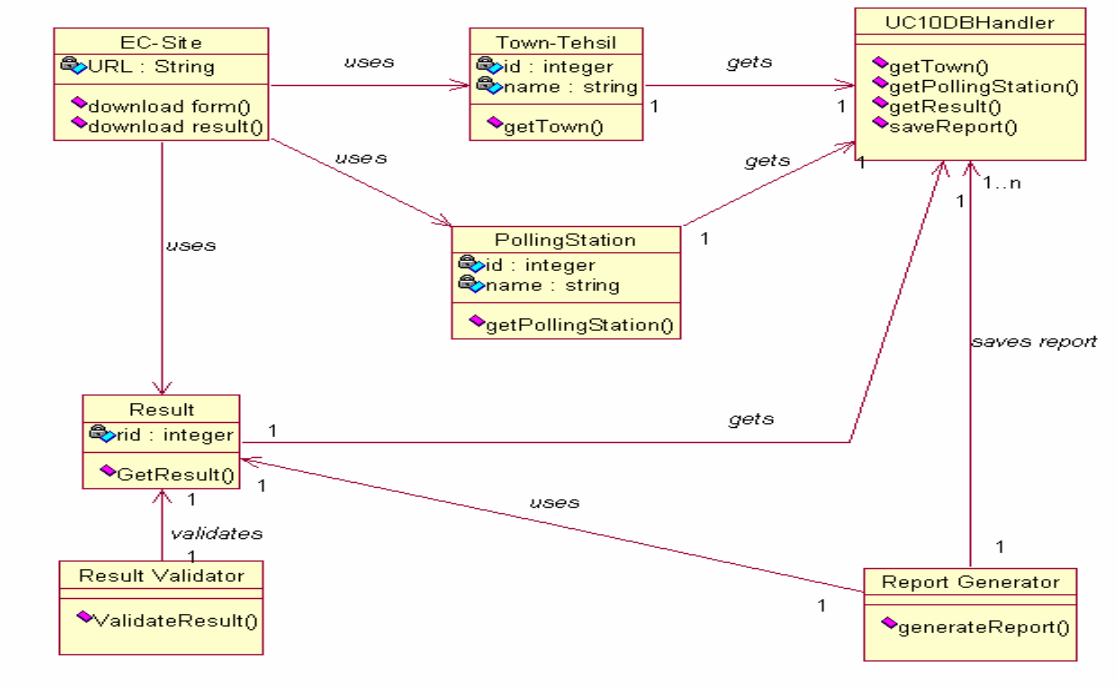


Figure 3.10.10

3.10.11 Design Class Diagram of UC_Submit_ROResult (UC_11)

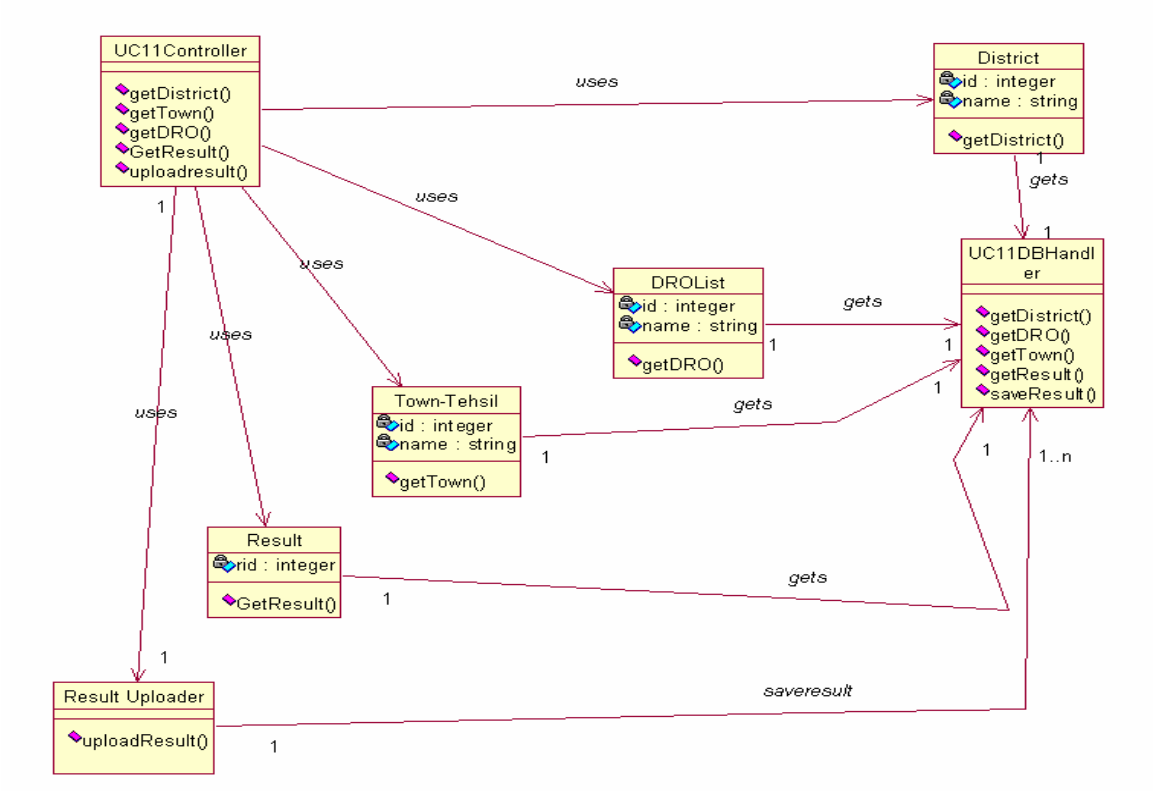


Figure 3.10.11

3.10.12 Design Class Diagram of UC_Compile_DROResult (UC_12)

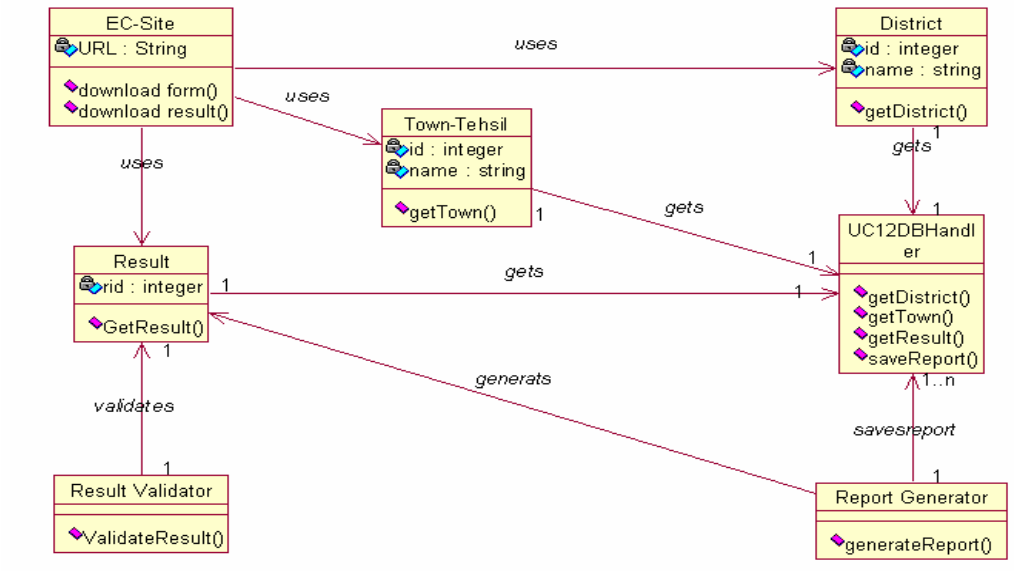


Figure 3.10.12

3.10.13 Design Class Diagram of UC_Submit_DROResult (UC_13)

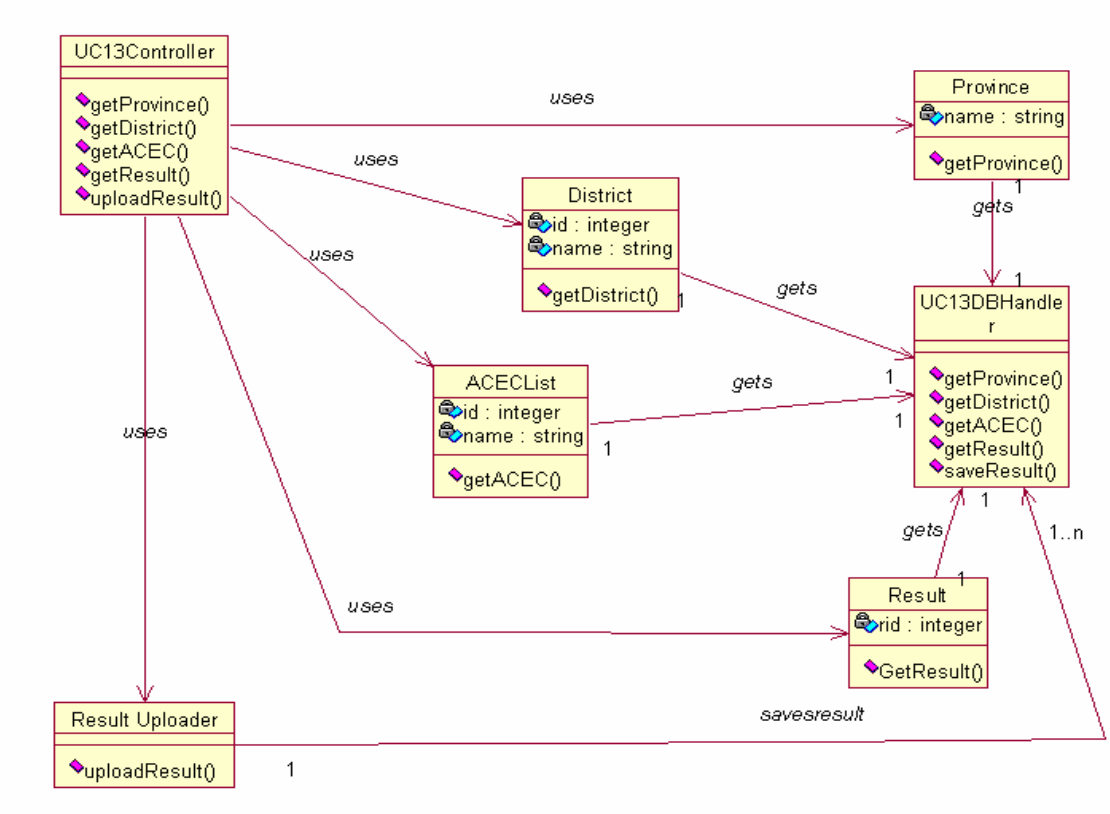


Figure 3.10.13

3.10.14 Design Class Diagram of UC_Compile_FinalResult (UC_14)

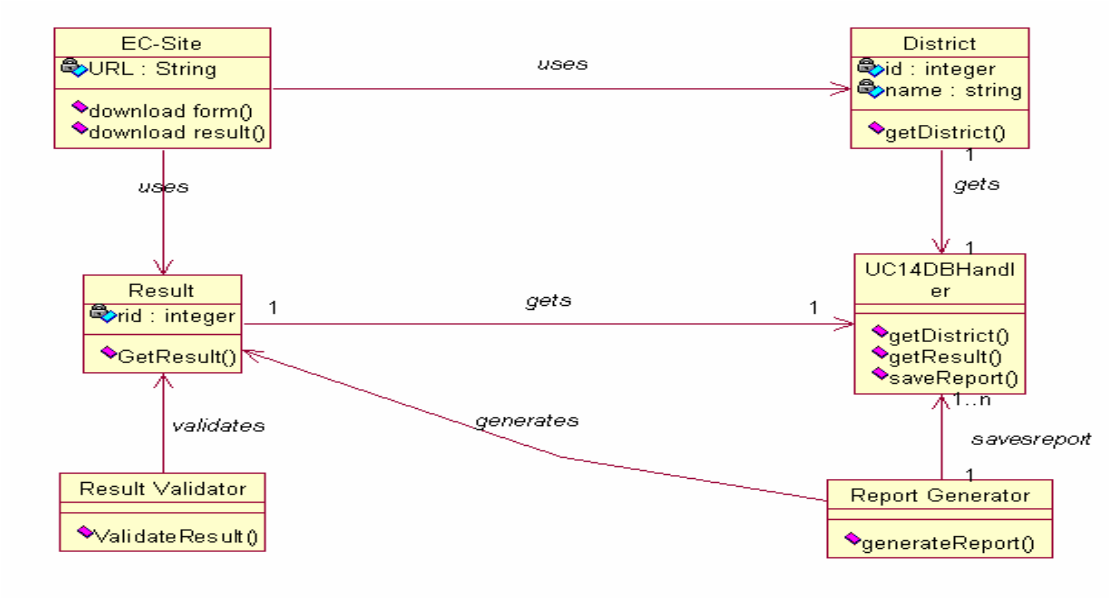


Figure 3.10.14

3.10.15 Design Class Diagram of UC_Prepare_DROList (UC_15)

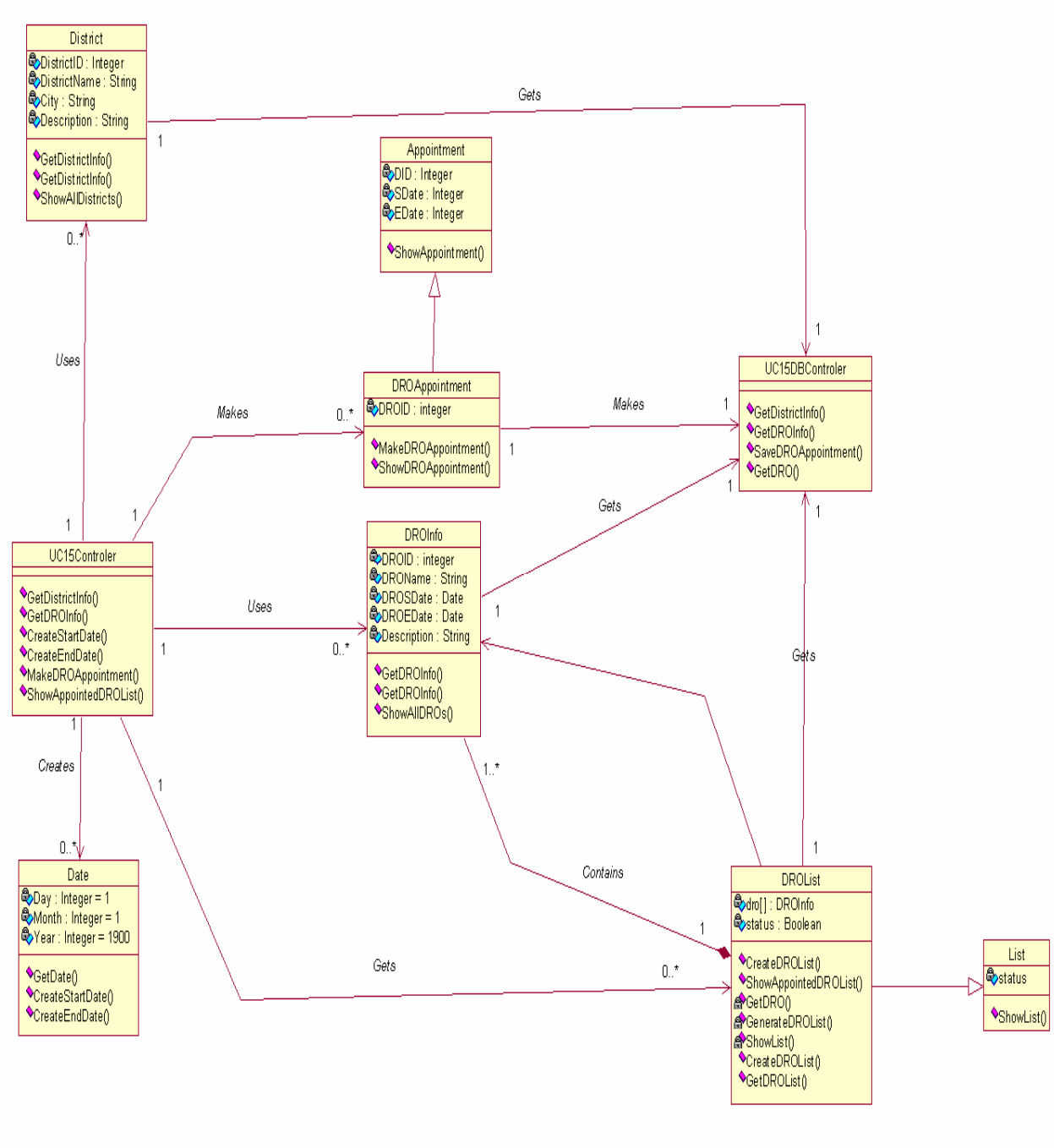


Figure 3.10.15

3.10.16 Design Class Diagram of UC_Approve_DROList (UC_16)

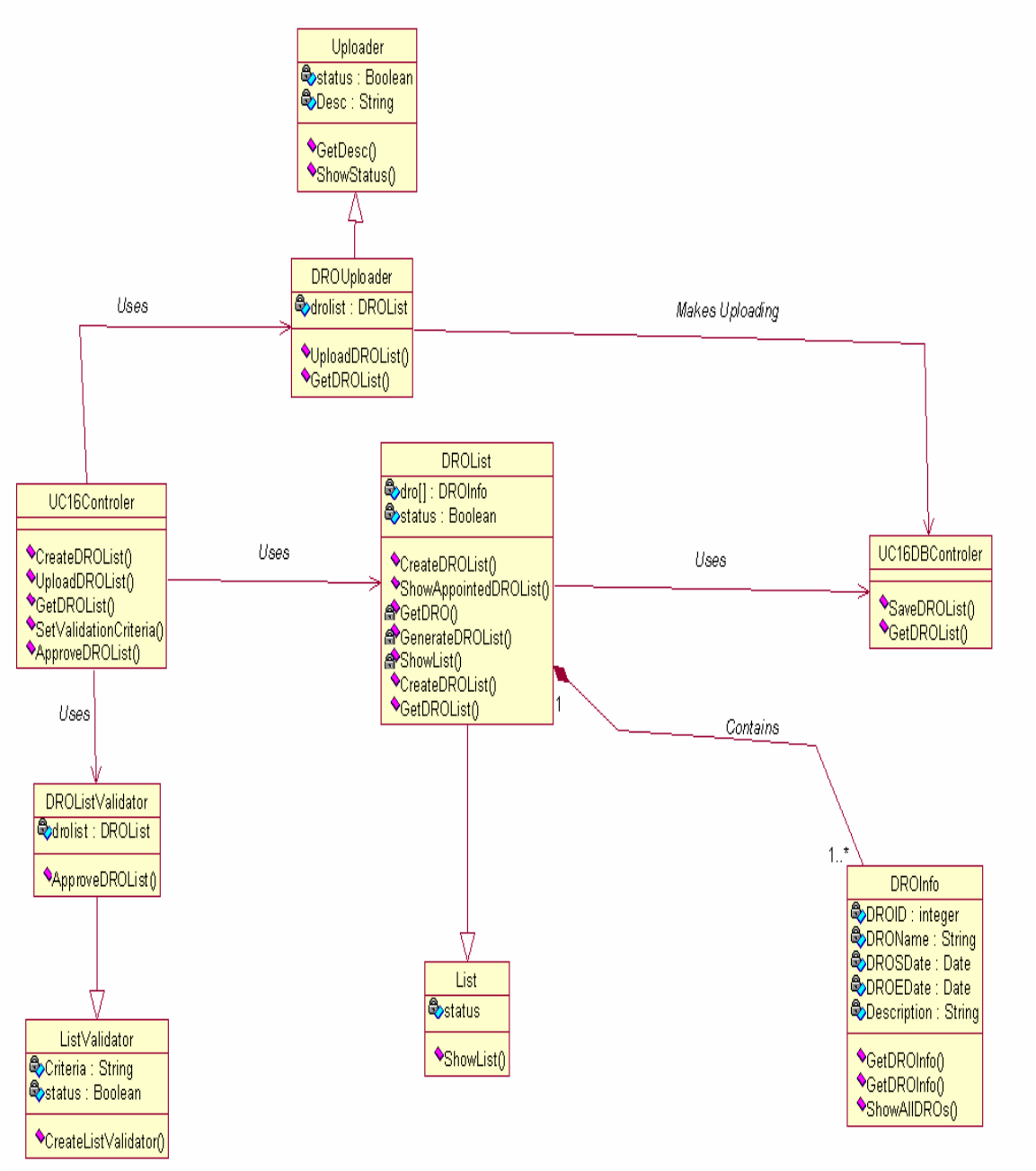


Figure 3.10.16

3.10.17 Design Class Diagram of UC_Prepare_ROList (UC_17)

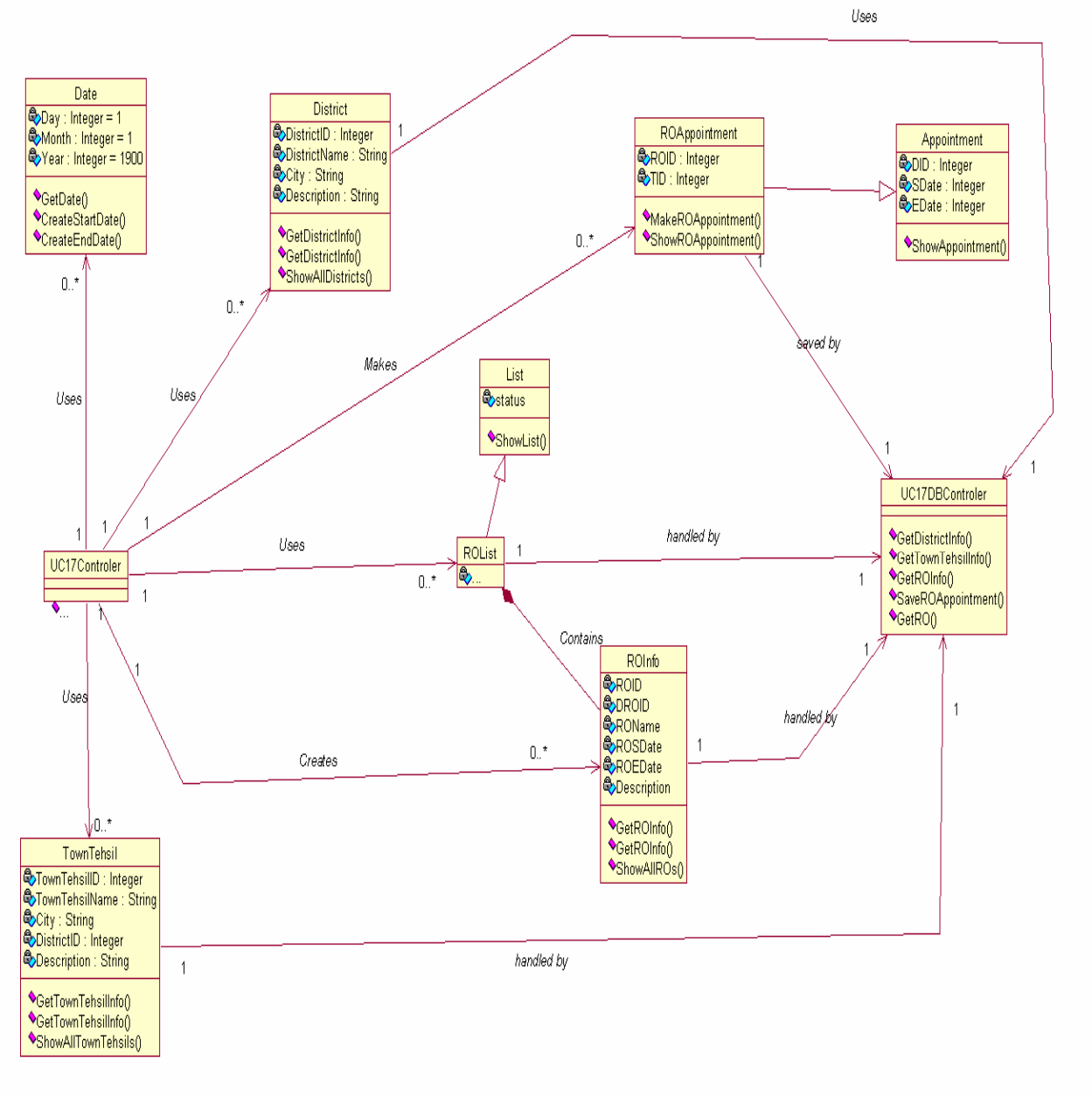


Figure 3.10.17

3.10.18 Design Class Diagram of UC_Approve_ROList (UC_18)

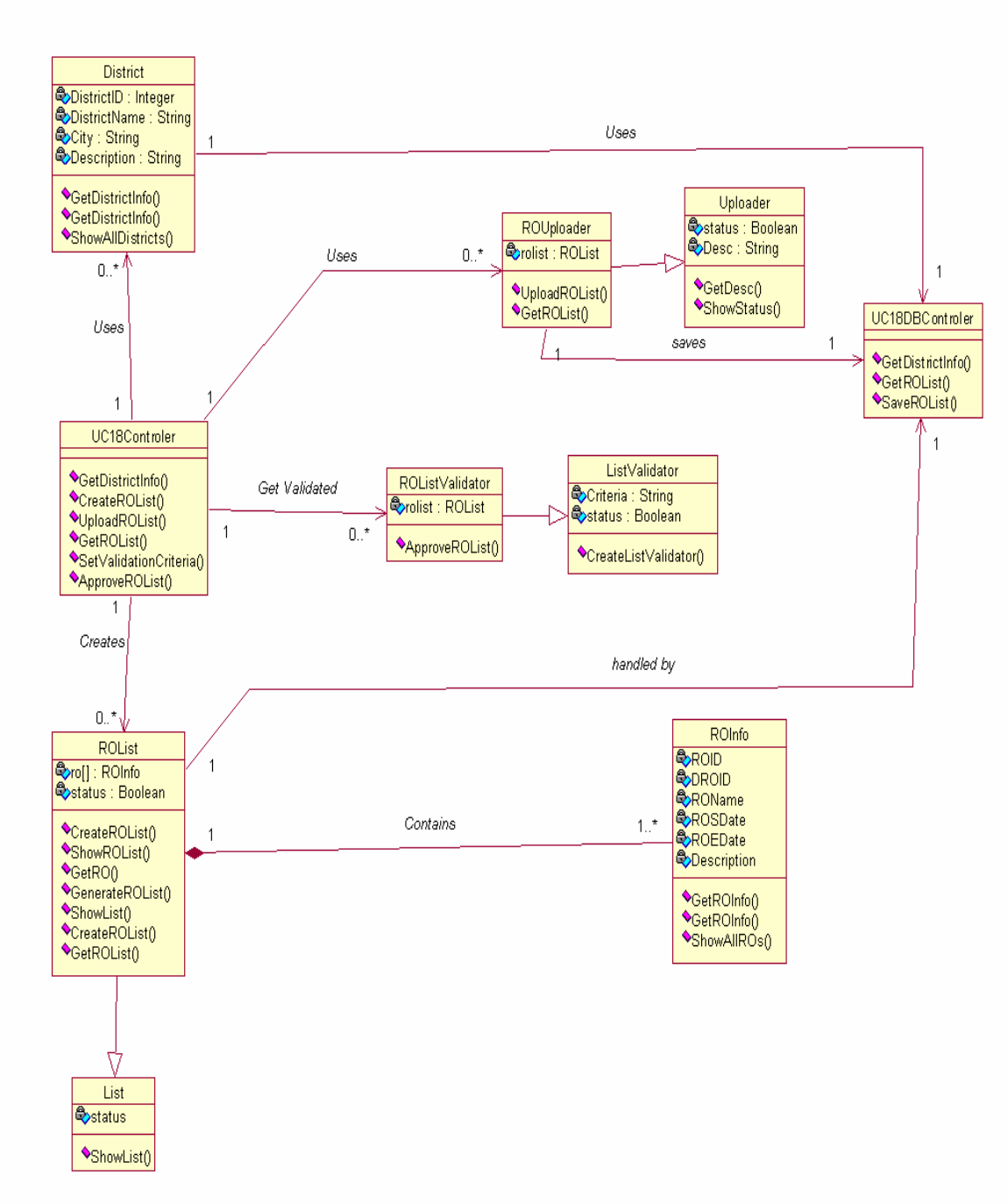


Figure 3.10.18

3.10.19 Design Class Diagram of UC_Prepare_PollingStationList (UC_19)

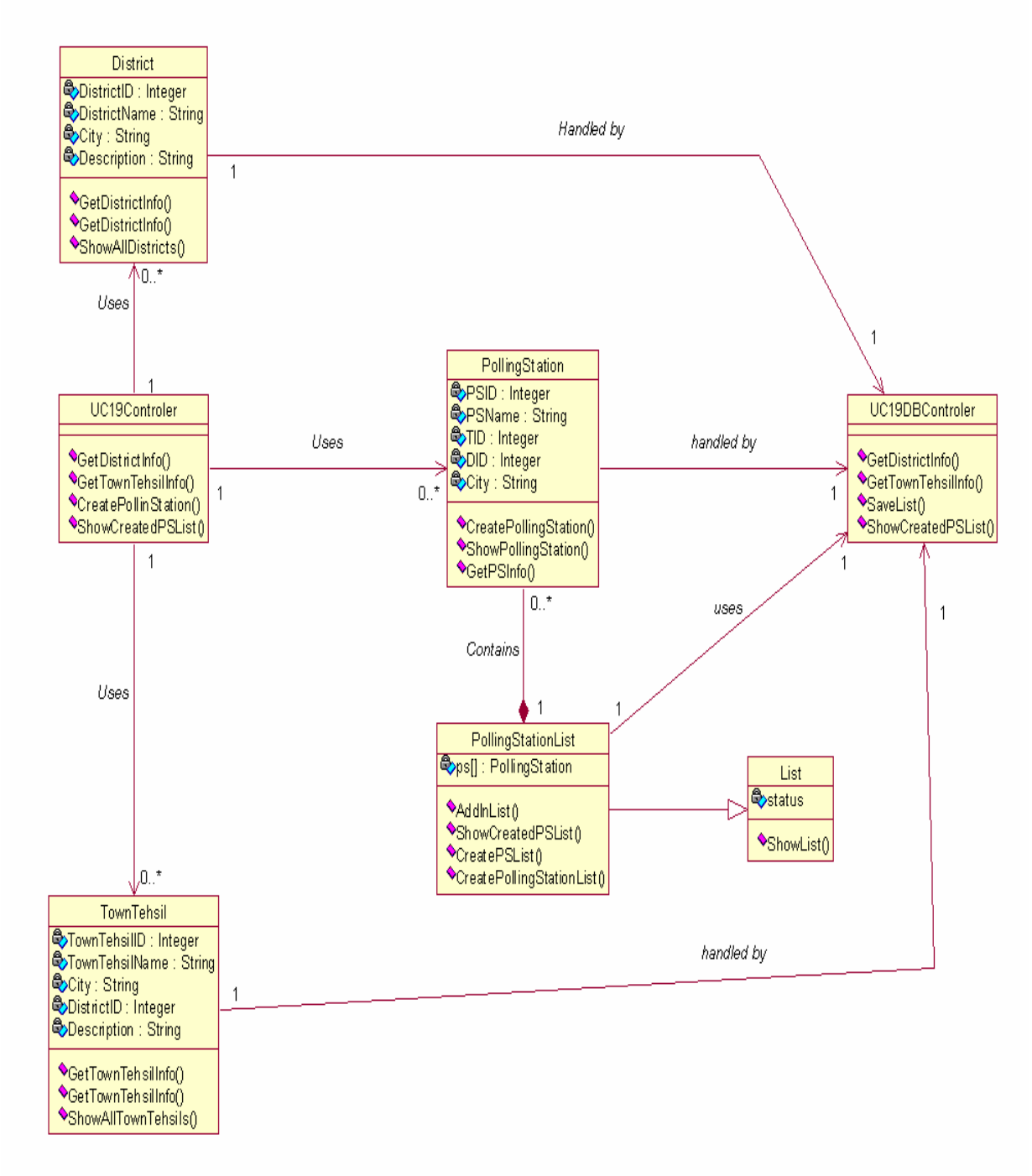


Figure 3.10.19

3.10.20 Design Class Diagram of UC_Approve_PollingStationList (UC_20)

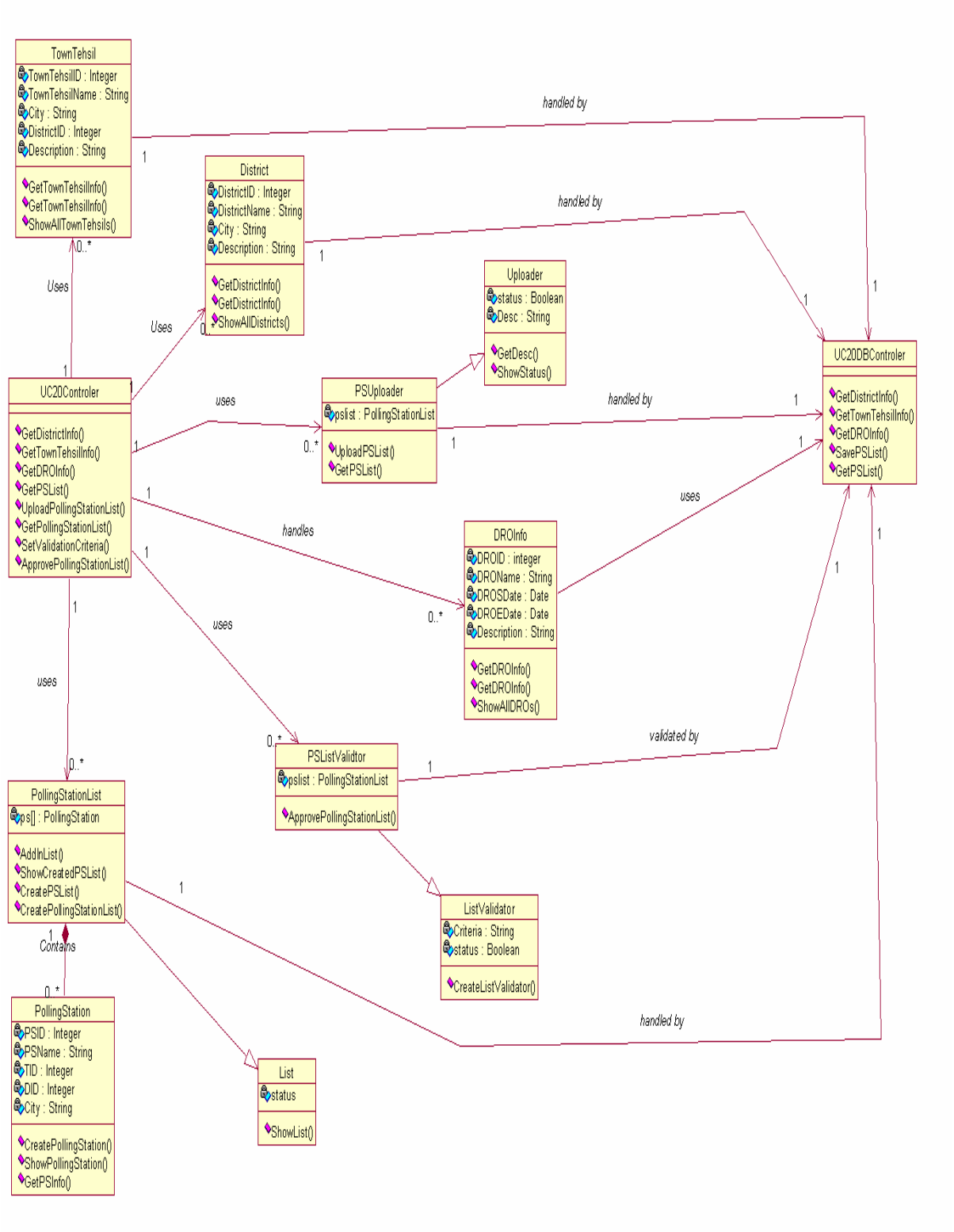


Figure 3.10.20

3.10.21 Design Class Diagram of UC_Prepare_PO_APO_PollingStaffList (UC_21)

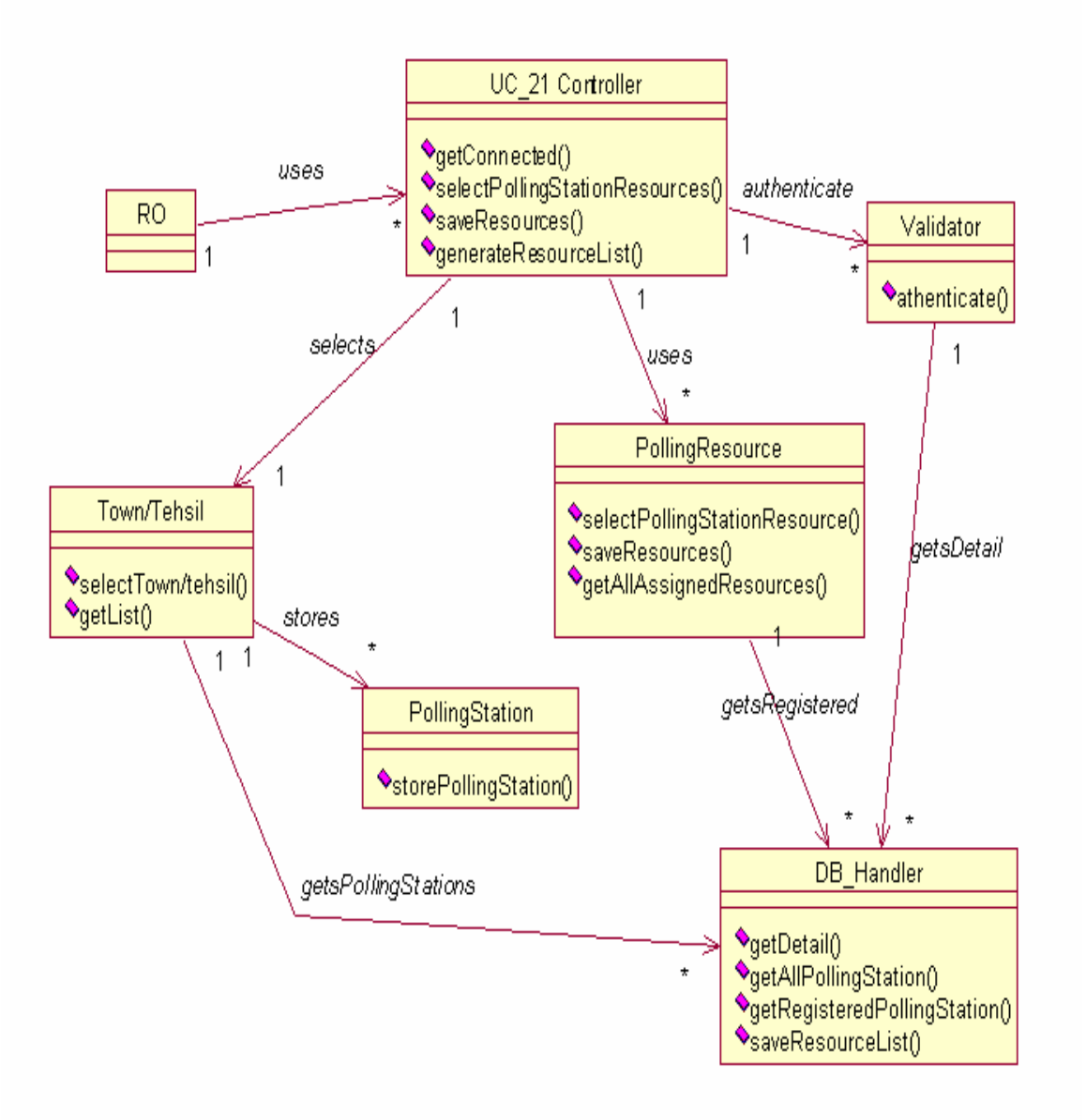


Figure 3.10.21

3.10.22 Design Class Diagram of UC_Approve_PO_APO_PollingStaffList (UC_22)

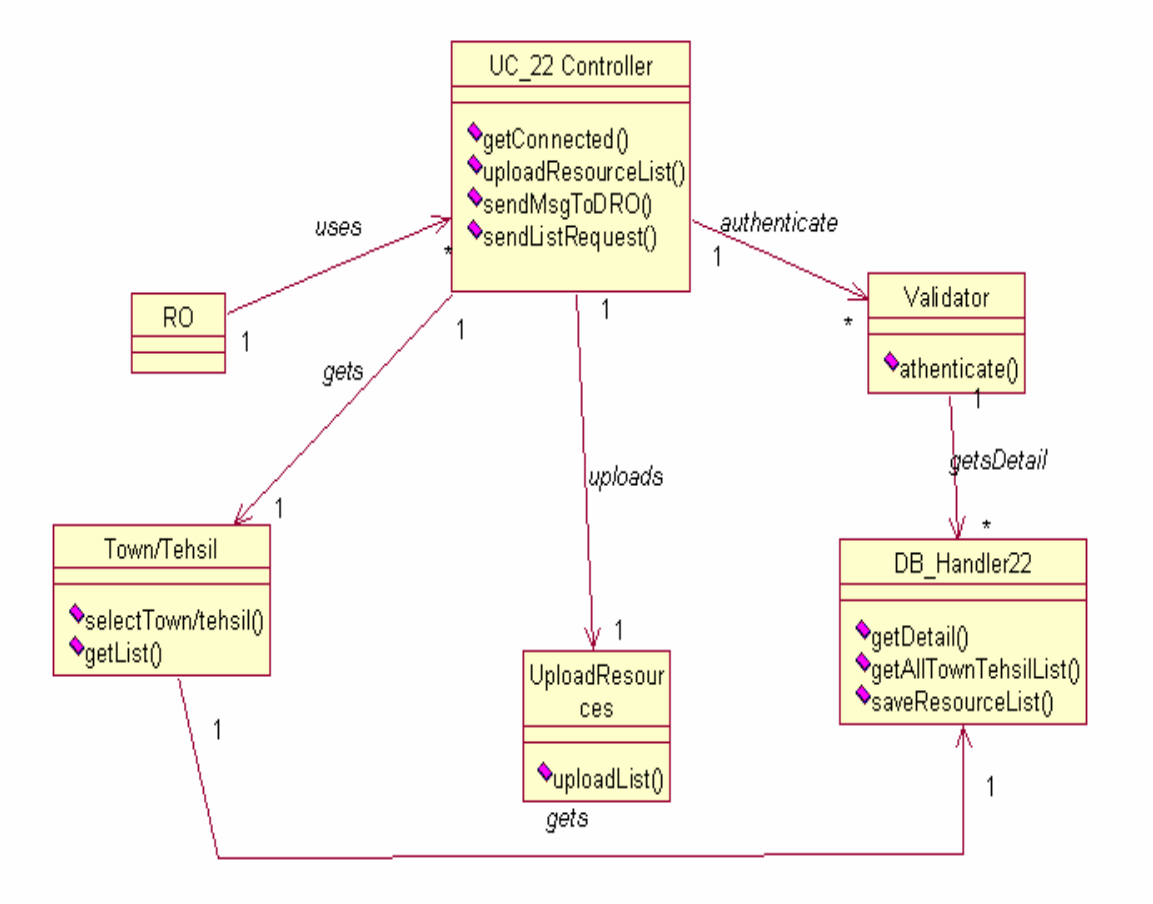


Figure 3.10.22

3.10.23 Design Class Diagram of UC_UpdateResultPOToRO (UC_23)

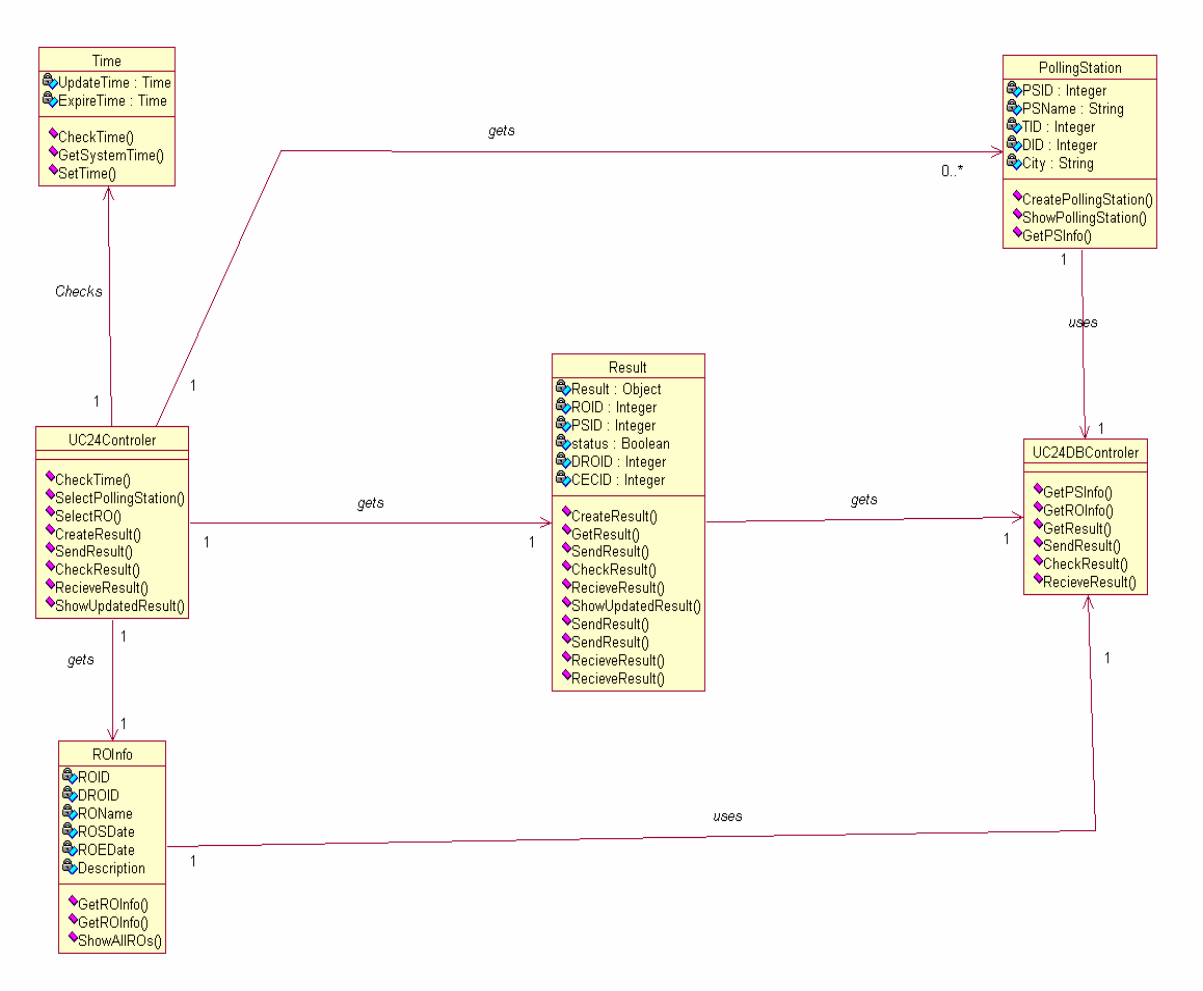


Figure 3.10.23

3.10.24 Design Class Diagram of UC_UpdateResultROToDROCEC (UC_24)

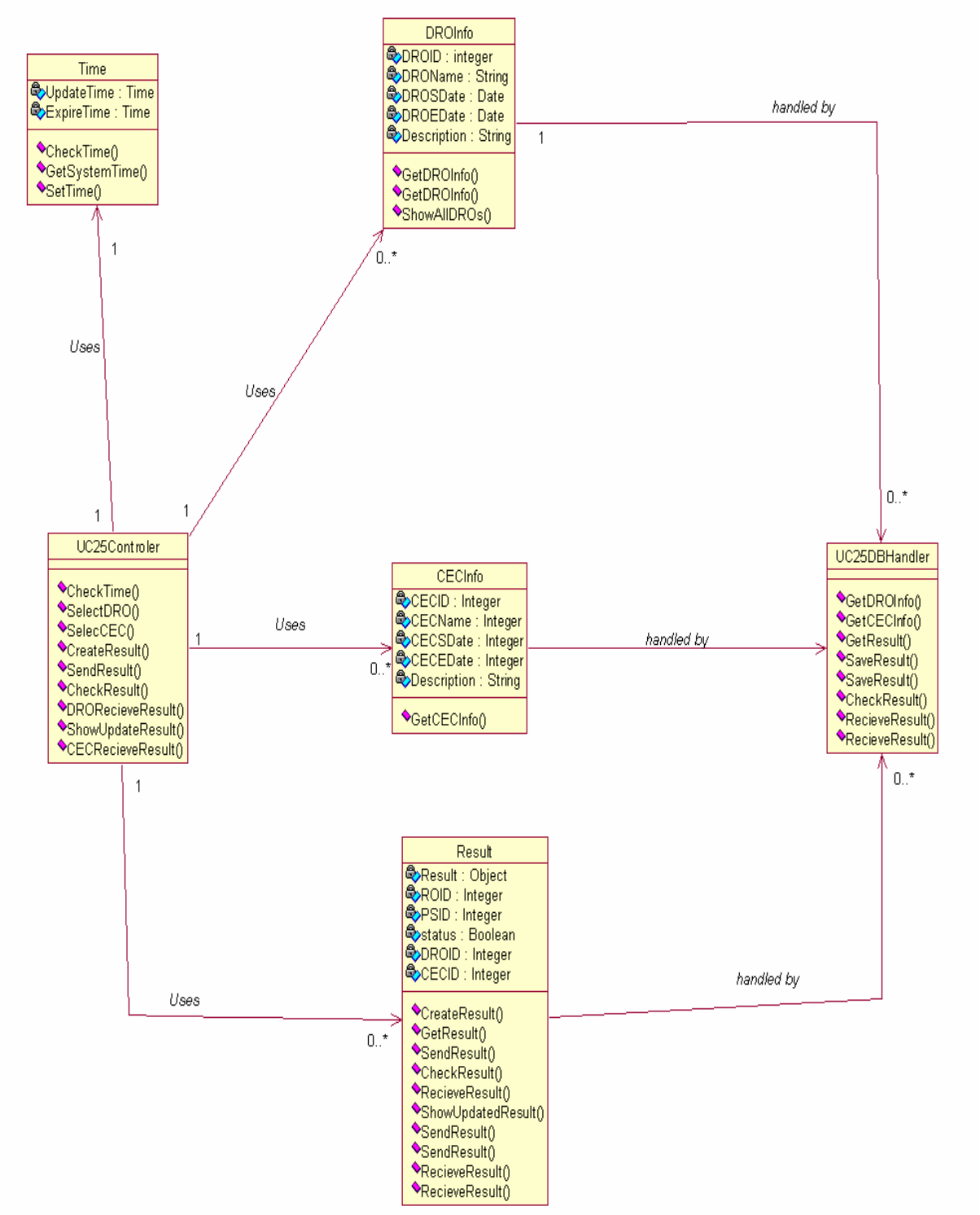


Figure 3.10.24

3.11 State Chart Diagrams

3.11.1 State Chart Diagram of UC_Prepare_ElectorlRoll (UC_1)

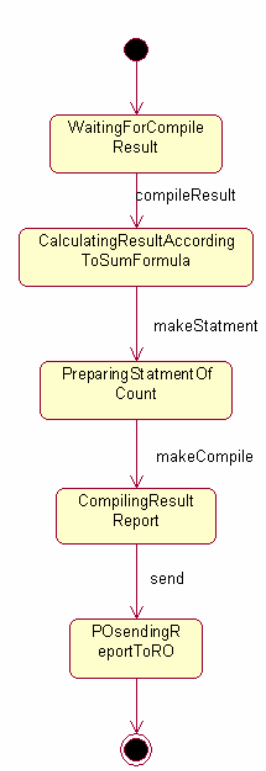


Figure 3.11.1

3.11.2 State Chart Diagram of UC_Candidate_Nomination (UC_2)

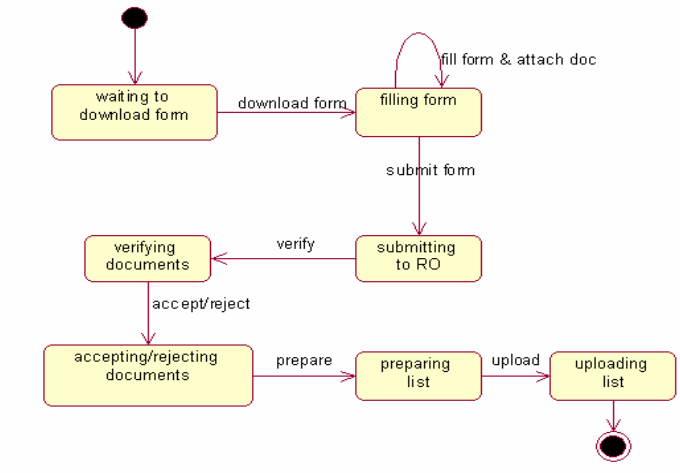


Figure 3.11.2

3.11.3 State Chart Diagram of UC_Prepare_Symbol _CandidateList (UC_3)

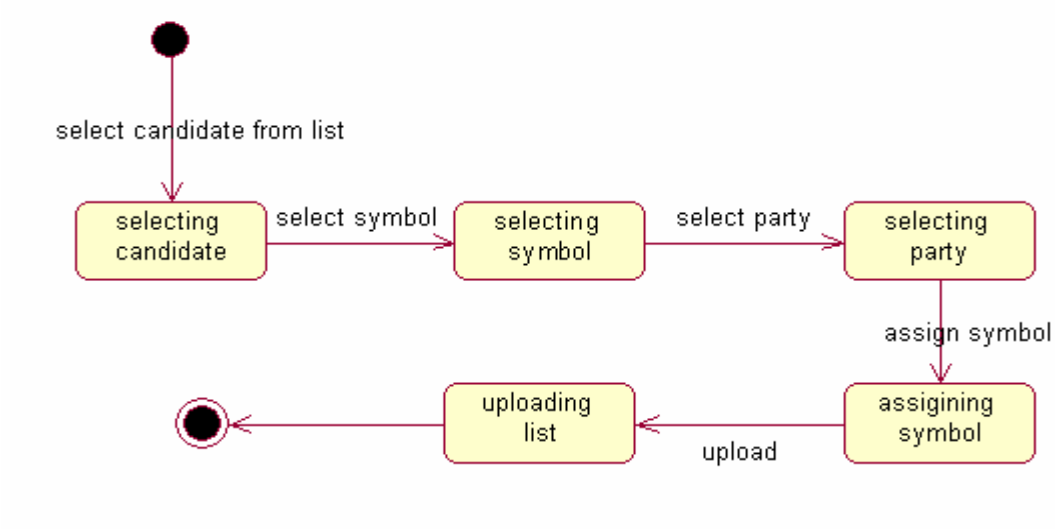


Figure 3.11.3

3.11.4 State Chart Diagram of UC_VoterList_DRODistribution (UC_4)

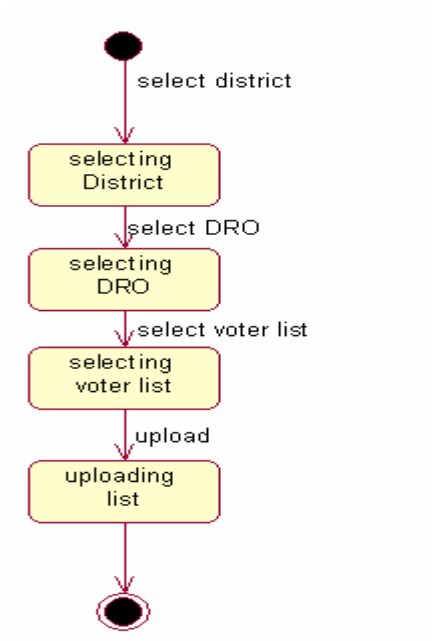


Figure 3.11.4

3.11.5 State Chart Diagram of UC_VoterList_RODistribution (UC_5)

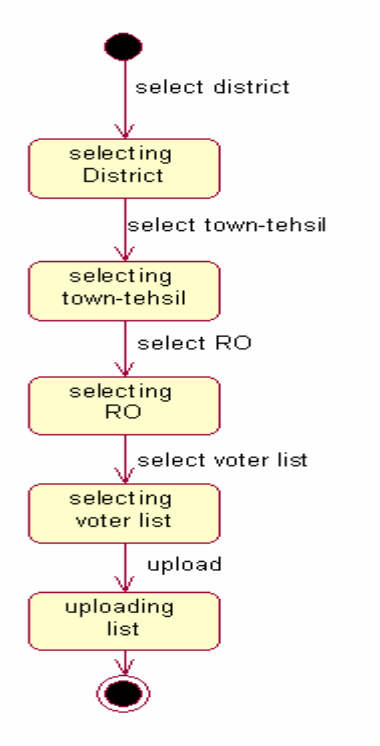


Figure 3.11.5

3.11.6 State Chart Diagram of UC_VoterList_PODistribution (UC_6)

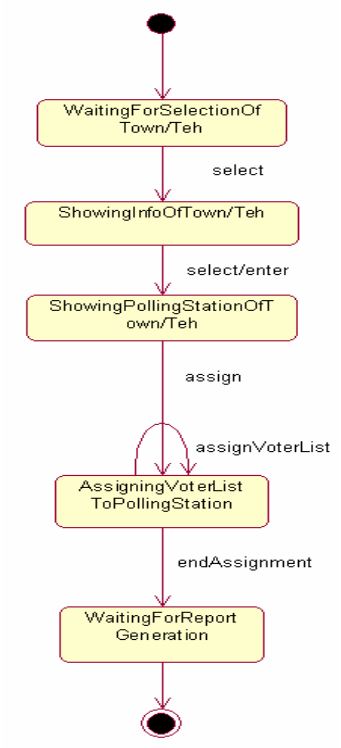


Figure 3.11.6

3.11.7 State Chart Diagram of UC_Validate_Voter (UC_7)

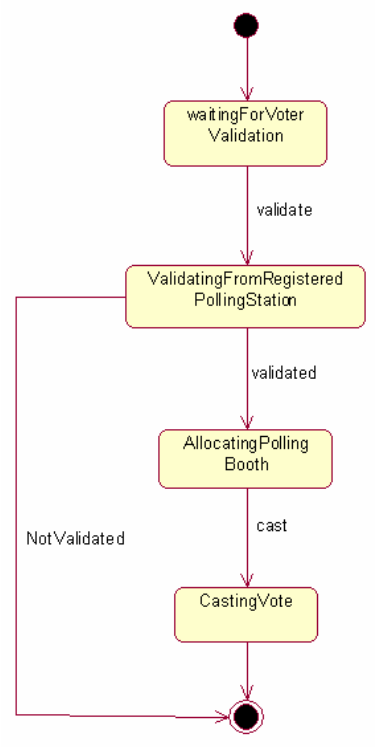


Figure 3.11.7

3.11.8 State Chart Diagram of UC_Cast_Vote (UC_8)

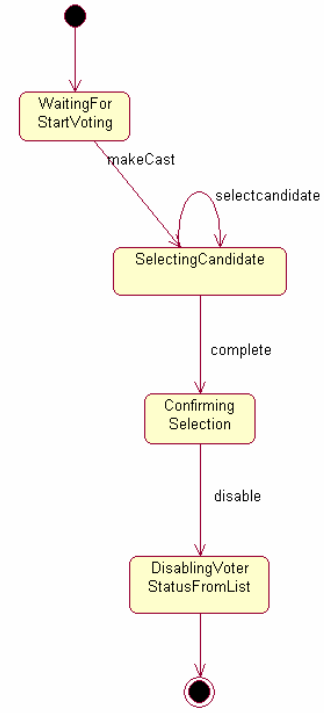


Figure 3.11.8

3.11.9 State Chart Diagram of UC_Compile_Submit_POResult (UC_9)

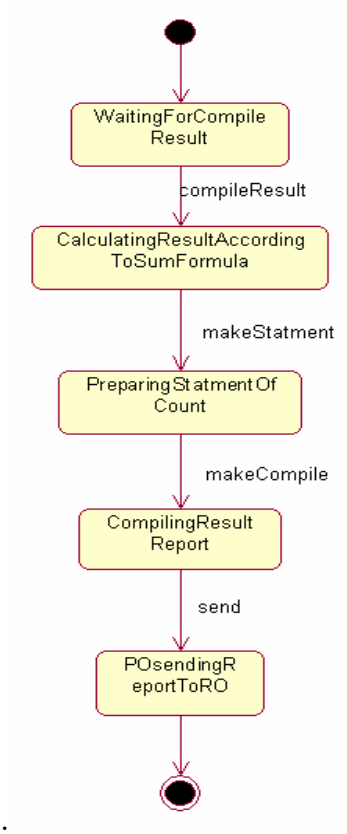


Figure 3.11.9

3.11.10 State Chart Diagram of UC_Compile_ROResult (UC_10)

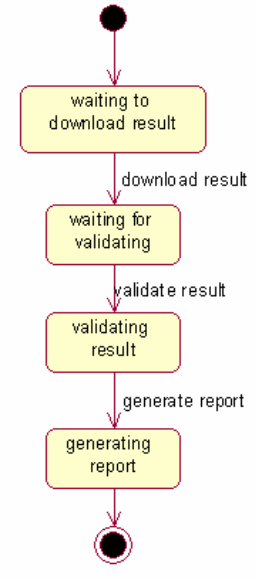


Figure 3.11.10

3.11.11 State Chart Diagram of UC_Submit_ROResult (UC_11)

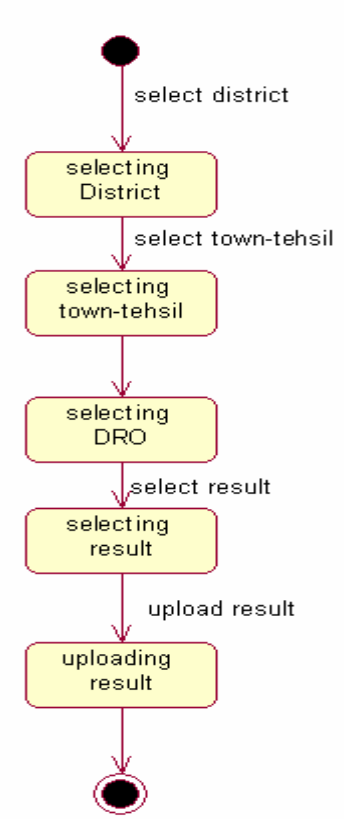


Figure 3.11.11

3.11.12 State Chart Diagram of UC_Compile_DROResult (UC_12)

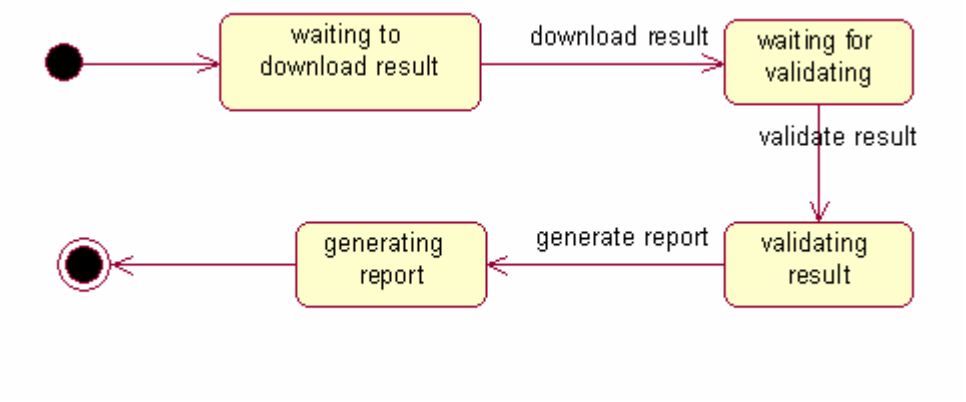


Figure 3.11.12

3.11.13 State Chart Diagram of UC_Submit_DROResult (UC_13)

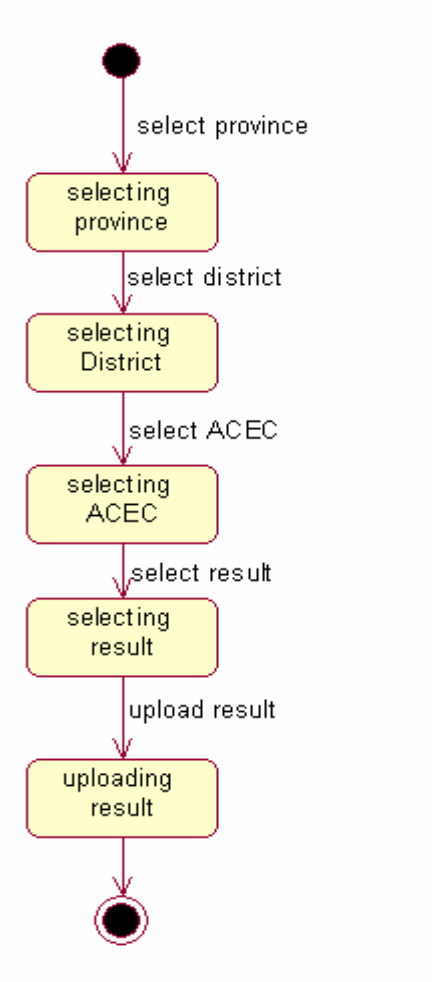


Figure 3.11.13

3.11.14 State Chart Diagram of UC_Compile_FinalResult (UC_14)

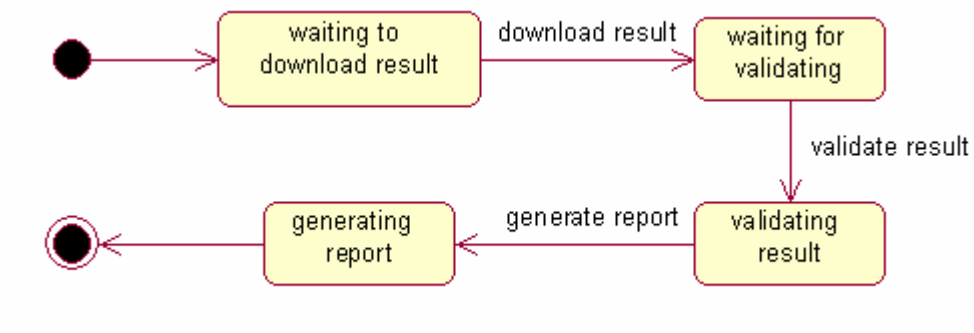


Figure 3.11.14

3.11.15 State Chart Diagram of UC_Prepare_DROList (UC_15)

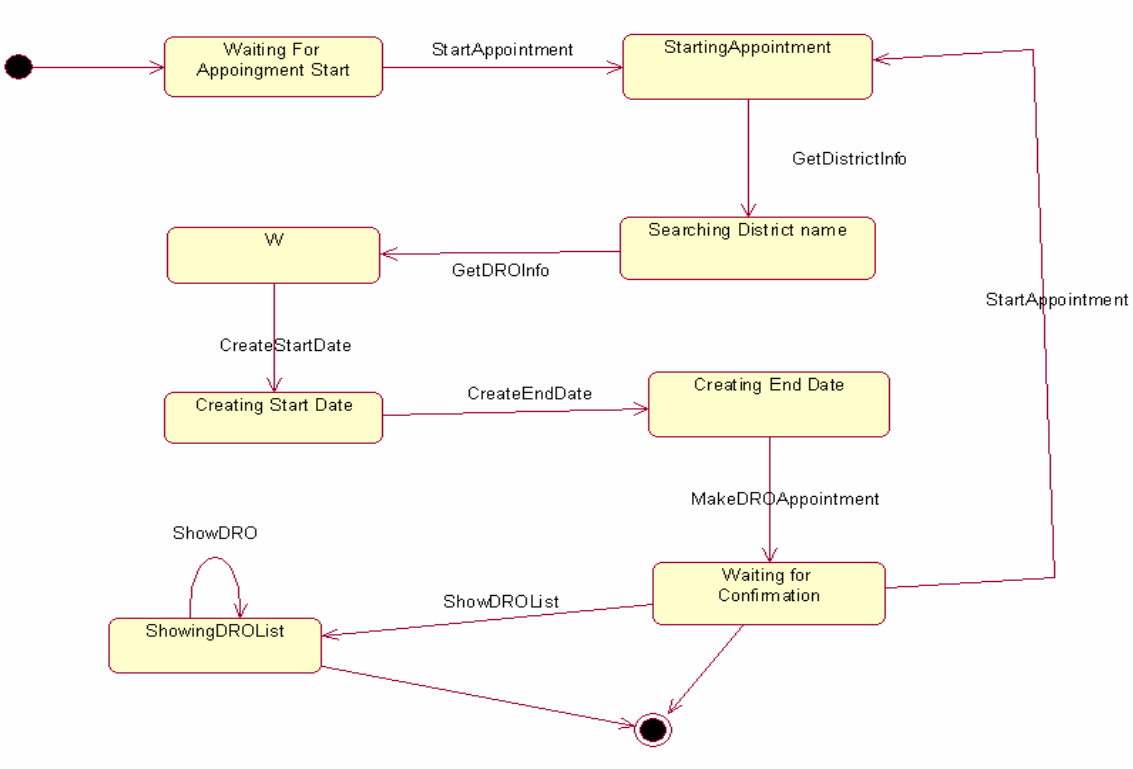


Figure 3.11.15

3.11.16 State Chart Diagram of UC_Approve_DROList (UC_16)

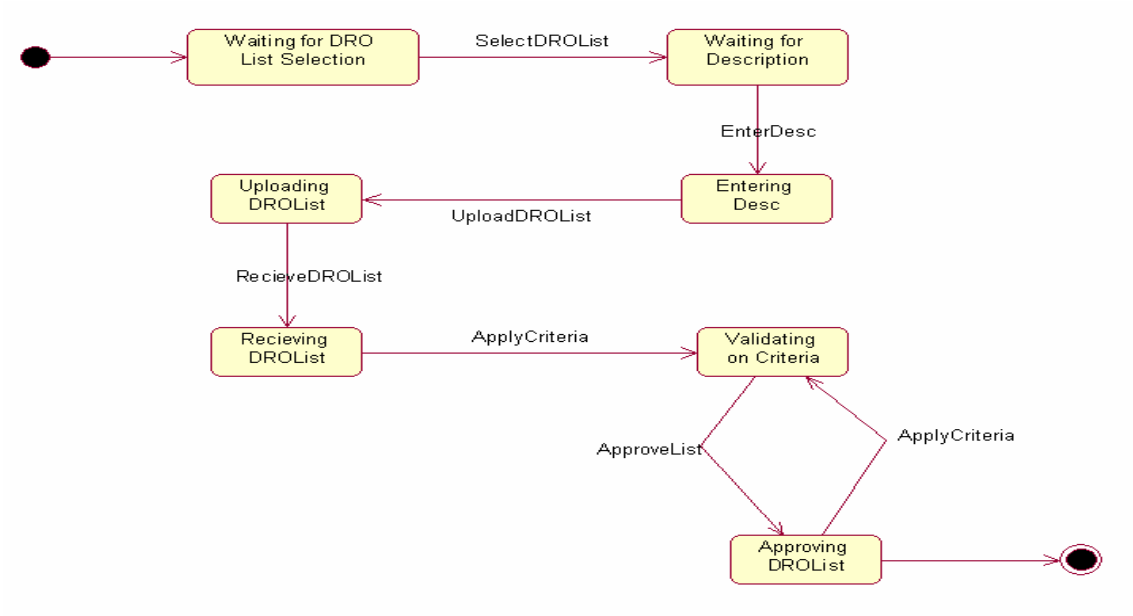


Figure 3.11.16

3.11.17 State Chart Diagram of UC_Prepare_ROList (UC_17)

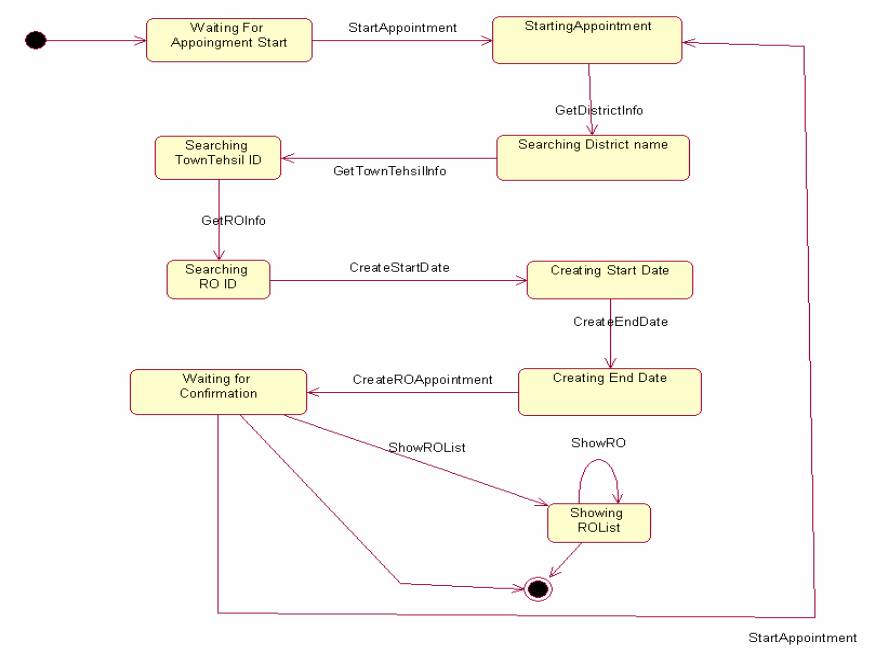


Figure 3.11.17

3.11.18 State Chart Diagram of UC_Approve_ROList (UC_18)

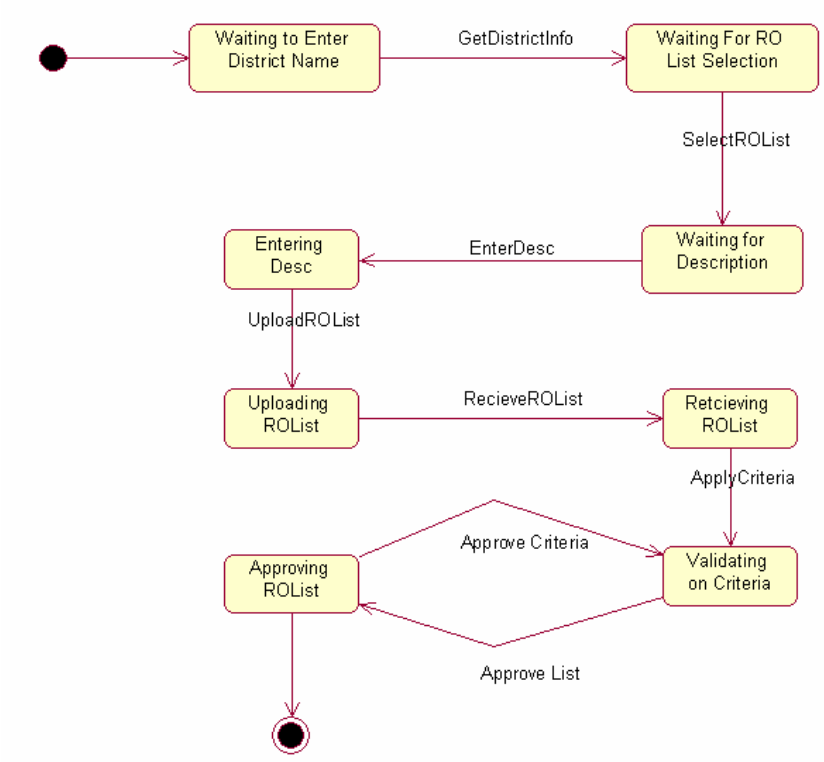


Figure 3.11.18

3.11.19 State Chart Diagram of UC_Prepare_PollingStationList (UC_19)

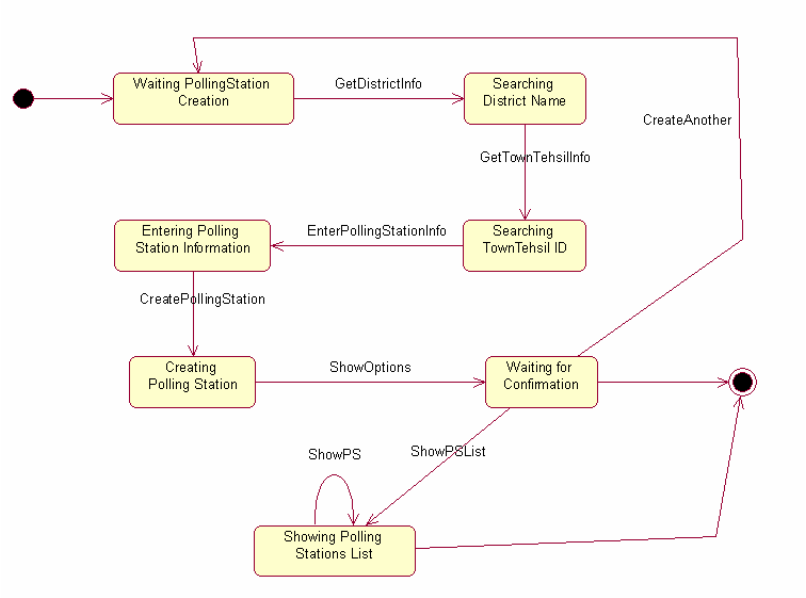


Figure 3.11.19

3.11.20 State Chart Diagram of UC_Approve_PollingStationList (UC_20)

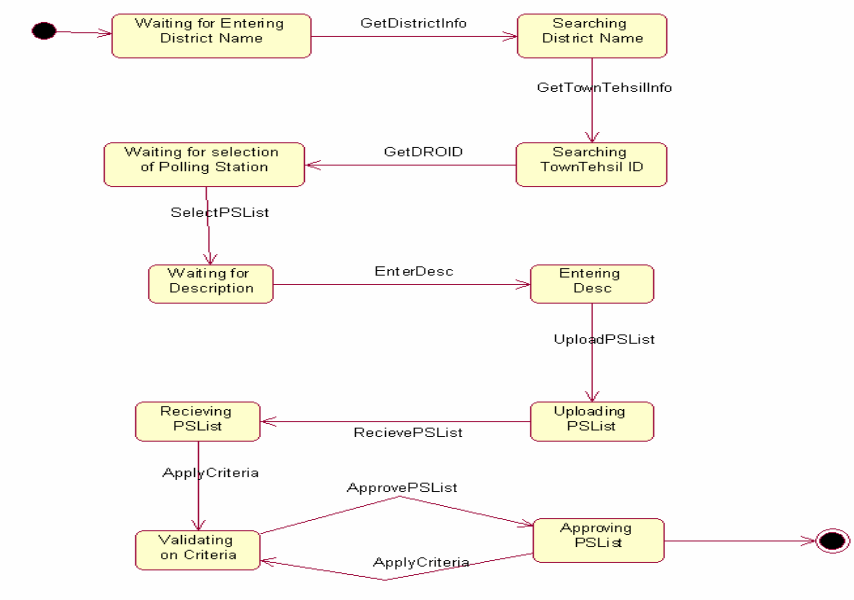


Figure 3.11.20

3.11.21 State Chart Diagram of UC_Prepare_PO_APO_PollingStaffList (UC_21)

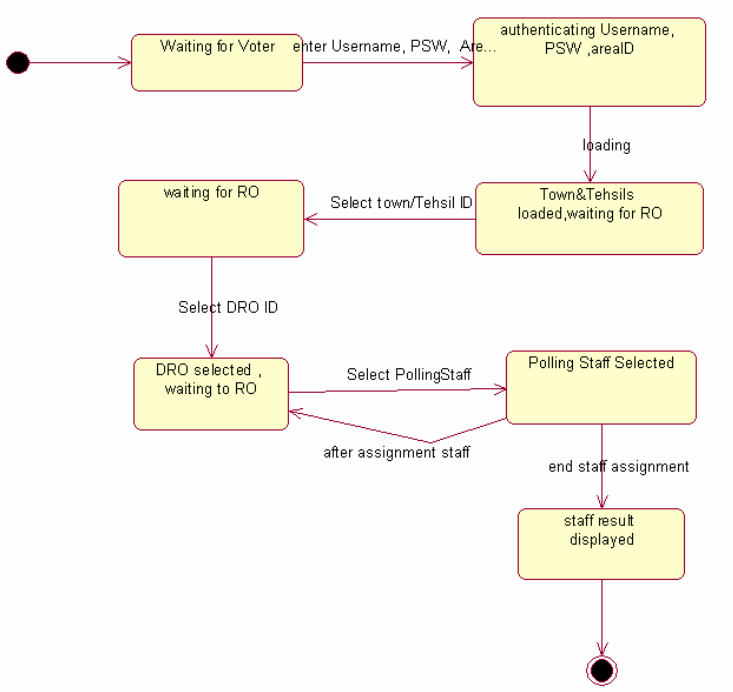


Figure 3.11.21

3.11.22 State Chart Diagram of UC_Approve_PO_APO_PollingStaffList (UC_22)

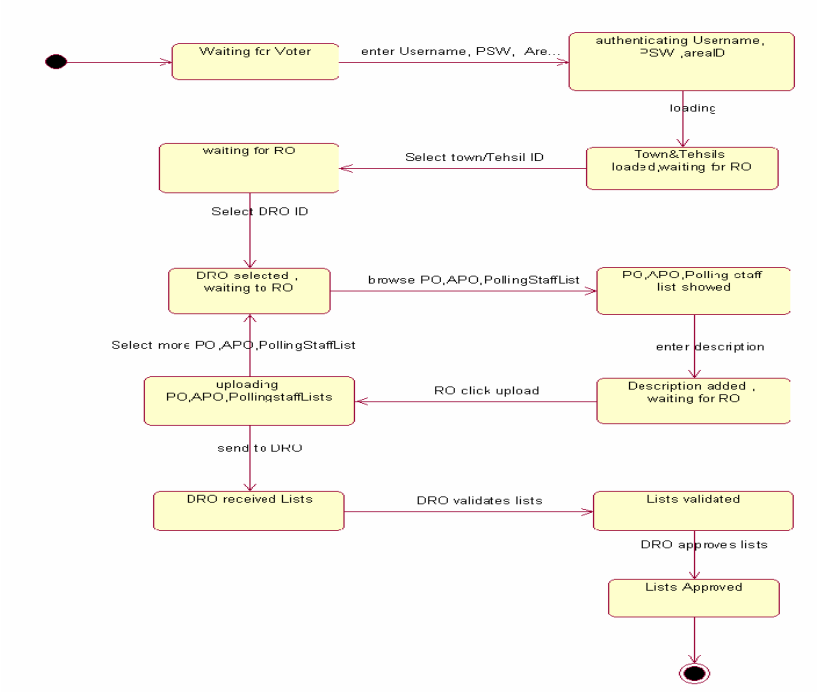


Figure 3.11.22

3.11.23 State Chart Diagram of UC_UpdateResultPOToRO (UC_23)

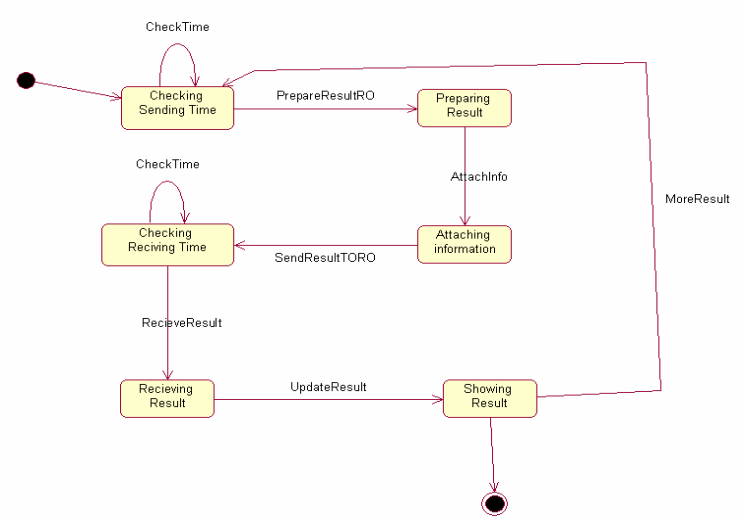


Figure 3.11.23

3.11.24 State Chart Diagram of UC_UpdateResultROToDROCEC (UC_24)

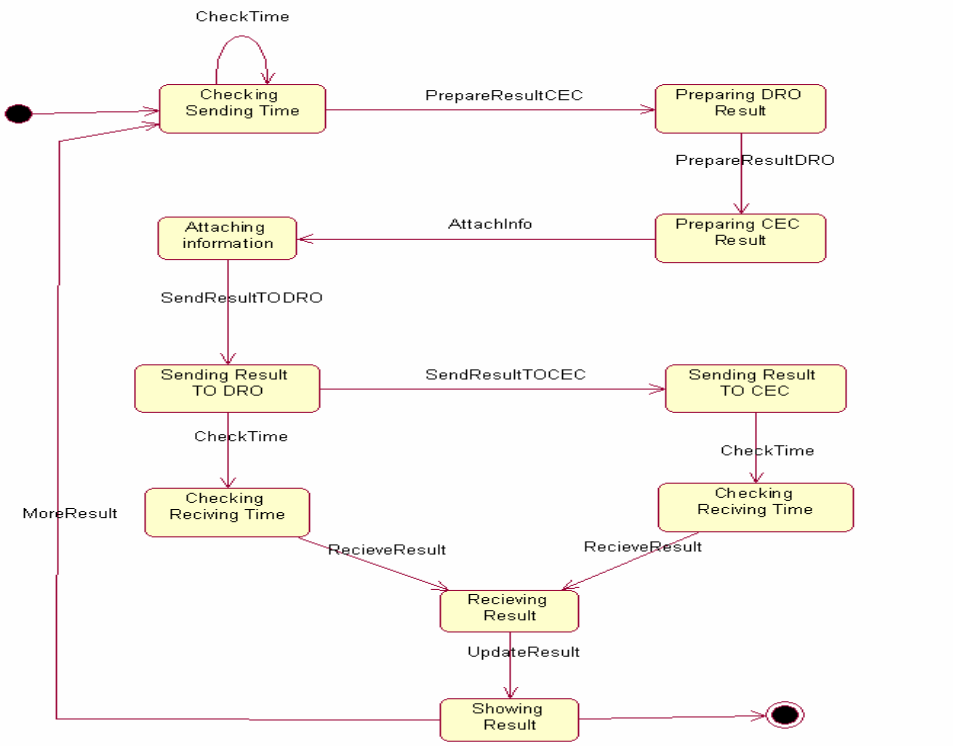


Figure 3.11.24

CHAPTER 4

IMPLEMENTATION

(Classes Specifications)

4. Class Specifications

```
public partial class ACEC : System.Web.UI.MasterPage
{
    protected void Page_Load(object sender, EventArgs e)
    {

        string Name = "";
        string type = "";
        try
        {

            Name = Session["userid"].ToString();
            type = Session["Type"].ToString();
        }
        catch (Exception ex)
        {
            Response.Redirect("../EmployeeLoginForm.aspx?id=1");
        }
        if (type != "ACEC")
        {
            Response.Redirect("../EmployeeLoginForm.aspx?id=1");
        }
        if (Name == "")
        {
            Response.Redirect("../EmployeeLoginForm.aspx?id=1");
        }

        lblACECName.Text = "LoggedIn User: " + Name;


    }

    protected void ImageButton6_Click(object sender, ImageClickEventArgs e)
    {
        try
        {
            Session.Abandon();
            Session.Remove("userid");
            Session.Remove("Type");
        }
        catch (Exception ex)
        {

        }
        Response.Redirect("../EmployeeLoginForm.aspx?id=2");
    }
}
```

```
public partial class ACECMainPage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
}
```

```
public partial class ACECResult : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        //Response.Cookies["WWF"].Values.Add("ACEC_Id", "1");
    }
    protected void Timer1_Tick(object sender, EventArgs e)
    {
        GridView1.DataBind();
    }
}
```

```
public partial class DROListCECDistribution : System.Web.UI.Page
{
    Database db1, db2;
    int droid;
    protected void Page_Load(object sender, EventArgs e)
    {
        db1 = DatabaseFactory.CreateDatabase("LocalSqlConnection");
        Connect();
        if (!IsPostBack)
        {
            FillCombo(cmbProvince, "SelectProvince");
        }

    }
    protected void FillCombo(DropDownList combo, string spName)
    {
        try
        {
            DbCommand cmd = db1.GetSqlStringCommand(spName);
            cmd.CommandType = CommandType.StoredProcedure;

            IDataReader reader = db1.ExecuteReader(cmd);
            int i = 1;
            combo.Items.Clear();
            combo.Items.Add("");
            while (reader.Read())
            {
                combo.Items.Add(reader.GetValue(1).ToString());
                combo.Items[i].Value = reader.GetValue(0).ToString();
                i++;
            }
            combo.SelectedIndex = 0;
        }
        catch (Exception ex)
        {
            Response.Write(ex.Message);
        }
    }//fill Simple Combo
    protected void cmbProvince_SelectedIndexChanged(object sender, EventArgs e)
    {
        string pid = cmbProvince.SelectedValue;
        if (pid != "")
        {
            if (int.Parse(pid) > 0)
                SetCECByProvince(int.Parse(pid));
```

```
    }
    else
    {
        lblDRO.Text = "";
        btnSend.Enabled = false;
    }

}
protected void SetCECByProvince(int did)
{
    try
    {
        lblDRO.Text = "";
        DbCommand cmd = db1.GetSqlStringCommand("SelectCEC");

        cmd.CommandType = CommandType.StoredProcedure;
        db1.AddInParameter(cmd, "Province_Id", DbType.Int64, did);
        IDataReader reader = db1.ExecuteReader(cmd);
        while (reader.Read())
        {
            lblDRO.Text = reader.GetValue(1).ToString();
            ViewState.Add("droid", reader.GetValue(0).ToString());
            btnSend.Enabled = true;
        }
    }
    catch (Exception ex)
    {
        //Response.Write(ex.Message);
        Response.Redirect("~/Problem.aspx");
    }
}

protected Boolean Connect()
{
    try
    {
        db2 = DatabaseFactory.CreateDatabase("ServerSqlConnection");
        return true;

    }
    catch (Exception ex)
    {
        return false;
    }
}

protected void btnSend_Click(object sender, EventArgs e)
{
    string did = cmbProvince.SelectedValue;
    if (did != "")
    {
        if (int.Parse(did) > 0)
        {
            droid = int.Parse(ViewState["droid"].ToString());

            if (UpdateVoterListOnServer(int.Parse(did), droid) == true)
                lblMSG.Text = "Records Send!";
            else
                lblMSG.Text = "Nothing to Send.";
        }//int.parse
```

```
        }
        else
        {
            lblMSG.Text = "Wrong Province Selection!";
        }


    }
    private Boolean UpdateVoterListOnServer(int DistID, int DROID)
    {

        try
        {
            DbCommand cmd = db2.GetSqlStringCommand("SendToCECDROList");
            cmd.CommandType = CommandType.StoredProcedure;
            db2.AddInParameter(cmd, "Province_Id", DbType.Int32, DistID);
            db2.AddInParameter(cmd, "CEC_Id", DbType.Int32, DROID);

            int count = db2.ExecuteNonQuery(cmd);
            if (count > 0)
            {
                return true;
            }
            return false;
        }
        catch (Exception ex)
        {
            return false;
        }


    }
}
public partial class EditVoterList : System.Web.UI.Page
{
    Database db1;
    string id = "";
    protected void Page_Load(object sender, EventArgs e)
    {
        db1 = DatabaseFactory.CreateDatabase("LocalSqlConnection");

        try
        {
            id = Request.QueryString["id"];
        }
        catch (Exception ex) {
        }

        if (id == null)
        {
        }
        else {
            if (id.Trim() != "") {
                ShowVoterInfo(int.Parse(id));
                LinkButton1.PostBackUrl = "~/ACEC/UpdateVoterList.aspx?id=" + id;
            }

        }
```

```
    }
    protected void ShowVoterInfo(int id)
    {
        try
        {
            DbCommand cmd = db1.GetSqlStringCommand("selectVoter");
            cmd.CommandType = CommandType.StoredProcedure;
            db1.AddInParameter(cmd, "Voter_Id", DbType.Int64, id);
            IDataReader reader = db1.ExecuteReader(cmd);

            while (reader.Read())
            {
                Label1.Text = reader.GetValue(0).ToString();
                Label2.Text = reader.GetValue(1).ToString();
                Label20.Text = reader.GetValue(2).ToString();
                Label21.Text = reader.GetValue(3).ToString();
                Label22.Text = reader.GetValue(4).ToString();
                Label23.Text = reader.GetValue(5).ToString();
                Label24.Text = reader.GetValue(6).ToString();
                Label25.Text = reader.GetValue(7).ToString();
                Label26.Text = reader.GetValue(8).ToString();
                Label27.Text = reader.GetValue(9).ToString();
                Label28.Text = reader.GetValue(11).ToString();
                Label29.Text = reader.GetValue(10).ToString();
                Label30.Text = reader.GetValue(16).ToString();
                Label31.Text = reader.GetValue(15).ToString();
                Label32.Text = reader.GetValue(14).ToString();
                Label33.Text = reader.GetValue(13).ToString();
                Label34.Text = reader.GetValue(12).ToString();
            }

        }
        catch (Exception ex)
        {
            Response.Redirect("~/Problem.aspx");
        }
    }//fill Combo

}
```

```
public partial class ACEC_NadraService : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        NadraWebService.ValidateNIC wsValidateNIC = new NadraWebService.ValidateNIC();
        String NIC = TextBox1.Text.Trim();
        Boolean flg = wsValidateNIC.Validate(NIC);
        if(flg ==true){
            msg.Text = "NIC is Valid";
        }else{
            msg.Text = "Invalid NIC Try Again";
        }

    }
}
```

```
public partial class PrepareDROList : System.Web.UI.Page
{
    Database db;
    long id;
    protected void Page_Load(object sender, EventArgs e)
    {

        db = DatabaseFactory.CreateDatabase("LocalSqlConnection");

        if (!IsPostBack)
        {
            FillCombo(DropDownList4, "SelectProvince");
            FillCombo(DropDownList1, "SelectEmployee");
        }
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        try
        {
            if (DropDownList1.SelectedIndex > 0)
            {
                DbCommand dbcmd = db.GetSqlStringCommand("AddDRO");
                dbcmd.CommandType = CommandType.StoredProcedure;
                db.AddInParameter(dbcmd, "Employee_Id", DbType.Int32, DropDownList1.SelectedValue);
                db.AddInParameter(dbcmd, "Category_Id", DbType.Int32, 3);
                db.AddInParameter(dbcmd, "District_Id", DbType.Int32, DropDownList3.SelectedValue);
                db.AddOutParameter(dbcmd, "flg", DbType.Int16, 1);
                db.ExecuteNonQuery(dbcmd);

                String flg = db.GetParameterValue(dbcmd, "flg").ToString();

                if (flg == "1")
                {
                    lblMsg.Text = "Sorry: Employee is already assigned to some designation.";
                }
                else if (flg == "2")
                {
                    lblMsg.Text = "Sorry: This District is already assigned to some employee.";
                }
                else if (flg == "3")
                {
                    FillCombo(DropDownList1, "SelectEmployee");
                    lblMsg.Text = "Confirmation: DRO is successfully added.";
                }
                else
                {
                    lblMsg.Text = "Problem: Illegal Operation!, Try again later.";
                }
            }

        }
        catch (Exception ex) {
            lblMsg.Text = "Sorry: Server is not responding at this time, Try some later time.";
        }

    }
    protected void FillCombo(DropDownList combo, string spName)
    {
        try
```

```
        {
            DbCommand cmd = db.GetSqlStringCommand(spName);
            cmd.CommandType = CommandType.StoredProcedure;

            IDataReader reader = db.ExecuteReader(cmd);
            int i = 1;
            combo.Items.Clear();
            combo.Items.Add("");
            while (reader.Read())
            {
                combo.Items.Add(reader.GetValue(1).ToString());
                combo.Items[i].Value = reader.GetValue(0).ToString();
                i++;
            }
            combo.SelectedIndex = 0;
        }
        catch (Exception ex)
        {
            lblMsg.Text = "Sorry: Data is not availabe at this time to fill list..";
        }
    }//fill Simple Combo
    protected void FillCombo(DropDownList combo, string spName, string fieldName, int id)
    {
        try
        {
            DbCommand cmd = db.GetSqlStringCommand(spName);
            cmd.CommandType = CommandType.StoredProcedure;
            db.AddInParameter(cmd, fieldName, DbType.Int64, id);
            IDataReader reader = db.ExecuteReader(cmd);
            int i = 1;
            combo.Items.Clear();
            combo.Items.Add("");
            while (reader.Read())
            {
                combo.Items.Add(reader.GetValue(1).ToString());
                combo.Items[i].Value = reader.GetValue(0).ToString();
                i++;
            }
            combo.SelectedIndex = 0;
        }
        catch (Exception ex)
        {
            lblMsg.Text = "Sorry: Data is not availabe at this time to fill list..";
        }
    }//fill Combo


    protected void DropDownList4_SelectedIndexChanged(object sender, EventArgs e)
    {
        string pid = DropDownList4.SelectedValue;
        if (pid != "")
        {
            if (int.Parse(pid) > 0)
                FillCombo(DropDownList3, "SelectDistrictForDROAllocation", "Province_id",
int.Parse(pid));
        }
        else
        {
            DropDownList3.Items.Clear();
```

```
    }
  }

}
```

```
public partial class PrepareElectrolRoll : System.Web.UI.Page
{
  Database db1;
  protected void Page_Load(object sender, EventArgs e)
  {

    db1 = DatabaseFactory.CreateDatabase("LocalSqlConnection");
    if (!IsPostBack) {
      FillCombo(cmbProvince, "SelectProvince");
    }

  }
  protected void Button1_Click(object sender, EventArgs e)
  {
    try
    {
      DbCommand dbcmd = db1.GetSqlStringCommand("AddVoter");
      dbcmd.CommandType = CommandType.StoredProcedure;

      db1.AddInParameter(dbcmd, "Voter_Name", DbType.String, TextBox2.Text);
      db1.AddInParameter(dbcmd, "Voter_Father_HusbandName", DbType.String, TextBox3.Text);
      db1.AddInParameter(dbcmd, "Voter_HomeNo", DbType.String, TextBox4.Text);
      db1.AddInParameter(dbcmd, "Voter_StreetBlock", DbType.String, TextBox5.Text);
      db1.AddInParameter(dbcmd, "Voter_NIC", DbType.String, TextBox6.Text);
      db1.AddInParameter(dbcmd, "Voter_DOB", DbType.String, TextBox7.Text);
      db1.AddInParameter(dbcmd, "Voter_Religon", DbType.String, TextBox8.Text);
      db1.AddInParameter(dbcmd, "Voter_HomePhone", DbType.String, TextBox9.Text);
      db1.AddInParameter(dbcmd, "Voter_MobilePhone", DbType.String, TextBox10.Text);
      db1.AddInParameter(dbcmd, "Voter_FamilyNo", DbType.Int16, TextBox11.Text);
      db1.AddInParameter(dbcmd, "Voter_Gender", DbType.String,
RadioButtonList1.SelectedValue);
      db1.AddInParameter(dbcmd, "Voter_Female_Category", DbType.String, TextBox13.Text);
      db1.AddInParameter(dbcmd, "PollingStation_Id", DbType.Int32,
cmbPollStation.SelectedValue);
      db1.AddOutParameter(dbcmd, "Flg", DbType.Int16, 2);
      db1.ExecuteNonQuery(dbcmd);
      string flg = dbcmd.Parameters[13].Value.ToString();
      if (flg == "1")
        lblMsg.Text = "Confirmation: Record is added successfully..";
      else
        lblMsg.Text = "Sorry: Voter with this NIC is already exists, Record is not Added.";
    }
    catch (Exception ex)
    {
      lblMsg.Text = ex.ToString();

      //lblMsg.Text = "Sorry: Server is not responding, Record is not added successfully..";
     }


  }


  protected void FillCombo(DropDownList combo,string spName) {
```

```
    try
    {
        DbCommand cmd = db1.GetSqlStringCommand(spName);
        cmd.CommandType = CommandType.StoredProcedure;

        IDataReader reader = db1.ExecuteReader(cmd);
        int i = 1;
        combo.Items.Clear();
        combo.Items.Add("");
        while (reader.Read())
        {
            combo.Items.Add(reader.GetValue(1).ToString());
            combo.Items[i].Value = reader.GetValue(0).ToString();
            i++;
        }
        combo.SelectedIndex = 0;
    }
    catch(Exception ex) {
        //Response.Write(ex.Message);
        Response.Redirect("~/Problem.aspx");
     }
}//fill Simple Combo
protected void FillCombo(DropDownList combo, string spName,string fieldName,int id)
{
    try
    {
        DbCommand cmd = db1.GetSqlStringCommand(spName);
        cmd.CommandType = CommandType.StoredProcedure;
        db1.AddInParameter(cmd, fieldName, DbType.Int64, id);
        IDataReader reader = db1.ExecuteReader(cmd);
        int i = 1;
        combo.Items.Clear();
        combo.Items.Add("");
        while (reader.Read())
        {
            combo.Items.Add(reader.GetValue(1).ToString());
            combo.Items[i].Value = reader.GetValue(0).ToString();
            i++;
        }
        combo.SelectedIndex = 0;
    }
    catch (Exception ex)
    {
        //Response.Write(ex.Message);
        Response.Redirect("~/Problem.aspx");
    }
}//fill Combo


protected void cmbProvince_SelectedIndexChanged(object sender, EventArgs e)
{
    string pid = cmbProvince.SelectedValue;
    if (pid != "")
    {
        if (int.Parse(pid) > 0)
            FillCombo(cmbDistrict, "SelectDistrict", "Province_id", int.Parse(pid));
    }
    else {
        cmbDistrict.Items.Clear();
        cmbTown.Items.Clear();
```

```
            cmbUnion.Items.Clear();
            cmbPollStation.Items.Clear();
        }
    }
    protected void cmbDistrict_SelectedIndexChanged(object sender, EventArgs e)
    {
        string pid = cmbDistrict.SelectedValue;
        if (pid != "")
        {
            if (int.Parse(pid) > 0)
                FillCombo(cmbTown, "SelectTown", "District_Id", int.Parse(pid));
        }
        else
        {
            cmbTown.Items.Clear();
            cmbUnion.Items.Clear();
            cmbPollStation.Items.Clear();
        }
    }
    protected void cmbTown_SelectedIndexChanged(object sender, EventArgs e)
    {
        string pid = cmbTown.SelectedValue;
        if (pid != "")
        {
            if (int.Parse(pid) > 0)
                FillCombo(cmbUnion, "SelectUnionCouncil", "TownTehsil_Id", int.Parse(pid));
        }
        else
        {
            cmbUnion.Items.Clear();
            cmbPollStation.Items.Clear();
        }
    }
    protected void cmbUnion_SelectedIndexChanged(object sender, EventArgs e)
    {
        string pid = cmbUnion.SelectedValue;
        if (pid != "")
        {
            if (int.Parse(pid) > 0)
                FillCombo(cmbPollStation, "SelectPollingStation", "UnionCouncil_Id", int.Parse(pid));
        }
        else
        {
            cmbPollStation.Items.Clear();
        }
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        TextBox2.Text = "";
        TextBox3.Text = "";
        TextBox4.Text = "";
        TextBox5.Text = "";
        TextBox6.Text = "";
        TextBox7.Text = "";
        TextBox8.Text = "";
        TextBox9.Text = "";
        TextBox10.Text = "";
        TextBox11.Text = "";
        TextBox13.Text = "";
        //RadioButtonList1.Items.Clear();
```

```
        //cmbProvince.Items.Clear();
        cmbDistrict.Items.Clear();
        cmbTown.Items.Clear();
        cmbUnion.Items.Clear();
        cmbPollStation.Items.Clear();
    }
    protected void cmbPollStation_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (cmbPollStation.SelectedIndex > 0)
            Button1.Visible = true;
        else
            Button1.Visible = false;
    }
}
```

```
public partial class PrepareEmployeeList : System.Web.UI.Page
{
    Database db;
    protected void Page_Load(object sender, EventArgs e)
    {
        db = DatabaseFactory.CreateDatabase("LocalSqlConnection");

    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        try
        {
            lblMsg.Text = "";
            DbCommand dbcmd = db.GetSqlStringCommand("AddEmployee");
            dbcmd.CommandType = CommandType.StoredProcedure;

            db.AddInParameter(dbcmd, "Employee_FirstName", DbType.String, TextBox2.Text);
            db.AddInParameter(dbcmd, "Employee_LastName", DbType.String, TextBox3.Text);
            db.AddInParameter(dbcmd, "Employee_FatherName", DbType.String, TextBox4.Text);
            db.AddInParameter(dbcmd, "Employee_NIC_No", DbType.String, TextBox5.Text);
            db.AddInParameter(dbcmd, "Employee_EmailID", DbType.String, TextBox6.Text);
            db.AddInParameter(dbcmd, "Employee_Fax_No", DbType.String, TextBox7.Text);
            db.AddInParameter(dbcmd, "Employee_HomePhoneNo", DbType.String, TextBox8.Text);
            db.AddInParameter(dbcmd, "Employee_OfficePhoneNo", DbType.String, TextBox9.Text);
            db.AddInParameter(dbcmd, "Employee_MobileNo", DbType.String, TextBox10.Text);
            db.AddInParameter(dbcmd, "User_Name", DbType.String, txtUserName.Text);
            db.AddInParameter(dbcmd, "Pwd", DbType.String, txtPassword.Text);
            db.AddOutParameter(dbcmd, "Flg", DbType.Int16, 2);
            db.ExecuteNonQuery(dbcmd);
            string flg = dbcmd.Parameters[11].Value.ToString();
            lblMsg.Text = flg;
            if (flg == "1")
            {
                lblMsg.Text = "Confirmation: New Employee is successfully added.";
            }
            else if (flg == "2")
            {
                lblMsg.Text = "Sorry: Add Failed, Employee with this login name already added.";
            }
            else
            {
                lblMsg.Text = "Sorry: Add Failed, Employee with this NiC is already added.";
            }
```

```
        }
        catch (Exception ex)
        {
            lblMsg.Text = "Sorry: Server is not responding at this time, Try again at some later time.";
        }



    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        TextBox2.Text = "";
        TextBox3.Text = "";
        TextBox4.Text = "";
        TextBox5.Text = "";
        TextBox6.Text = "";
        TextBox7.Text = "";
        TextBox8.Text = "";
        TextBox9.Text = "";
        TextBox10.Text = "";
        lblMsg.Text = "";
    }
}
```

```
public partial class PrepareROList : System.Web.UI.Page
{
    Database db;
    protected void Page_Load(object sender, EventArgs e)
    {
        db = DatabaseFactory.CreateDatabase("LocalSqlConnection");
        if (!IsPostBack)
        {
            FillCombo(cmbProvince, "SelectProvince");
            FillCombo(DropDownList1, "SelectEmployee");
        }

    }
    protected void FillCombo(DropDownList combo, string spName)
    {
        try
        {
            DbCommand cmd = db.GetSqlStringCommand(spName);
            cmd.CommandType = CommandType.StoredProcedure;

            IDataReader reader = db.ExecuteReader(cmd);
            int i = 1;
            combo.Items.Clear();
            combo.Items.Add("");
            while (reader.Read())
            {
                combo.Items.Add(reader.GetValue(1).ToString());
                combo.Items[i].Value = reader.GetValue(0).ToString();
                i++;
            }
            combo.SelectedIndex = 0;
        }
        catch (Exception ex)
        {
            lblMsg.Text = "Sorry: Data is not availabe at this time to fill list..";
```

```
    }
}//fill Simple Combo
protected void FillCombo(DropDownList combo, string spName, string fieldName, int id)
{
    try
    {
        DbCommand cmd = db.GetSqlStringCommand(spName);
        cmd.CommandType = CommandType.StoredProcedure;
        db.AddInParameter(cmd, fieldName, DbType.Int64, id);
        IDataReader reader = db.ExecuteReader(cmd);
        int i = 1;
        combo.Items.Clear();
        combo.Items.Add("");
        while (reader.Read())
        {
            combo.Items.Add(reader.GetValue(1).ToString());
            combo.Items[i].Value = reader.GetValue(0).ToString();
            i++;
        }
        combo.SelectedIndex = 0;
    }
    catch (Exception ex)
    {
        lblMsg.Text = "Sorry: Data is not availabe at this time to fill list..";
    }
}//fill Combo

protected void Button1_Click(object sender, EventArgs e)
{
    try
    {
        DbCommand dbcmd = db.GetSqlStringCommand("AddRO");
        dbcmd.CommandType = CommandType.StoredProcedure;
        db.AddInParameter(dbcmd, "Employee_Id", DbType.Int32, DropDownList1.SelectedValue);
        db.AddInParameter(dbcmd, "Category_Id", DbType.Int32, 4);
        db.AddInParameter(dbcmd, "TownTehsil_Id", DbType.Int32, cmbTown.SelectedValue);
        db.AddOutParameter(dbcmd, "flg", DbType.Int16, 1);
        db.ExecuteNonQuery(dbcmd);
        String flg = db.GetParameterValue(dbcmd, "flg").ToString();

        if (flg == "1")
        {
            lblMsg.Text = "Sorry: Employee is already assigned to some designation.";
        }
        else if (flg == "2")
        {
            lblMsg.Text = "Sorry: This TownTehsil is already assigned to some employee.";
        }
        else if (flg == "3")
        {
            lblMsg.Text = "Confirmation: RO is successfully added.";
            FillCombo(DropDownList1, "SelectEmployee");
        }
        else
        {
            lblMsg.Text = "Problem: Illegal Operation!, Try again later.";
        }
    }
    catch (Exception ex)
    {
```

```
            lblMsg.Text = "Sorry: Server is not responding at this time, Try some later time.";
        }

    }
    protected void cmbProvince_SelectedIndexChanged(object sender, EventArgs e)
    {
        string pid = cmbProvince.SelectedValue;
        cmbDistrict.Items.Clear();
        cmbTown.Items.Clear();
        if (pid != "")
        {
            if (int.Parse(pid) > 0)
                FillCombo(cmbDistrict, "SelectDistrict", "Province_id", int.Parse(pid));
        }

    }
    protected void cmbDistrict_SelectedIndexChanged(object sender, EventArgs e)
    {
        string pid = cmbDistrict.SelectedValue;
        cmbTown.Items.Clear();
        if (pid != "")
        {
            if (int.Parse(pid) > 0)
                FillCombo(cmbTown, "SelectTownForROAllocation", "District_Id", int.Parse(pid));
        }
    }
}
```

```
public partial class ROListCECDistribution : System.Web.UI.Page
{
    Database db1, db2;
    int droid;
    protected void Page_Load(object sender, EventArgs e)
    {
        db1 = DatabaseFactory.CreateDatabase("LocalSqlConnection");
        Connect();
        if (!IsPostBack)
        {
            FillCombo(cmbProvince, "SelectProvince");
        }


    }
    protected void FillCombo(DropDownList combo, string spName)
    {
        try
        {
            DbCommand cmd = db1.GetSqlStringCommand(spName);
            cmd.CommandType = CommandType.StoredProcedure;

            IDataReader reader = db1.ExecuteReader(cmd);
            int i = 1;
            combo.Items.Clear();
            combo.Items.Add("");
            while (reader.Read())
            {
                combo.Items.Add(reader.GetValue(1).ToString());
                combo.Items[i].Value = reader.GetValue(0).ToString();
                i++;
```

```
        }
        combo.SelectedIndex = 0;
    }
    catch (Exception ex)
    {
        //Response.Write(ex.Message);
        Response.Redirect("~/Problem.aspx");
    }
}//fill Simple Combo
protected void cmbProvince_SelectedIndexChanged(object sender, EventArgs e)
{
    string pid = cmbProvince.SelectedValue;
    if (pid != "")
    {
        if (int.Parse(pid) > 0)
            SetCECByProvince(int.Parse(pid));
    }
    else
    {
        lblDRO.Text = "";
        btnSend.Enabled = false;
    }

}
protected void SetCECByProvince(int did)
{
    try
    {
        lblDRO.Text = "";
        DbCommand cmd = db1.GetSqlStringCommand("SelectCEC");

        cmd.CommandType = CommandType.StoredProcedure;
        db1.AddInParameter(cmd, "Province_Id", DbType.Int64, did);
        IDataReader reader = db1.ExecuteReader(cmd);
        while (reader.Read())
        {
            lblDRO.Text = reader.GetValue(1).ToString();
            ViewState.Add("droid", reader.GetValue(0).ToString());
            btnSend.Enabled = true;
        }
    }
    catch (Exception ex)
    {
        //Response.Write(ex.Message);
        Response.Redirect("~/Problem.aspx");
    }
}

protected Boolean Connect()
{
    try
    {
        db2 = DatabaseFactory.CreateDatabase("ServerSqlConnection");
        return true;

    }
    catch (Exception ex)
    {
        return false;
    }
```

```
    }
    protected void btnSend_Click(object sender, EventArgs e)
    {
        string did = cmbProvince.SelectedValue;
        if (did != "")
        {
            if (int.Parse(did) > 0)
            {
                droid = int.Parse(ViewState["droid"].ToString());

                if (UpdateVoterListOnServer(int.Parse(did), droid) == true)
                    lblMSG.Text = "Records Send!";
                else
                    lblMSG.Text = "Nothing to Send.";
            }//int.parse
        }
        else
        {
            lblMSG.Text = "Wrong Province Selection!";
        }

    }
    private Boolean UpdateVoterListOnServer(int DistID, int DROID)
    {

        try
        {
            DbCommand cmd = db2.GetSqlStringCommand("SendToCECROList");
            cmd.CommandType = CommandType.StoredProcedure;
            db2.AddInParameter(cmd, "Province_Id", DbType.Int32, DistID);
            db2.AddInParameter(cmd, "CEC_Id", DbType.Int32, DROID);

            int count = db2.ExecuteNonQuery(cmd);
            if (count > 0)
            {
                return true;
            }
            return false;
        }
        catch (Exception ex)
        {
            return false;
        }


    }
}
```

```
public partial class ShowDRO : System.Web.UI.Page
{
    Database db1;
    string id = "";
    protected void Page_Load(object sender, EventArgs e)
    {
        db1 = DatabaseFactory.CreateDatabase("LocalSqlConnection");

        try
        {
            id = Request.QueryString["id"];
```

```
        }
        catch (Exception ex)
        {
        }

        if (id == null)
        {
        }
        else
        {
            if (id.Trim() != "")
            {
                ShowROInfo(int.Parse(id));
                LinkButton1.PostBackUrl = "~/ACEC/UpdateDRO.aspx?id=" + id;
            }

        }

    }
    protected void ShowROInfo(int id)
    {
        try
        {
            DbCommand cmd = db1.GetSqlStringCommand("GetDRO");
            cmd.CommandType = CommandType.StoredProcedure;
            db1.AddInParameter(cmd, "Id", DbType.Int64, id);
            IDataReader reader = db1.ExecuteReader(cmd);

            while (reader.Read())
            {
                Label1.Text = reader.GetValue(3).ToString();
                Label6.Text = reader.GetValue(2).ToString();
                Label12.Text = reader.GetValue(0).ToString();
                Label13.Text = reader.GetValue(1).ToString();


            }

        }
        catch (Exception ex)
        {
            //Response.Write(ex.Message);
            Response.Redirect("~/Problem.aspx");
        }
    }//fill Combo
}
```

```
public partial class ShowRO : System.Web.UI.Page
{
    Database db1;
    string id = "";
    protected void Page_Load(object sender, EventArgs e)
    {
        db1 = DatabaseFactory.CreateDatabase("LocalSqlConnection");

        try
        {
            id = Request.QueryString["id"];
        }
        catch (Exception ex)
        {
        }

        if (id == null)
        {
        }
        else
        {
            if (id.Trim() != "")
            {
                ShowROInfo(int.Parse(id));
                LinkButton1.PostBackUrl = "~/ACEC/UpdateRO.aspx?id=" + id;
            }
        }
    }
    protected void ShowROInfo(int id)
    {
        try
        {
            DbCommand cmd = db1.GetSqlStringCommand("GetRO");
            cmd.CommandType = CommandType.StoredProcedure;
            db1.AddInParameter(cmd, "Id", DbType.Int64, id);
            IDataReader reader = db1.ExecuteReader(cmd);

            while (reader.Read())
            {
                Label1.Text = reader.GetValue(4).ToString();
                Label6.Text = reader.GetValue(3).ToString();
                Label11.Text = reader.GetValue(2).ToString();
                Label12.Text = reader.GetValue(0).ToString();
                Label13.Text = reader.GetValue(1).ToString();


            }

        }
        catch (Exception ex)
        {
            Response.Redirect("~/Problem.aspx");
        }
    }//fill Combo
}
```

```
public partial class UpdateDRO : System.Web.UI.Page
{
    Database db;
    string id = "";
    protected void Page_Load(object sender, EventArgs e)
    {
        db = DatabaseFactory.CreateDatabase("LocalSqlConnection");
        try
        {
            id = Request.QueryString["id"];
            if (id == null)
            {
            }
            else
            {
                if (id.Trim() != "")
                {

                }
                else
                {
                    Response.Redirect("~/Problem.aspx");
                }

            }

        }
        catch (Exception ex)
        { }
        if (!IsPostBack)
        {

            FillCombo(DropDownList4, "SelectProvince");
            //FillCombo(DropDownList1, "SelectEmployee");
            //int pid = 3;
            //FillCombo(DropDownList2, "SelectEmployeeCategory", "Category_Id", pid);

try
            {
                if (id == null)
                {
                }
                else
                {
                    if (id.Trim() != "")
                    {
                        Fill(int.Parse(id));
                    }

                }

            }
            catch (Exception ex)
            { }
        }


    }
    protected void Fill(int id)
    {
```

```
    try
    {
        DbCommand cmd = db.GetSqlStringCommand("GetDRO");
        cmd.CommandType = CommandType.StoredProcedure;
        db.AddInParameter(cmd, "Id", DbType.Int64, id);
        IDataReader reader = db.ExecuteReader(cmd);

        while (reader.Read())
        {
            DropDownList4.SelectedValue = reader.GetValue(5).ToString();
            LoadDistrictsBySelectedProvince();
            DropDownList3.SelectedValue = reader.GetValue(4).ToString();
            Label3.Text = reader.GetValue(1).ToString();

        }

    }
    catch (Exception ex)
    {
        //Response.Write(ex.Message);
        Response.Redirect("~/Problem.aspx");
    }
}//fill Combo
protected void DropDownList4_SelectedIndexChanged(object sender, EventArgs e)
{
    LoadDistrictsBySelectedProvince();
}
private void LoadDistrictsBySelectedProvince() {
    string pid = DropDownList4.SelectedValue;
    if (pid != "")
    {
        if (int.Parse(pid) > 0)
            FillCombo(DropDownList3, "SelectDistrict", "Province_id", int.Parse(pid));
    }
    else
    {
        DropDownList3.Items.Clear();

    }
}
protected void DropDownList2_SelectedIndexChanged(object sender, EventArgs e)
{

}

protected void Button1_Click(object sender, EventArgs e)
{
    int flag = 0;
    if (id == null)
    {
    }
    else
    {
        if (id.Trim() != "")
        {
            try
            {

                if (UpdateR(int.Parse(id)) == true)
                {
```

```
                flag = 1;
               // break;
              }
              else
                flag = 2;
            }
            catch (Exception ex)
            {
              flag = 3;
            }
            if (flag == 3)
            {
              Response.Redirect("~/Problem.aspx");
            }
            else if (flag == 1) {
              Label6.Text = "Confirmation: Record is Updated.";
            }
            else if (flag == 2)
            {
              Label6.Text = "Sorry: Record is not Updated.";
            }


        }

    }




}
private Boolean UpdateR(int id)
{
    try
    {
      DbCommand dbcmd = db.GetSqlStringCommand("UpdateDRO");
      db.AddInParameter(dbcmd, "Id", DbType.Int32, id);
      dbcmd.CommandType = CommandType.StoredProcedure;
      //db.AddInParameter(dbcmd, "Employee_Id", DbType.Int32, DropDownList1.SelectedValue);
      //db.AddInParameter(dbcmd, "Category_Id", DbType.Int32, DropDownList2.SelectedValue);
      db.AddInParameter(dbcmd, "District_Id", DbType.Int32, DropDownList3.SelectedValue);
      db.AddOutParameter(dbcmd, "flg", DbType.Int16, 1);

      db.ExecuteNonQuery(dbcmd);

      String flg = db.GetParameterValue(dbcmd, "flg").ToString();

      if (flg == "1")
      {
        Label6.Text = "This district is already assigned to an employee.";
      }
      else if (flg == "2")
      {
        Label6.Text = "DRO is successfully Updated.";
      }
      else if (flg == "0")
      {
        Label6.Text = "DRO is not updated! Try again later";
```

```
        }


        return true;
    }
    catch (Exception ex)
    {
        return false;
    }
}
protected void FillCombo(DropDownList combo, string spName)
{
    try
    {
        DbCommand cmd = db.GetSqlStringCommand(spName);
        cmd.CommandType = CommandType.StoredProcedure;

        IDataReader reader = db.ExecuteReader(cmd);
        int i = 1;
        combo.Items.Clear();
        combo.Items.Add("");
        while (reader.Read())
        {
            combo.Items.Add(reader.GetValue(1).ToString());
            combo.Items[i].Value = reader.GetValue(0).ToString();
            i++;
        }
        combo.SelectedIndex = 0;
    }
    catch (Exception ex)
    {
        Response.Write(ex.Message);
    }
}//fill Simple Combo
protected void FillCombo(DropDownList combo, string spName, string fieldName, int id)
{
    try
    {
        DbCommand cmd = db.GetSqlStringCommand(spName);
        cmd.CommandType = CommandType.StoredProcedure;
        db.AddInParameter(cmd, fieldName, DbType.Int64, id);
        IDataReader reader = db.ExecuteReader(cmd);
        int i = 1;
        combo.Items.Clear();
        combo.Items.Add("");
        while (reader.Read())
        {
            combo.Items.Add(reader.GetValue(1).ToString());
            combo.Items[i].Value = reader.GetValue(0).ToString();
            i++;
        }
        combo.SelectedIndex = 0;
    }
    catch (Exception ex)
    {
        //Response.Write(ex.Message);
        Response.Redirect("~/Problem.aspx");
    }
}//fill Combo
```

```
    protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
    {

    }
    protected void DropDownList3_SelectedIndexChanged(object sender, EventArgs e)
    {
        Label6.Text = "";
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        Response.Redirect("ShowDRO.aspx?id=" + id);
    }
}
```

```
public partial class UpdateRO : System.Web.UI.Page
{
    Database db;
    string id = "";
    protected void Page_Load(object sender, EventArgs e)
    {
        db = DatabaseFactory.CreateDatabase("LocalSqlConnection");
        try
        {
            id = Request.QueryString["id"];
            if (id == null)
            {
            }
            else
            {
                if (id.Trim() != "")
                {

                }
                else
                {
                    Response.Write("Error");
                }

            }

        }
        catch (Exception ex)
        { }
        if (!IsPostBack)
        {
            FillCombo(cmbProvince, "SelectProvince");
            //int pid = 4;
            //FillCombo(DropDownList2, "SelectEmployeeCategory", "Category_Id", pid);

            //FillCombo(DropDownList2, "SelectEmployeeCategory");
            //FillCombo(DropDownList1, "SelectEmployee");
            try
            {
                if (id == null)
                {
                }
                else
                {
                    if (id.Trim() != "")
                    {
                        Fill(int.Parse(id));
                    }

                }

            }
            catch (Exception ex)
            { }
        }


        }
        protected void Fill(int id)
        {
```

```
            try
            {
               DbCommand cmd = db.GetSqlStringCommand("GetRO");
               cmd.CommandType = CommandType.StoredProcedure;
               db.AddInParameter(cmd, "Id", DbType.Int64, id);
               IDataReader reader = db.ExecuteReader(cmd);

               while (reader.Read())
               {
                   cmbProvince.SelectedValue = reader.GetValue(5).ToString();
                  LoadDistrictBySelectProvince();

                  cmbDistrict.SelectedValue = reader.GetValue(6).ToString();
                  LoadTownTehsilBySelectDistrict();

                  cmbTown.SelectedValue = reader.GetValue(7).ToString();
                  Label3.Text = reader.GetValue(1).ToString();
                  Button1.Visible = true;
               }

            }
            catch (Exception ex)
            {
               Label7.Text = "Sorry: Currently required information is not available, Try again at some later
time.";
            }
         }//fill Combo
      protected void FillCombo(DropDownList combo, string spName)
      {
         try
         {
            DbCommand cmd = db.GetSqlStringCommand(spName);
            cmd.CommandType = CommandType.StoredProcedure;

            IDataReader reader = db.ExecuteReader(cmd);
            int i = 1;
            combo.Items.Clear();
            combo.Items.Add("");
            while (reader.Read())
            {
               combo.Items.Add(reader.GetValue(1).ToString());
               combo.Items[i].Value = reader.GetValue(0).ToString();
               i++;
            }
            combo.SelectedIndex = 0;
         }
         catch (Exception ex)
         {
            Label7.Text = "Sorry: Data is not available this time.";
         }
      }//fill Simple Combo
      protected void FillCombo(DropDownList combo, string spName, string fieldName, int id)
      {
         try
         {
            DbCommand cmd = db.GetSqlStringCommand(spName);
            cmd.CommandType = CommandType.StoredProcedure;
            db.AddInParameter(cmd, fieldName, DbType.Int64, id);
            IDataReader reader = db.ExecuteReader(cmd);
            int i = 1;
```

```
            combo.Items.Clear();
            combo.Items.Add("");
            while (reader.Read())
            {
                combo.Items.Add(reader.GetValue(1).ToString());
                combo.Items[i].Value = reader.GetValue(0).ToString();
                i++;
            }
            combo.SelectedIndex = 0;
        }
        catch (Exception ex)
        {
            Label7.Text = "Sorry: Data is not available this time.";
        }
    }//fill Combo


    protected void  Button1_Click(object sender, EventArgs e)
    {
        if (id == null)
        {
        }
        else
        {
            int flag = 0;
            try
            {
                if (id.Trim() != "")
                {
                    if (UpdateR(int.Parse(id)) == true)
                    {
                        flag = 1;
                    }
                    else
                        //Response.Write("Updation failed,Try again!");
                        flag = 2;
                }
            }
            catch (Exception ex)
            {
                flag = 3;
            }
            if (flag == 1)
            {
                Label6.Text = "Confirmation: Record is Updated.";
            }
            else if (flag == 2)
            { Label6.Text = "Sorry: Record is not Updated."; }
            else if (flag == 3)
            { Response.Redirect("~/Problem.aspx"); }
        }
}
    private Boolean UpdateR(int id)
    {
        try
        {
            DbCommand dbcmd = db.GetSqlStringCommand("UpdateRO");
            dbcmd.CommandType = CommandType.StoredProcedure;
            db.AddInParameter(dbcmd, "Id", DbType.Int32, id);
            //db.AddInParameter(dbcmd, "Employee_Id", DbType.Int32, DropDownList1.SelectedValue);
```

```
        //db.AddInParameter(dbcmd, "Category_Id", DbType.Int32, 4);
        db.AddInParameter(dbcmd, "TownTehsil_Id", DbType.Int32, cmbTown.SelectedValue);
        db.AddOutParameter(dbcmd, "flg", DbType.Int16, 1);
        db.ExecuteNonQuery(dbcmd);
        String flg = db.GetParameterValue(dbcmd, "flg").ToString();

        if (flg == "1")
        {
            Label7.Text = "Sorry: This Town Tehsil is already assigned to an employee.";
        }
        else if (flg == "2")
        {
            Label7.Text = "Conrirmation: RO is successfully Updated.";
        }
        else if (flg == "0")
        {
            Label7.Text = "Sorry: RO is not updated! Try again later";
        }
        return true;
    }
    catch (Exception ex)
    {
        return false;
    }
}
protected void cmbTown_SelectedIndexChanged(object sender, EventArgs e)
{
    Label7.Text = "";
}


protected void cmbProvince_SelectedIndexChanged(object sender, EventArgs e)
{
    LoadDistrictBySelectProvince();
}
private void LoadDistrictBySelectProvince()
{
    try
    {
        string pid = cmbProvince.SelectedValue;
        cmbDistrict.Items.Clear();
        cmbTown.Items.Clear();
        if (pid != "")
        {
            if (int.Parse(pid) > 0)
                FillCombo(cmbDistrict, "SelectDistrict", "Province_id", int.Parse(pid));
        }
    }
    catch (Exception ex)
    {

    }

}

protected void cmbDistrict_SelectedIndexChanged(object sender, EventArgs e)
{
    LoadTownTehsilBySelectDistrict();

}
```

```
    private void LoadTownTehsilBySelectDistrict()
    {
        try
        {
            string pid = cmbDistrict.SelectedValue;
            cmbTown.Items.Clear();

            if (pid != "")
            {
                if (int.Parse(pid) > 0)
                    FillCombo(cmbTown, "SelectTown", "District_Id", int.Parse(pid));
            }
        }
        catch (Exception ex)
        {

        }

    }

}
```

```
public partial class UpdateVoterList : System.Web.UI.Page
{
    Database db1;
    string id = "";
    protected void Page_Load(object sender, EventArgs e)
    {
        db1 = DatabaseFactory.CreateDatabase("LocalSqlConnection");
        try
        {
            id = Request.QueryString["id"];
            if (id == null)
            {
            }
            else
            {
                if (id.Trim() != "")
                {

                }
                else {
                    Response.Write("Error");
                }

            }

        }
        catch (Exception ex)
        { }

        if (!IsPostBack)
        {
            FillCombo(cmbProvince, "SelectProvince");
            try
            {
                if (id == null)
                {
                }
                else
                {
                    if (id.Trim() != "")
                    {
                        Fill(int.Parse(id));
                    }

                }

            }
            catch (Exception ex)
            {}
        }




    }
    protected void Fill(int id)
    {
        try
```

```
    {
        DbCommand cmd = db1.GetSqlStringCommand("selectVoterForUpdate");
        cmd.CommandType = CommandType.StoredProcedure;
        db1.AddInParameter(cmd, "Voter_Id", DbType.Int64, id);
        IDataReader reader = db1.ExecuteReader(cmd);

        while (reader.Read())
        {
            TextBox2.Text = reader.GetValue(0).ToString();
            TextBox3.Text = reader.GetValue(1).ToString();
            TextBox4.Text = reader.GetValue(2).ToString();
            TextBox5.Text = reader.GetValue(3).ToString();
            TextBox6.Text = reader.GetValue(4).ToString();
            TextBox7.Text = reader.GetValue(5).ToString();
            TextBox8.Text = reader.GetValue(6).ToString();
            TextBox9.Text = reader.GetValue(7).ToString();
            TextBox10.Text = reader.GetValue(8).ToString();
            TextBox11.Text = reader.GetValue(9).ToString();
            TextBox13.Text = reader.GetValue(11).ToString();
            RadioButtonList1.Text = reader.GetValue(10).ToString();


            cmbProvince.SelectedValue  = reader.GetValue(16).ToString();
            LoadDistrictBySelectProvince();

            cmbDistrict.SelectedValue = reader.GetValue(15).ToString();
            LoadTownTehsilBySelectDistrict();

            cmbTown.SelectedValue = reader.GetValue(14).ToString();
            LoadUnionBySelectTown();

            cmbUnion.SelectedValue = reader.GetValue(13).ToString();
            LoadPollinStationBySelectUnion();

            cmbPollStation.SelectedValue = reader.GetValue(12).ToString();
        }

    }
    catch (Exception ex)
    {
        //Response.Write(ex.Message);
        Response.Redirect("~/Problem.aspx");
    }
}//fill Combo
protected void FillCombo(DropDownList combo, string spName)
{
    try
    {
        DbCommand cmd = db1.GetSqlStringCommand(spName);
        cmd.CommandType = CommandType.StoredProcedure;

        IDataReader reader = db1.ExecuteReader(cmd);
        int i = 1;
        combo.Items.Clear();
        combo.Items.Add("");
        while (reader.Read())
        {
            combo.Items.Add(reader.GetValue(1).ToString());
            combo.Items[i].Value = reader.GetValue(0).ToString();
            i++;
```

```
        }
        combo.SelectedIndex = 0;
    }
    catch (Exception ex)
    {
        Response.Write(ex.Message);
    }
}//fill Simple Combo
protected void FillCombo(DropDownList combo, string spName, string fieldName, int id)
{
    try
    {
        DbCommand cmd = db1.GetSqlStringCommand(spName);
        cmd.CommandType = CommandType.StoredProcedure;
        db1.AddInParameter(cmd, fieldName, DbType.Int64, id);
        IDataReader reader = db1.ExecuteReader(cmd);
        int i = 1;
        combo.Items.Clear();
        combo.Items.Add("");
        while (reader.Read())
        {
            combo.Items.Add(reader.GetValue(1).ToString());
            combo.Items[i].Value = reader.GetValue(0).ToString();
            i++;
        }
        combo.SelectedIndex = 0;
    }
    catch (Exception ex)
    {

    }
}//fill Combo

protected void Button1_Click(object sender, EventArgs e)
{
    lblMsg.Text = "";
    id = Request.QueryString["id"];
    if (id == null)
    {
    }
    else
    {
        int flag = 0;
        try
        {
            if (id.Trim() != "")
            {
                if (UpdateVoter(int.Parse(id)) == true)
                {
                    flag = 1;
                }
                else
                    flag = 2;
            }
        }
        catch (Exception ex)
        {
            flag = 3;
        }
```

```
            if (flag == 1)
            {
                lblMsg.Text = "Confirmation: Record is Updated.";
            }
            else if (flag == 2)
            { lblMsg.Text = "Sorry: Record is not Updated."; }
            else if (flag == 3)
            { Response.Redirect("~/Problem.aspx"); }

        }
    }
    private Boolean UpdateVoter(int id) {
        try
        {
            DbCommand dbcmd = db1.GetSqlStringCommand("UpdateVoter");
            dbcmd.CommandType = CommandType.StoredProcedure;

            db1.AddInParameter(dbcmd, "Voter_Id", DbType.Int32, id);
            db1.AddInParameter(dbcmd, "Voter_Name", DbType.String, TextBox2.Text);
            db1.AddInParameter(dbcmd, "Voter_Father_HusbandName", DbType.String, TextBox3.Text);
            db1.AddInParameter(dbcmd, "Voter_HomeNo", DbType.String, TextBox4.Text);
            db1.AddInParameter(dbcmd, "Voter_StreetBlock", DbType.String, TextBox5.Text);
            db1.AddInParameter(dbcmd, "Voter_NIC", DbType.String, TextBox6.Text);
            db1.AddInParameter(dbcmd, "Voter_DOB", DbType.String, TextBox7.Text);
            db1.AddInParameter(dbcmd, "Voter_Religon", DbType.String, TextBox8.Text);
            db1.AddInParameter(dbcmd, "Voter_HomePhone", DbType.String, TextBox9.Text);
            db1.AddInParameter(dbcmd, "Voter_MobilePhone", DbType.String, TextBox10.Text);
            db1.AddInParameter(dbcmd, "Voter_FamilyNo", DbType.String, TextBox11.Text);
            db1.AddInParameter(dbcmd, "Voter_Gender", DbType.String,
RadioButtonList1.SelectedValue);
            db1.AddInParameter(dbcmd, "Voter_Female_Category", DbType.String, TextBox13.Text);
            db1.AddInParameter(dbcmd, "PollingStation_Id", DbType.Int32,
cmbPollStation.SelectedValue);

            db1.ExecuteNonQuery(dbcmd);
            return true;
        }
        catch (Exception ex) {
            return false;
        }
    }

    protected void cmbProvince_SelectedIndexChanged(object sender, EventArgs e)
    {
        LoadDistrictBySelectProvince();
    }
    private void LoadDistrictBySelectProvince(){
        try
                {
                        string pid = cmbProvince.SelectedValue;
            cmbDistrict.Items.Clear();
            cmbTown.Items.Clear();
            cmbUnion.Items.Clear();
            cmbPollStation.Items.Clear();
            if (pid != "")
            {
                if (int.Parse(pid) > 0)
                    FillCombo(cmbDistrict, "SelectDistrict", "Province_id", int.Parse(pid));
            }
                }
```

```
            catch (Exception ex)
            {

            }

}
protected void cmbDistrict_SelectedIndexChanged(object sender, EventArgs e)
{
   LoadTownTehsilBySelectDistrict();

}
private void LoadTownTehsilBySelectDistrict() {
   try
   {
      string pid = cmbDistrict.SelectedValue;
      cmbTown.Items.Clear();
      cmbUnion.Items.Clear();
      cmbPollStation.Items.Clear();
      if (pid != "")
      {
         if (int.Parse(pid) > 0)
            FillCombo(cmbTown, "SelectTown", "District_Id", int.Parse(pid));
      }
   }
   catch (Exception ex)
   {

   }

}
protected void cmbTown_SelectedIndexChanged(object sender, EventArgs e)
{
   LoadUnionBySelectTown();
}
private void LoadUnionBySelectTown() {
   try
   {
      string pid = cmbTown.SelectedValue;
      cmbUnion.Items.Clear();
      cmbPollStation.Items.Clear();
      if (pid != "")
      {
         if (int.Parse(pid) > 0)
            FillCombo(cmbUnion, "SelectUnionCouncil", "TownTehsil_Id", int.Parse(pid));
      }
   }
   catch (Exception ex)
   {
   }

}
protected void cmbUnion_SelectedIndexChanged(object sender, EventArgs e)
{
   LoadPollinStationBySelectUnion();
}
private void LoadPollinStationBySelectUnion() {
   try
   {
      string pid = cmbUnion.SelectedValue;
      cmbPollStation.Items.Clear();
```

```
            if (pid != "")
            {
                if (int.Parse(pid) > 0)
                    FillCombo(cmbPollStation, "SelectPollingStation", "UnionCouncil_Id", int.Parse(pid));
            }
        }
        catch (Exception ex)
        {
        }

    }
    protected void Button2_Click(object sender, EventArgs e)
    {

    }
}
```

```
public partial class ViewDROList : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
}
```

```
public partial class ViewEmployeeList : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
}
```

```
public partial class ViewROList : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
}
```

```
public partial class ViewVoterList : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
}
```

```
public partial class VoterListCECDistribution : System.Web.UI.Page
{
    Database db1, db2;
    int cecid;
    protected void Page_Load(object sender, EventArgs e)
    {
        db1 = DatabaseFactory.CreateDatabase("LocalSqlConnection");
        Connect();
        if (!IsPostBack)
        {
            FillCombo(cmbProvince, "SelectProvince");
        }
    }
    protected void FillCombo(DropDownList combo, string spName)
    {
        try
        {
            DbCommand cmd = db1.GetSqlStringCommand(spName);
            cmd.CommandType = CommandType.StoredProcedure;

            IDataReader reader = db1.ExecuteReader(cmd);
            int i = 1;
            combo.Items.Clear();
            combo.Items.Add("");
```

```
        while (reader.Read())
        {
            combo.Items.Add(reader.GetValue(1).ToString());
            combo.Items[i].Value = reader.GetValue(0).ToString();
            i++;
        }
        combo.SelectedIndex = 0;
    }
    catch (Exception ex)
    {
        //Response.Write(ex.Message);
        Response.Redirect("~/Problem.aspx");
    }
}//fill Simple Combo
protected void cmbProvince_SelectedIndexChanged(object sender, EventArgs e)
{
    string pid = cmbProvince.SelectedValue;
    if (pid != "")
    {
        if (int.Parse(pid) > 0)
            SetCECByProvince(int.Parse(pid));
    }
    else
    {
        lblDRO.Text = "";
        btnSend.Enabled = false;
    }

}
protected void SetCECByProvince(int did)
{
    try
    {
        lblDRO.Text = "";
        DbCommand cmd = db1.GetSqlStringCommand("SelectCEC");

        cmd.CommandType = CommandType.StoredProcedure;
        db1.AddInParameter(cmd, "Province_Id", DbType.Int64, did);
        IDataReader reader = db1.ExecuteReader(cmd);
        while (reader.Read())
        {
            lblDRO.Text = reader.GetValue(1).ToString();
            ViewState.Add("cecid", reader.GetValue(0).ToString());
            btnSend.Enabled = true;
        }
    }
    catch (Exception ex)
    {
        Response.Write(ex.Message);
    }
}

protected Boolean Connect()
{
    try
    {
        db2 = DatabaseFactory.CreateDatabase("ServerSqlConnection");
        return true;

    }
```

```
        catch (Exception ex)
        {
            return false;
        }
    }
    protected void btnSend_Click(object sender, EventArgs e)
    {
        string did = cmbProvince.SelectedValue;
        if (did != "")
        {
            if (int.Parse(did) > 0)
            {
                cecid = int.Parse(ViewState["cecid"].ToString());

                if (UpdateVoterListOnServer(int.Parse(did), cecid) == true)
                    lblMSG.Text = "Records Send!";
                else
                    lblMSG.Text = "Nothing to Send.";
            }//int.parse
        }
        else
        {
            lblMSG.Text = "Wrong Province Selection!";
        }

    }
    private Boolean UpdateVoterListOnServer(int DistID, int cecid)
    {

        try
        {
            DbCommand cmd = db2.GetSqlStringCommand("SendToCECVoterList");
            cmd.CommandType = CommandType.StoredProcedure;
            db2.AddInParameter(cmd, "Province_Id", DbType.Int32, DistID);
            db2.AddInParameter(cmd, "CEC_Id", DbType.Int32, cecid);

            int count = db2.ExecuteNonQuery(cmd);
            if (count > 0)
            {
                return true;
            }
            return false;
        }
        catch (Exception ex)
        {
            return false;
        }


    }
}
```

```
public partial class ApprovePollingStationList : System.Web.UI.Page
{
    Database db1;
    protected void Page_Load(object sender, EventArgs e)
    {
        db1 = DatabaseFactory.CreateDatabase("LocalSqlConnection");
        if (!IsPostBack)
```

```
        {
            FillCombo(cmbProvince, "SelectProvince");
        }


    }
    protected void FillCombo(DropDownList combo, string spName)
    {
        try
        {
            DbCommand cmd = db1.GetSqlStringCommand(spName);
            cmd.CommandType = CommandType.StoredProcedure;

            IDataReader reader = db1.ExecuteReader(cmd);
            int i = 1;
            combo.Items.Clear();
            combo.Items.Add("");
            while (reader.Read())
            {
                combo.Items.Add(reader.GetValue(1).ToString());
                combo.Items[i].Value = reader.GetValue(0).ToString();
                i++;
            }
            combo.SelectedIndex = 0;
        }
        catch (Exception ex)
        {
            //Response.Write(ex.Message);
            Response.Redirect("~/Problem.aspx");
        }
    }//fill Simple Combo
    protected void FillCombo(DropDownList combo, string spName,
string fieldName, int id)
    {
        try
        {
            DbCommand cmd = db1.GetSqlStringCommand(spName);
            cmd.CommandType = CommandType.StoredProcedure;
            db1.AddInParameter(cmd, fieldName, DbType.Int64, id);
            IDataReader reader = db1.ExecuteReader(cmd);
            int i = 1;
            combo.Items.Clear();
            combo.Items.Add("");
            while (reader.Read())
            {
                combo.Items.Add(reader.GetValue(1).ToString());
                combo.Items[i].Value = reader.GetValue(0).ToString();
                i++;
            }
            combo.SelectedIndex = 0;
        }
        catch (Exception ex)
        {
            Response.Write(ex.Message);
        }
    }//fill Combo
    protected void cmbProvince_SelectedIndexChanged(object sender,
EventArgs e)
    {
        string pid = cmbProvince.SelectedValue;
        HideControls();
```

```
        if (cmbProvince.SelectedIndex > 0)
        {
            btnShowPollingStationList.Visible = true;
        }
        else
        {
            btnShowPollingStationList.Visible = false;

        }
        cmbDistrict.Items.Clear();
        cmbTown.Items.Clear();
        cmbUnion.Items.Clear();

        if (pid != "")
        {
            if (int.Parse(pid) > 0)
                FillCombo(cmbDistrict, "SelectDistrict",
"Province_id", int.Parse(pid));
        }

    }
    protected void cmbDistrict_SelectedIndexChanged(object sender,
EventArgs e)
    {
        string pid = cmbDistrict.SelectedValue;
        HideControls();
        cmbTown.Items.Clear();
        cmbUnion.Items.Clear();

        if (pid != "")
        {
            if (int.Parse(pid) > 0)
                FillCombo(cmbTown, "SelectTown", "District_Id",
int.Parse(pid));
        }
    }
    protected void cmbTown_SelectedIndexChanged(object sender,
EventArgs e)
    {
        string pid = cmbTown.SelectedValue;
        HideControls();
        cmbUnion.Items.Clear();
        if (pid != "")
        {
            if (int.Parse(pid) > 0)
                FillCombo(cmbUnion, "SelectUnionCouncil",
"TownTehsil_Id", int.Parse(pid));
        }


    }
    protected void cmbUnion_SelectedIndexChanged(object sender,
EventArgs e)
    {
        HideControls();
    }

    protected void btnShowPollingStationList_Click(object sender,
EventArgs e)
    {
        try
```

```
        {
            DataTable dtable = null;
            string pid = cmbProvince.SelectedValue;
            string did = cmbDistrict.SelectedValue;
            string tid = cmbTown.SelectedValue;
            string pollid = cmbUnion.SelectedValue;

            if (pollid != "")
            {
                if (int.Parse(pollid) > 0)
                {
                    GetVoters(int.Parse(pollid), "UnionCouncil_Id",
"SelectStation");
                }
            }
            else if (tid != "")
            {
                if (int.Parse(tid) > 0)
                {
                    GetVoters(int.Parse(tid), "T_id",
"SelectStationByTownTehsilId");
                }
            }
            else if (did != "")
            {
                if (int.Parse(did) > 0)
                {
                    GetVoters(int.Parse(did), "District_id",
"SelectStationByDistrictId");
                }
            }
            else if (pid != "")
            {
                if (int.Parse(pid) > 0)
                {
                    GetVoters(int.Parse(pid), "P_id",
"SelectStationByProvinceId");
                }
            }
        }
        catch (Exception ex)
        {
            Response.Redirect("~/Problem.aspx");

        }

    }
    private Boolean GetVoters(int id, string fieldName, string
spName)
    {
        try
        {
            DbCommand cmd = db1.GetSqlStringCommand(spName);
            cmd.CommandType = CommandType.StoredProcedure;
            db1.AddInParameter(cmd, fieldName, DbType.Int64, id);

            DataSet ds = db1.ExecuteDataSet(cmd);

            if (ds.Tables[0].Rows.Count > 0)
            {
                btnApprove.Visible = true;
```

```
                GridView1.Visible = true;
                lblMsg.Text = "";
            }
            else
            {
                btnApprove.Visible = false;
                GridView1.Visible = false;
                lblMsg.Text = "Sorry: Data is not available for
selected Criteria, Try with different criteria.";
            }
            GridView1.DataSource = ds.Tables[0];
            GridView1.DataBind();
            return true;
        }
        catch (Exception ex)
        {
            return false;
        }
    }

    protected void btnApprove_Click(object sender, EventArgs e)
    {
        try
        {
        DataTable dtable = GetSelectedVoters();
        if (dtable != null)
        {
            foreach (DataRow drow in dtable.Rows)
            {
                ApproveSelectedVoters(int.Parse(drow[0].ToString()));
            }
        }
    }
    catch (Exception ex)
    {
        Response.Redirect("~/Problem.aspx");

    }


    }
    private DataTable GetSelectedVoters()
    {
        try
        {
            DataTable dtable = null;
            string pid = cmbProvince.SelectedValue;
            string did = cmbDistrict.SelectedValue;
            string tid = cmbTown.SelectedValue;
            string pollid = cmbUnion.SelectedValue;

            if (pollid != "")
            {
                if (int.Parse(pollid) > 0)
                {
                    dtable = GetVotersTable(int.Parse(pollid),
"UnionCouncil_Id", "SelectStation");
                }
            }
            else if (tid != "")
            {
```

```
                if (int.Parse(tid) > 0)
                {
                    dtable = GetVotersTable(int.Parse(tid), "T_id",
"SelectStationByTownTehsilId");
                }
            }
            else if (did != "")
            {
                if (int.Parse(did) > 0)
                {
                    dtable = GetVotersTable(int.Parse(did),
"District_id", "SelectStationByDistrictId");
                }
            }
            else if (pid != "")
            {
                if (int.Parse(pid) > 0)
                {
                    dtable = GetVotersTable(int.Parse(pid), "P_id",
"SelectStationByProvinceId");
                }
            }
            return dtable;
        }
        catch (Exception ex)
        {
            return null;
        }
    }
    private Boolean ApproveSelectedVoters(int id)
    {
        try
        {
            DbCommand cmd =
db1.GetSqlStringCommand("ApproveStation");
            cmd.CommandType = CommandType.StoredProcedure;
            db1.AddInParameter(cmd, "id", DbType.Int64, id);
            db1.ExecuteNonQuery(cmd);
            return true;
        }
        catch (Exception ex)
        {
            return false;
        }
    }
    private DataTable GetVotersTable(int id, string fieldName, string
spName)
    {
        try
        {
            DbCommand cmd = db1.GetSqlStringCommand(spName);
            cmd.CommandType = CommandType.StoredProcedure;
            db1.AddInParameter(cmd, fieldName, DbType.Int64, id);

            DataSet ds = db1.ExecuteDataSet(cmd);

            return ds.Tables[0];
        }
        catch (Exception ex)
        {
            return null;
```

```
        }
    }
    private void HideControls()
    {
        GridView1.Visible = false;
        btnApprove.Visible = false;
        lblMsg.Text = "";
    }

}
```

```
public partial class PO_POAllocateVoter : System.Web.UI.Page
{
    Database db;
    DataTable tab;
    int pollId = 1;
    protected void Page_Load(object sender, EventArgs e)
    {
        int flg = -1;
        db = DatabaseFactory.CreateDatabase("LocalSqlConnection");
        try
        {
            IDataReader rdr = db.ExecuteReader(CommandType.Text,
"select PollingStation_Id from PSEmployee_Assignment where
Employee_Id=" + Request.Cookies["PO_Id"].Value.ToString());
            if (rdr.Read())
            {
                pollId = Convert.ToInt32(rdr[0].ToString());

            }
        }
        catch (Exception ex)
        {
            Response.Write(ex.ToString());
        }
        LoadBoothsAndGrid();
        if (!IsPostBack)
        {

            String NIC = "";
            try
            {
                NIC = Request.Cookies["NIC"].Value.ToString();
                flg = 1;
                if(NIC == null || NIC =="")
                    flg=0;
            }
            catch (Exception ex) {
                flg = 0;
            }
            if (flg == 1)
            {
                if (getVoterInfo(NIC) == true)
                {
                    Button2.Enabled = true;
                }
                else
                    lblMsg.Text = "Some Problem While getting Voter
Information..";
            }
            else
```

```
Response.Redirect("~/PO/POValidateVoter.aspx?id=193139");
        }
    }
    protected Boolean LoadBooths(int Pid)
    {
        try
        {
            DbCommand dbcmd =
db.GetSqlStringCommand("GetAvailableBooths");
            dbcmd.CommandType = CommandType.StoredProcedure;
            db.AddInParameter(dbcmd, "PollingStation_Id",
DbType.Int64, Pid);
            IDataReader reader = db.ExecuteReader(dbcmd);


            //DDLBoothID.DataSource = reader;
            //DDLBoothID.DataBind();
            //DDLBoothID.DataTextField = "Description";
            //DDLBoothID.DataValueField = "Booth_Id";
            int i = 0;
            DDLBoothID.Items.Clear();
            while (reader.Read())
            {
                string str = reader.GetValue(2).ToString();
                DDLBoothID.Items.Add(str);
                DDLBoothID.Items[i].Value =
reader.GetValue(0).ToString();
                i++;
            }
            if (DDLBoothID.Items.Count == 0)
                Button2.Enabled = false;
            return true;
        }
        catch (Exception e)
        {
            Response.Write(e.Message);
            return false;
        }
    }
    private void LoadBoothsAndGrid()
    {
        DataTable dtable = ShowCurrentBoothsStatus();
        if (dtable != null)
        {
            DataTable dtab = GetDataTableForGrid(dtable);
            //if (dtable.Rows.Count > 0)
            //{
            GridView1.DataSource = dtab;
            GridView1.DataBind();
            //}
        }
        LoadBooths(pollId);
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        try {
            String booth = DDLBoothID.Text;
            string boothid = DDLBoothID.SelectedValue;

            if (boothid != "") {
```

```
                DataTable dtable = AllocateBooth(lblNICNO.Text,
int.Parse(boothid));
                if (dtable != null)
                {
                    if (dtable.Rows.Count > 0)
                    {
                        DataTable dtab = GetDataTableForGrid(dtable);
                        GridView1.DataSource = dtab;
                        GridView1.DataBind();
                    }
                }
                LoadBooths(pollId);
            }

        }
        catch (Exception ex) {
        }

    }
    private void ShowAllocatedBoothGrid(string nic, string name,
string gender, string booth)
    {

        tab = (DataTable)Session["GridTab"];

        DataRow drow = tab.NewRow();
        drow[0] = nic;
        drow[1] = name;
        drow[2] = gender;
        drow[3] = booth;
        tab.Rows.Add(drow);
        Session["GridTab"] = tab;
        GridView1.DataSource = tab;

        GridView1.DataBind();
    }
    private DataTable AllocateBooth(string NIC, int boothId)
    {

        try
        {
            Button2.Enabled = false;
            lblVoterName.Text ="";
            lblNICNO.Text = "";
            lblGender.Text = "";
            DbCommand dbcmd =
db.GetSqlStringCommand("AllocateBooth");
            dbcmd.CommandType = CommandType.StoredProcedure;
            db.AddInParameter(dbcmd, "NICNO", DbType.String, NIC);
            db.AddInParameter(dbcmd, "BoothID", DbType.Int16,
boothId);
            db.AddOutParameter(dbcmd, "Flg", DbType.Boolean, 1);
            DataSet ds = db.ExecuteDataSet(dbcmd);
            return ds.Tables[0];
        }
        catch (Exception ex)
        {
            return null;
        }
    }
    protected Boolean getVoterInfo(String NIC)
```

```
    {
        try
        {
            DbCommand dbcmd = db.GetSqlStringCommand("getVoterInfo");
            dbcmd.CommandType = CommandType.StoredProcedure;

            db.AddInParameter(dbcmd, "NICNO", DbType.String, NIC);
            //db.AddOutParameter(dbcmd, "flg", DbType.Boolean, 1);
            db.AddOutParameter(dbcmd, "Gender", DbType.String, 10);
            db.AddOutParameter(dbcmd, "Name", DbType.String, 10);
            db.ExecuteNonQuery(dbcmd);
            String votergender = db.GetParameterValue(dbcmd,
"Gender").ToString();
            String votername = db.GetParameterValue(dbcmd,
"Name").ToString();
            lblGender.Text = votergender;
            lblNICNO.Text = NIC;
            lblVoterName.Text = votername;
            return true;
        }
        catch (Exception e)
        {
            return false;
        }
       }
       private DataTable ShowCurrentBoothsStatus()
       {

           try
           {
               DbCommand dbcmd =
db.GetSqlStringCommand("ShowCurrentBoothsStatus");
               dbcmd.CommandType = CommandType.StoredProcedure;
               DataSet ds = db.ExecuteDataSet(dbcmd);
               return ds.Tables[0];
           }
           catch (Exception ex)
           {
               return null;
           }
       }
       private DataTable GetDataTableForGrid(DataTable dtable)
       {
           try
           {
               DataTable tab1 = new DataTable("AllocatedBooths");
               DataColumn dc = new DataColumn();

               dc = new DataColumn("Name",
System.Type.GetType("System.String"));
               tab1.Columns.Add(dc);
               dc = new DataColumn("NIC",
System.Type.GetType("System.String"));
               tab1.Columns.Add(dc);
               dc = new DataColumn("Father/Husband Name",
System.Type.GetType("System.String"));
               tab1.Columns.Add(dc);
               dc = new DataColumn("Booth Id",
System.Type.GetType("System.String"));
               tab1.Columns.Add(dc);
```

```
                dc = new DataColumn("Description",
System.Type.GetType("System.String"));
                tab1.Columns.Add(dc);
                dc = new DataColumn("Booth Allocation Time",
System.Type.GetType("System.String"));
                tab1.Columns.Add(dc);
                dc = new DataColumn("Status",
System.Type.GetType("System.String"));
                tab1.Columns.Add(dc);
                DataRow tabRow;
                if (dtable != null)
                {
                    foreach (DataRow drow in dtable.Rows)
                    {
                        tabRow = tab1.NewRow();
                        tabRow[0] = drow[0];
                        tabRow[1] = drow[1];
                        tabRow[2] = drow[2];
                        tabRow[3] = drow[3];
                        tabRow[4] = drow[4];
                        tabRow[5] = drow[5];
                        tabRow[6] = drow[6];
                        tab1.Rows.Add(tabRow);
                    }
                }

                return tab1;
            }
            catch (Exception ex)
            {
                return null;
            }
        }
    protected void Timer1_Tick(object sender, EventArgs e)
    {
        GridView1.DataBind();
    }
}
```

CHAPTER 5

TESTING

5.1 Introduction

Testing is an extremely important part of the product development life cycle of any software or project. By identifying the defects and problems during the software testing process,, the end product is indirectly improved. Although the software may not be 100% error free, it is cheaper to avoid problems, rather than fixing the problems after deployment. Therefore, planning for testing should start at the early stages in the requirements gathering. It should be refined and used continuously as the development proceeds.

5.2 Goals

Software testing is a critical software process. It has the intention to find errors in the execution of a system. It has a finite set of test cases in controlled condition, The control condition include normal and abnormal situations.

We identify the expected behaviors. Under the normal conditions, we ask “What will happen to the software if we enter correct input”

Under the abnormal conditions, we ask “what will happen to the software when we enter invalid input.”

Testing should intentionally make things go wrong. It determinates if things happen when they shouldn’t happen when the should

The purpose of testing is to check whether the newly developed system is performing the entire required task without any kind of errors. Basically, evaluation is measured through performance enhancement.

For testing the system that it performed all the business processes, we need to create the test inventory: the following table contains some of the most important test cases and their results.

5.3 Testing Approach

The strategies taken on for the testing of any given software are different at various testing levels. Testing levels are used in reference to the scale on which the software test is being implemented. Normally the testing start from a small portion of the

project or software and than moves on to gradually to its larger parts. The following are the tests used at different levels of software testing.

5.3.1 Unit Testing

In unit testing each method is tested according to its behavior in response to the requirements placed before it. In our developed product each method of the class/object has been tested thoroughly according to the requirements. Debuggers, testing by the client and the development team were used to verify the requirements.

5.3.2 Integration Testing

In integration testing, the interaction and collaboration between the various classes that combine to the form software package is pit to test. In this system, the interaction of the different modules both at the client/interface and the server end has been tested. The behavior of the modules under the integration tests was found according to the requirements they were expected to fulfill.

5.3.3 Validation Testing

In validation testing, the complete system is tested as unit. The system is expected to all possible types of input and under the various scenarios the we expected to encounter. In case of this system the tests were conducted using a wide range of inputs. Its response to all the inputs that it was confronted with was more than satisfactory.

5.3.4 System Testing

Software system testing is often equated with findings bugs. The goal of the system testing is to find discrepancies between the actual behaviors of the implemented system and the desired behavior as described in the system specification. The system testing can be performed at different levels. At the top level, module testing, integration tests, system tests and acceptance or user test. For proposed system, we have done the following types of testing.

- Black Box Testing
- White Box Testing

5.4 Test Case Specification (TCS)

Test Case ID	Action	Expected Result	P/F
T1	On login form user provided his/her login password and press OK button	If the information is correct then the main form is displayed else invalid login/password message appears.	P
T2	User will select and click on a menu item and the respective form is initialized and loaded.	The form is initialized and loaded respectively.	P
T3	CEC press show voters for approve or reject	It shows the list of all Voters	P
T4	CEC sent voter list to DRO	Shows DRO or respective District & confirmation message	P
T5	ACEC prepares voter & enters NIC #	If NIC format is not correct it will displays message	P
T6	ACEC prepares voter & enters details	If name, father name, family number & NIC is not entered (red *) will be displayed to	P

		proceed	
T7	ACEC prepares voter & renter NIC again	If the NIC # is entered again it will displays message	P
T8	ACEC Updates voter List	When Edit voter infor NIC# field will be disabled	P
T9	ACEC voter using NADRA webservice	If NIC # is valid or invalid or invalid format entered, message appears.	P
T10	DRO approve polling station & staff list after entering details	It shows polling station details & Approve button	P
T11	DRO send voter list to RO	When enter town/tehsil it automatically shows the relevant RO with confirmation or rejection message	P
T12	RO prepares Candidate details add, edit or delete	If no name entered message appears	P
T13	RO send candidate list to PO	On success & failure message appears	P
T14	PO allocates booth to voter	It will check NIC # if correct or wrong message appers	P
T15	PO check booth status	If available or not message appears	P

T16	Booth welcome screen	Continues if PO assign booth to respective voter after checking the NIC#	P
T17	Booth welcome screen	If not yet assigned message appears	P
T18	Vote Confirmation	After selection of all candidates confirmation, rejection & eligibility messages appears	P

5.5 Evaluation

A software system is evaluated by the user interface. Some of the factors which are considered in system evaluation are speed of performance, ease of use and user satisfaction. Every developer try to succeed in all categories but there is a trade off.

5.6 Merits & Demerits

- **Easily Customizable**

  The developed system is easily customizable i.e. in future any improvement relating to input/output design, can easily be introduced. This is due to the SQL server 2005 DBMS utilities.

- **Efficiency**

  The new computerized system has been design in such manner that the users feel no difficulty in entering the data and generating the desired outputs.

- **Accuracy**

  The outputs produced by the system are accurate, which is made possible by providing validation checks at the data entry time.

- **Duplication of Processes**

  As in the present system, similar types of activities are carry out at different places, with the adaptation of computerization system, this can be done centrally, which will facilitate in avoiding the duplication of processing and enforcing a better systematic control over activities.

- **Minimum Storage Requirements**

  Computerization system required less storage space, there is no needs of maintaining big registers more over the required information can be accessed faster compared to manual system.

- **Security**

  First of all user name and password is required to logon into the system. Further more database username & password is also required to access the system. This is done o provide maximum security to the system

- **Easy to use**

  The system is menu driven and user friendly. A simple user, even if without much computer knowledge, can use the system without any difficulties.

- **Speed of Performance**

  The main task performed by the system i.e. date entry and retrieval, that are carried out swiftly. During design and development phases every possible effort was made to overcome the deficiencies in the system but despite of this effort there may be roams for improvement.

- **Modularity**

  The system is dividing into number of modules integrated to fulfill user requirements. These modules are independent of each other. Another advantage of modularity is ease in modification.

CHAPTER 6

CONCLUSION

6.1 Conclusions

Systems are made after the detail & complete Project life cycle, theoretical and practice is different learning aspects. Theory provides a world to its ideal state but the development of the real environment is very difficult. Real working environment is achieved only by the interaction with the end user & their requirements.

Group of peoples who ultimately define the fate on any software, weather it is flashy software for a giant company or a study and small project from a computer graduate or student. It is the end user that has the real power to decide about the software.

Another main thing is the significance of the analysis of the system to be developed. It has always been a tendency to consider the analysis as a non-productive activity, but any mistake made here is replicated many times over the next phase, ending in the system to be re-analyze

At the end I just want to say that this product is very flexible, user friendly and easy to understand for a simple user, which will be the end user of this product ultimately. This system has been tested by using different testing strategies e.g. unit testing, system testing etc.

The following are some of the conclusions that could be made after development and implementation of this system.

- Effective, Efficient and Reliable
- Enhanced Manageability & Availability
- Timely and accurate decision Support
- Professional Business Practice
- To get rid of Paper work
- User Friendly Interface & easily understandable
- Maintainability & Verifiability.

APPENDIX A

OES Database Schema

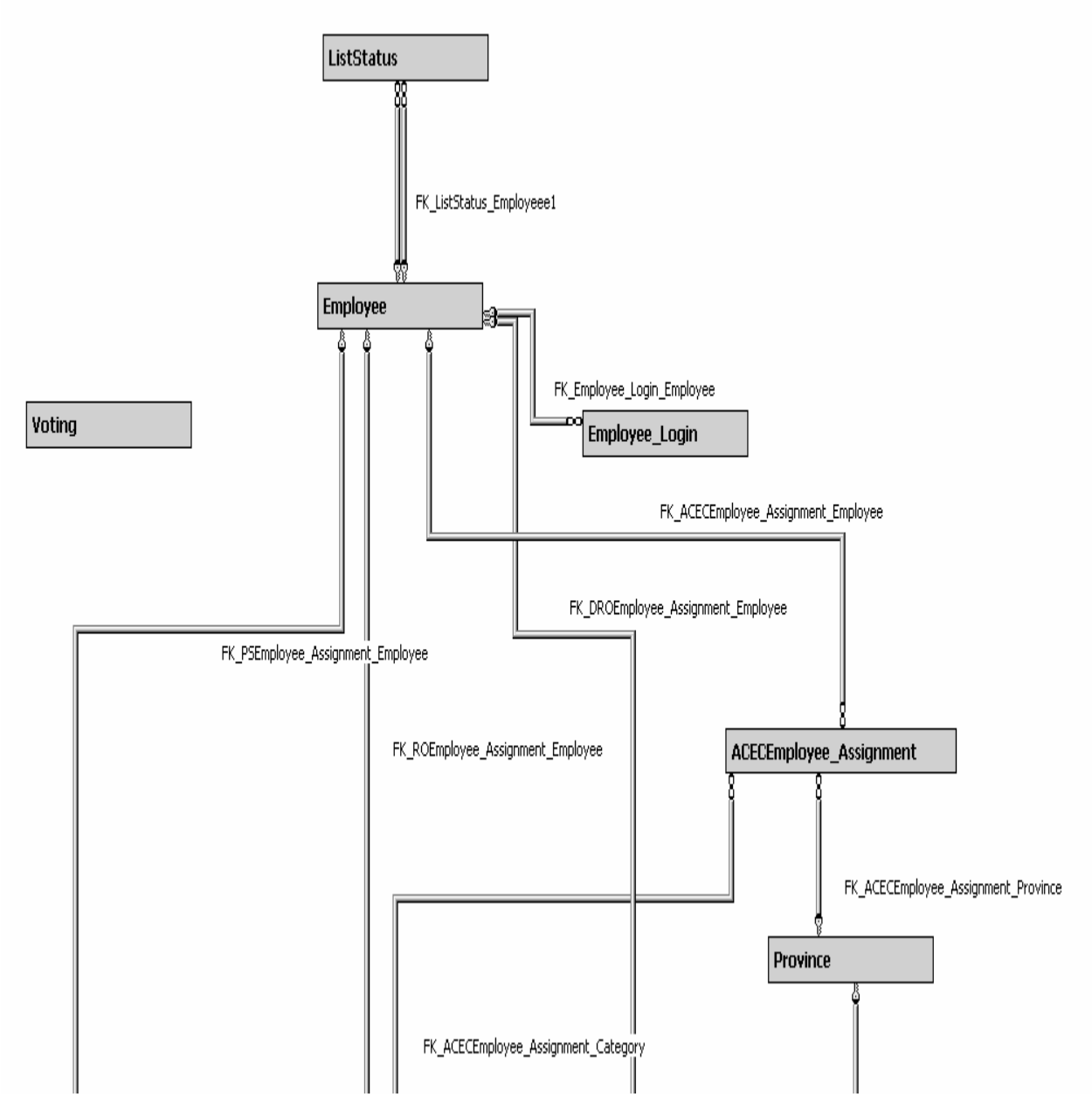

ListStatus
FK_ListStatus_Employeee1
Employee
FK_Employee_Login_Employee
Voting
Employee_Login
FK_ACECEmployee_Assignment_Employee
FK_DROEmployee_Assignment_Employee
FK_PSEmployee_Assignment_Employee
FK_ROEmployee_Assignment_Employee
ACECEmployee_Assignment
FK_ACECEmployee_Assignment_Province
Province
FK_ACECEmployee_Assignment_Category

Figure 3.12 (a)

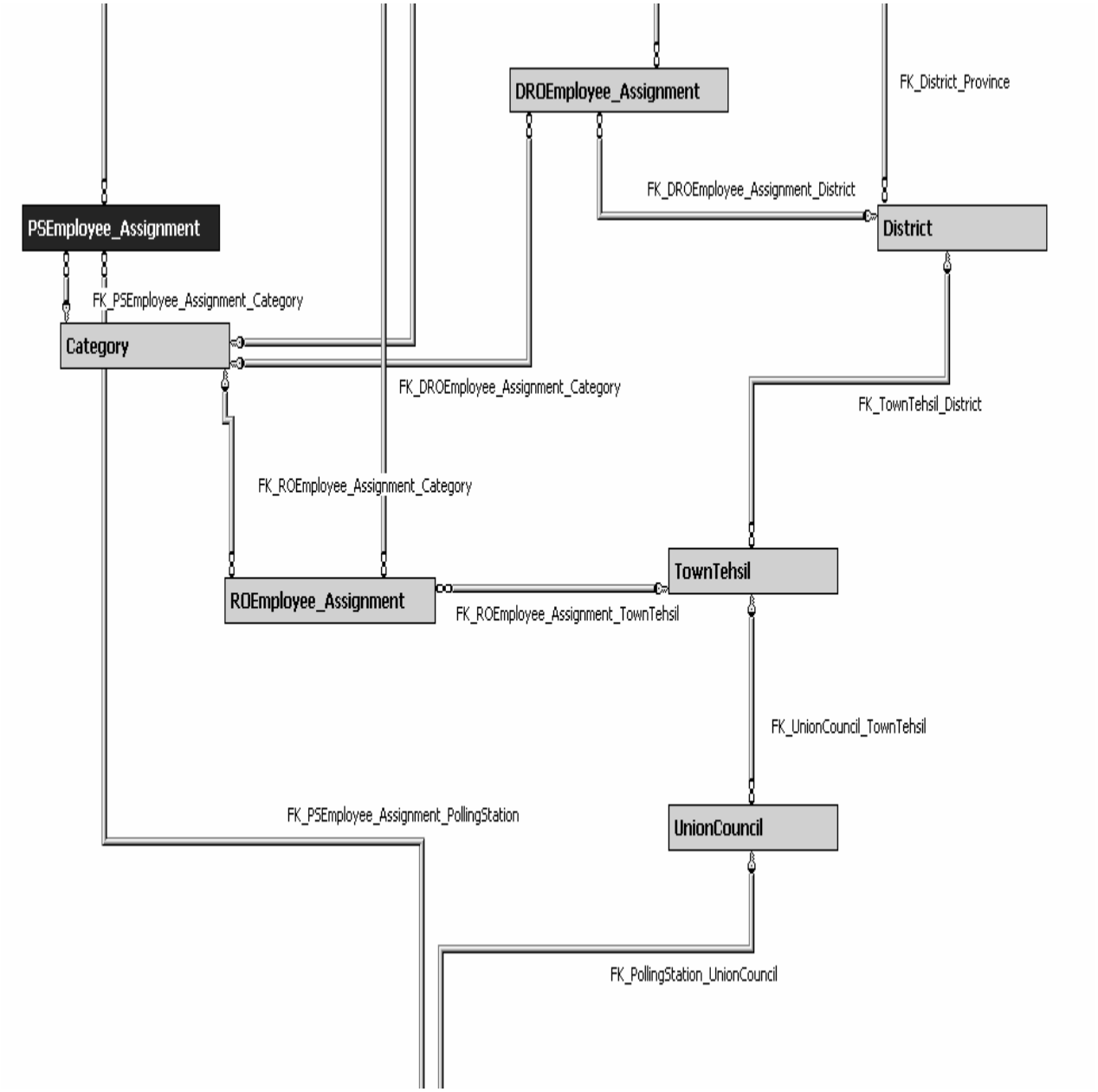

DROEmployee_Assignment
FK_District_Province
FK_DROEmployee_Assignment_District
PSEmployee_Assignment
District
FK_PSEmployee_Assignment_Category
Category
FK_DROEmployee_Assignment_Category
FK_TownTehsil_District
FK_ROEmployee_Assignment_Category
TownTehsil
ROEmployee_Assignment
FK_ROEmployee_Assignment_TownTehsil
FK_UnionCouncil_TownTehsil
FK_PSEmployee_Assignment_PollingStation
UnionCouncil
FK_PollingStation_UnionCouncil

Figure 3.12 (b)

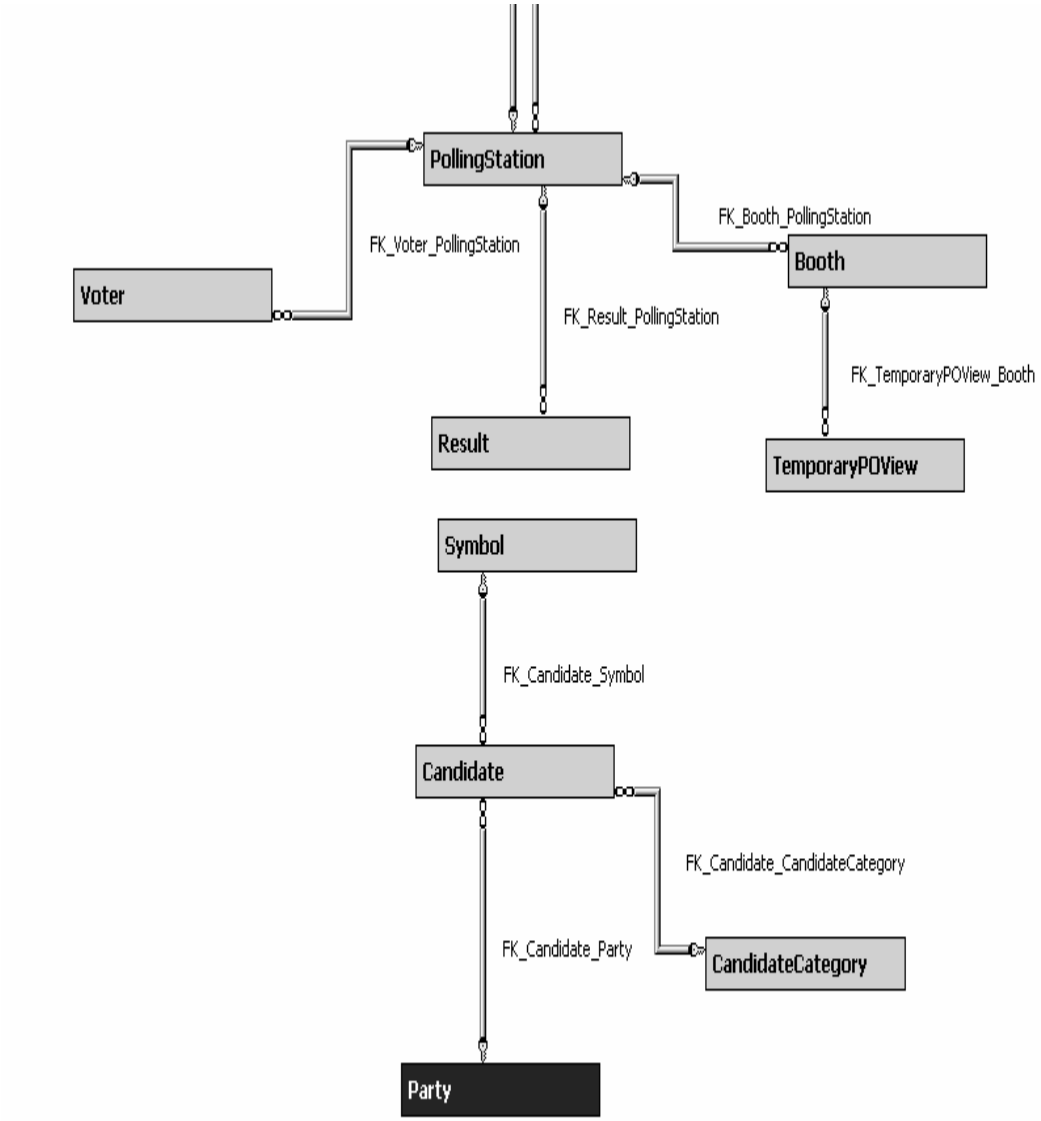
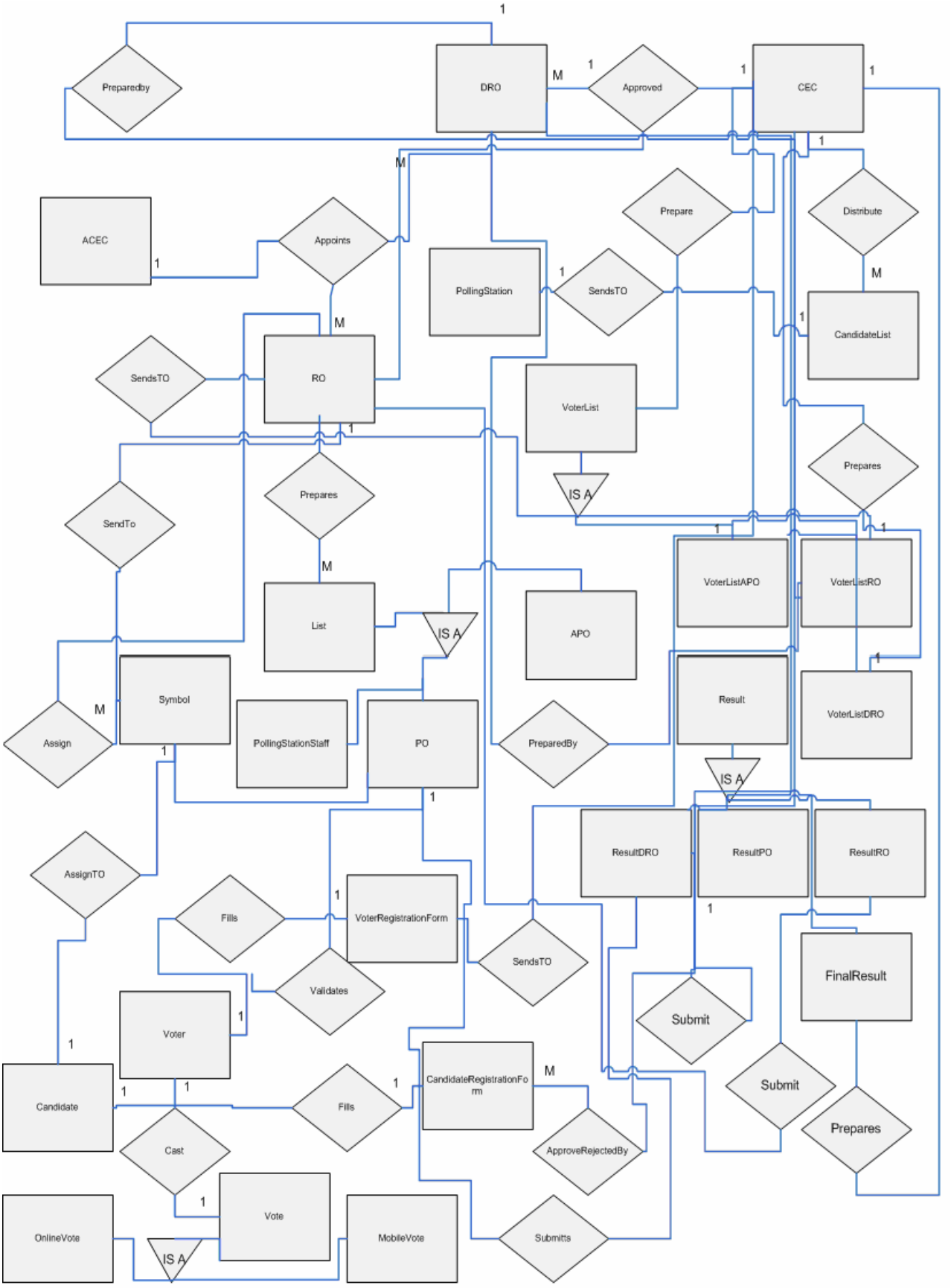

PollingStation
FK_Booth_PollingStation
FK_Voter_PollingStation
Booth
Voter
FK_Result_PollingStation
FK_TemporaryPOView_Booth
Result
TemporaryPOView
Symbol
FK_Candidate_Symbol
Candidate
FK_Candidate_CandidateCategory
FK_Candidate_Party
CandidateCategory
Party

Figure 3.12 (c)

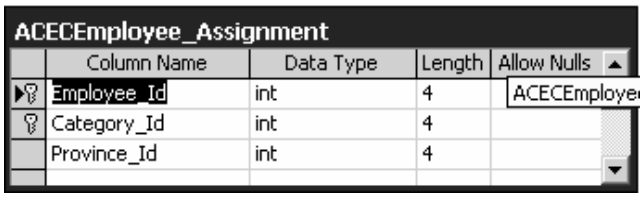
APPENDIX B

DATA MODEL


Preparedby
DRO
Approved
CEC
ACEC
Appoints
Prepare
Distribute
PollingStation
SendsTO
CandidateList
SendsTO
RO
VoterList
Prepares
SendTo
Prepares
IS A
VoterListAPO
VoterListRO
List
IS A
APO
Symbol
Result
VoterListDRO
Assign
PollingStationStaff
PO
PreparedBy
IS A
ResultDRO
ResultPO
ResultRO
AssignTO
Fills
VoterRegistrationForm
SendsTO
FinalResult
Validates
Submit
Voter
Submit
Candidate
Fills
CandidateRegistrationForm
Prepares
Cast
ApproveRejectedBy
Vote
OnlineVote
MobileVote
Submits
IS A

APPENDIX C

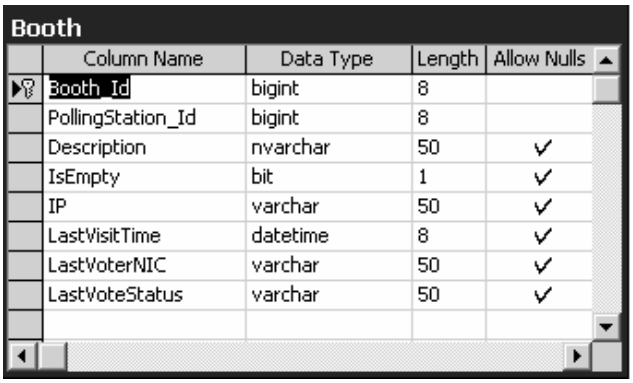
DATABASE TABLES



ACECEmployee_Assignment

	Column Name	Data Type	Length	Allow Nulls
🔑	Employee_Id	int	4	
🔑	Category_Id	int	4	
	Province_Id	int	4	

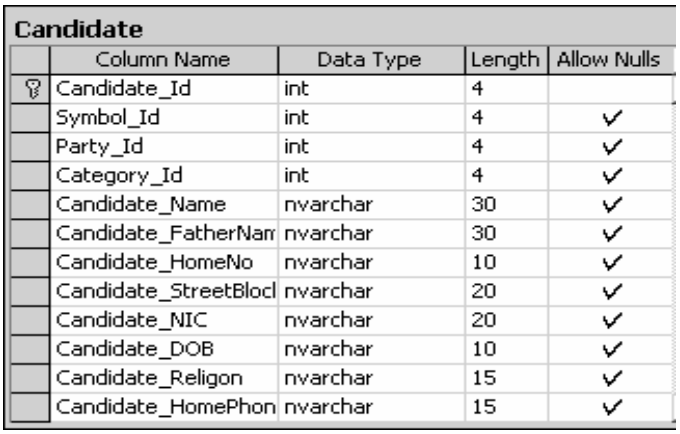
Figure DT-1



Booth

	Column Name	Data Type	Length	Allow Nulls
🔑	Booth_Id	bigint	8	
	PollingStation_Id	bigint	8	
	Description	nvarchar	50	✓
	IsEmpty	bit	1	✓
	IP	varchar	50	✓
	LastVisitTime	datetime	8	✓
	LastVoterNIC	varchar	50	✓
	LastVoteStatus	varchar	50	✓

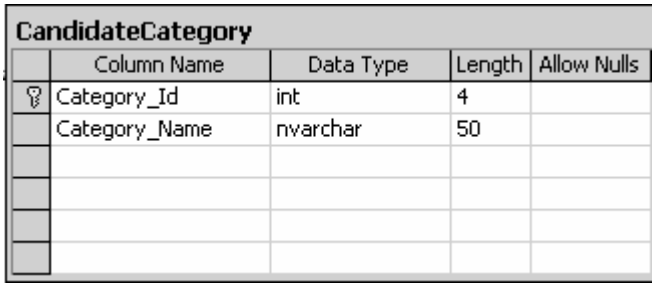
Figure DT-2



Candidate

	Column Name	Data Type	Length	Allow Nulls
🔑	Candidate_Id	int	4	
	Symbol_Id	int	4	✓
	Party_Id	int	4	✓
	Category_Id	int	4	✓
	Candidate_Name	nvarchar	30	✓
	Candidate_FatherNam	nvarchar	30	✓
	Candidate_HomeNo	nvarchar	10	✓
	Candidate_StreetBlocl	nvarchar	20	✓
	Candidate_NIC	nvarchar	20	✓
	Candidate_DOB	nvarchar	10	✓
	Candidate_Religon	nvarchar	15	✓
	Candidate_HomePhon	nvarchar	15	✓

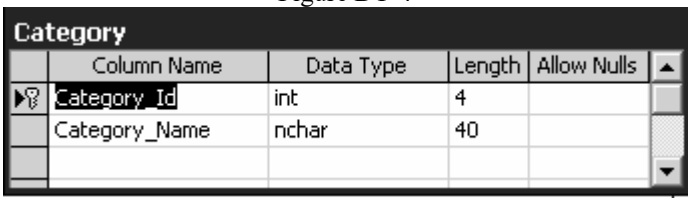
Figure DT-3



CandidateCategory

	Column Name	Data Type	Length	Allow Nulls
🔑	Category_Id	int	4	
	Category_Name	nvarchar	50	

Figure DT-4



Category

	Column Name	Data Type	Length	Allow Nulls
🔑	Category_Id	int	4	
	Category_Name	nchar	40	

Figure DT-5

District

Column Name	Data Type	Length	Allow Nulls
District_Id	int	4	
District_Name	text	16	
Province_id	int	4	
Allocated	int	4	✓

Figure DT-6

DROEmployee_Assignment

Column Name	Data Type	Length	Allow Nulls
Employee_Id	int	4	
Category_Id	int	4	
District_Id	int	4	
Id	int	4	
Approved	bit	1	✓
ACEC_Id	bigint	8	✓

Figure DT-7

Employee

Column Name	Data Type	Length	Allow Nulls
Employee_Id	int	4	
Employee_FirstName	nvarchar	50	
Employee_LastName	nvarchar	50	
Employee_FatherNam	nvarchar	50	
Employee_NIC_No	nvarchar	50	
Employee_EmailID	nvarchar	30	✓
Employee_Fax_No	nvarchar	20	✓
Employee_HomePhone	nvarchar	20	✓
Employee_OfficePhon	nvarchar	20	✓
Employee_MobileNo	nvarchar	20	✓
UserName	varchar	50	✓
Password	varchar	50	✓
Position_Allocated	bit	1	✓

Figure DT-8

Employee_Login

Column Name	Data Type	Length	Allow Nulls
UserName	varchar	50	
Password	nvarchar	50	
Employee_Id	int	4	✓

Figure DT-9

ListStatus

Column Name	Data Type	Length	Allow Nulls
ListType	nvarchar	50	✓
SubmittedByEmpID	int	4	✓
SubmittedToEmpID	int	4	✓
SubmittedDate	datetime	8	✓
Status	varchar	50	✓

Figure DT-10

NadraData

Column Name	Data Type	Length	Allow Nulls
NIC	varchar	50	
Name	varchar	50	

Figure DT-11

Party

Column Name	Data Type	Length	Allow Nulls
Party_Id	int	4	
Party_Name	nchar	50	
Party_Desc	nchar	50	✓

Party

Figure DT-12

PollingStation

Column Name	Data Type	Length	Allow Nulls
PollingStation_Id	bigint	8	
PollingStation_Name	nchar	50	✓
PollingStation_Addres:	nchar	50	✓
PollingStation_MaleBo	int	4	✓
PollingStation_Femalel	int	4	✓
UnionCouncil_Id	bigint	8	
Approved	bit	1	✓
DRO_Id	bigint	8	✓

Figure DT-13

Province

Column Name	Data Type	Length	Allow Nulls
Province_Id	int	4	
Province_Name	text	16	

Figure DT-14

PSEmployee_Assignment

Column Name	Data Type	Length	Allow Nulls
Id	int	4	
Employee_Id	int	4	
Category_Id	int	4	
PollingStation_Id	bigint	8	
Approved	bit	1	✓
DRO_Id	bigint	8	✓

Figure DT-15

Result

Column Name	Data Type	Length	Allow Nulls
PollingStation_Id	bigint	8	
Candidate_Id	int	4	
Vote_Count	int	4	
LastVote_Count	int	4	✓

Figure DT-16

ROEmployee_Assignment

Column Name	Data Type	Length	Allow Nulls
Employee_Id	int	4	
Category_Id	int	4	
TownTehsil_Id	int	4	
Id	int	4	
Approved	bit	1	✓
CEC_Id	bigint	8	✓

Figure DT-17

ServerResult

Column Name	Data Type	Length	Allow Nulls
PollingStation_Id	bigint	8	
Candidate_Id	int	4	
Vote_Count	int	4	
LastVote_Count	int	4	✓

Figure DT-18

Symbol

Column Name	Data Type	Length	Allow Nulls
Symbol_Id	int	4	
UnionCouncil_Id	int	4	
Symbol_Name	nvarchar	50	
Image_Path	nvarchar	100	
AssignedFlag	bit	1	

Figure DT-19

Temp_BoothAllocation

	Column Name	Data Type	Length	Allow Nulls
	Voter_Id	bigint	8	
	Allocated_Booth_Id	bigint	8	✓
	Allocation_Time	datetime	8	✓
	Finish_Time	datetime	8	✓
	Current_Status	nvarchar	50	✓

Figure DT-20

TemporaryPOView

	Column Name	Data Type	Length	Allow Nulls
🔑	VoterNIC	nvarchar	50	
	VoterName	nvarchar	50	
	Gender	nchar	10	
	EntryTime	datetime	8	
	TotalTime	datetime	8	
	Booth_Id	bigint	8	

Figure DT-21

TownTehsil

	Column Name	Data Type	Length	Allow Nulls
▶🔑	TownTehsil_Id	int	4	
	TownTehsil_Name	text	16	✓
	TownTehsil_Status	int	4	✓
	District_Id	int	4	✓
	Allocated	int	4	✓

Figure DT-22

UnionCouncil

	Column Name	Data Type	Length	Allow Nulls
▶🔑	UnionCouncil_Id	bigint	8	
	Name	nvarchar	50	✓
	Description	nvarchar	50	✓
	TownTehsil_Id	int	4	

Figure DT-23

Voter

	Column Name	Data Type	Length	Allow Nulls
🔑	Voter_Id	int	4	
	Voter_Name	nvarchar	30	✓
	Voter_Father_Husban	nvarchar	30	✓
	Voter_HomeNo	nvarchar	10	✓
	Voter_StreetBlock	nvarchar	20	✓
	Voter_NIC	nvarchar	20	✓
	Voter_DOB	nvarchar	10	✓
	Voter_Religon	nvarchar	15	✓
	Voter_HomePhone	nvarchar	15	✓
	Voter_MobilePhone	nvarchar	15	✓
	Voter_FamilyNo	int	4	✓
	Voter_Gender	nv Voter	10	✓
	Voter_Female_Categc	nvarchar	10	✓
	PollingStation_Id	bigint	8	✓
	DRO_Id	bigint	8	✓
	RO_Id	bigint	8	✓
	PO_Id	bigint	8	✓
	CEC_Id	bigint	8	✓
	Approved	bit	1	✓
	CanVote	int	4	✓

Figure DT-24

Voting

	Column Name	Data Type	Length	Allow Nulls
🔑	Voter_NIC	nvarchar	20	
	Voting_Status	bit	1	
	Vote_Type	bit	1	
	Nazim_Id	int	4	✓
	NaibNazim_Id	int	4	✓
	GeneralCouncellor_Id	int	4	✓
	LabourCouncellor_Id	int	4	✓
	LadyCouncellor_Id	int	4	✓
	MinorityCouncellor_Id	int	4	✓

Figure DT-25

APPENDIX D

USER'S MANUAL

Screen Shots CEC

- CEC Home Page
- Approve, Send Voter List to DRO & View Result

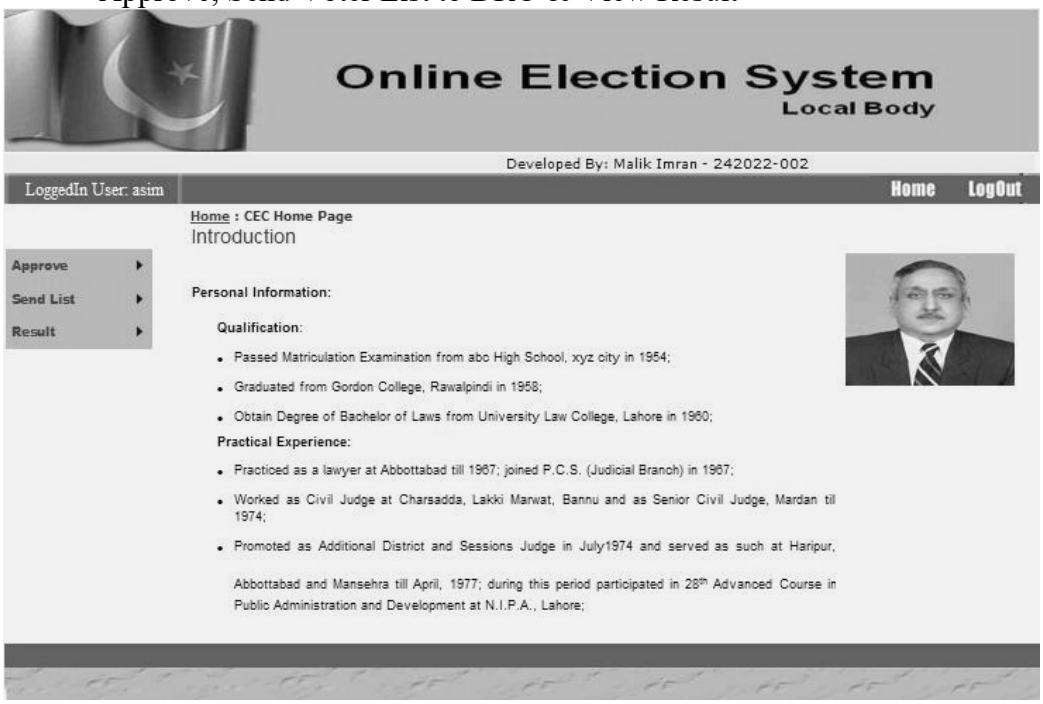


Figure: CEC Home page

- CEC View & Approves DRO list send by ACEC

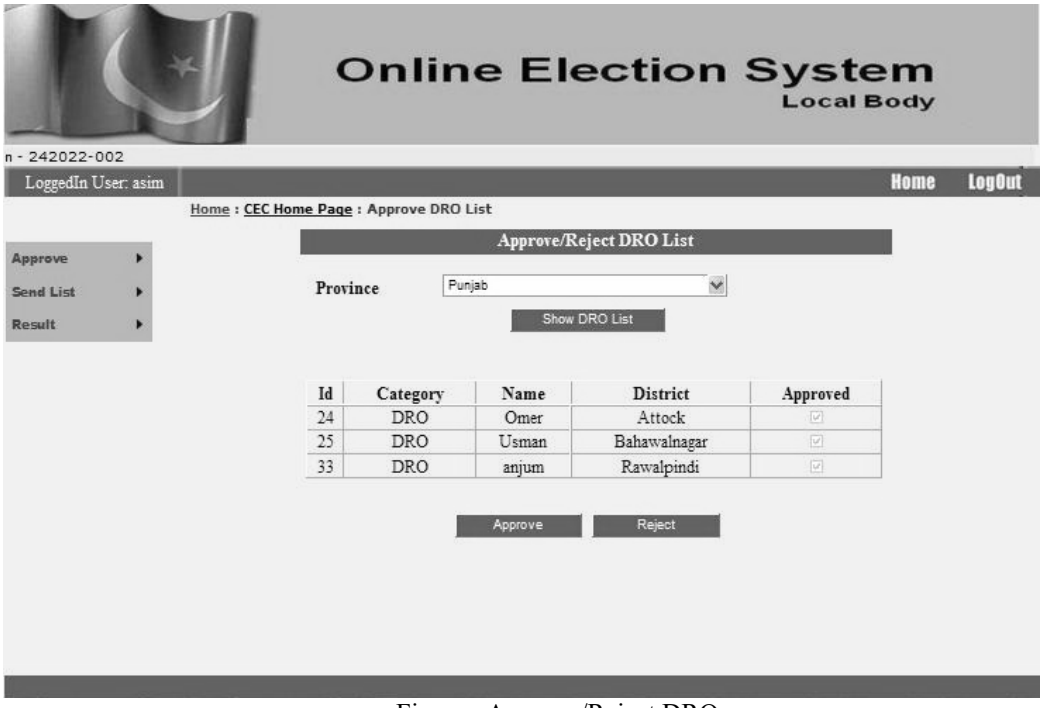


Figure: Approve/Reject DRO

- CEC View & Approves RO list send by ACEC

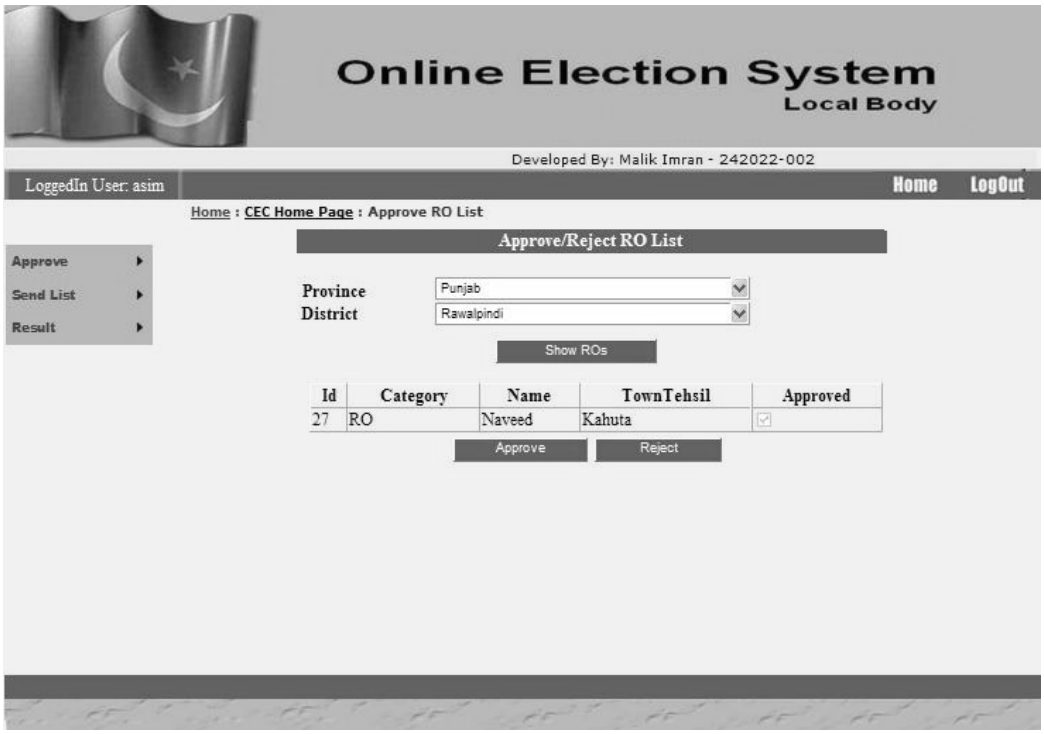


Figure: Approve/Reject RO

- CEC View & Approves VOTER list send by ACEC

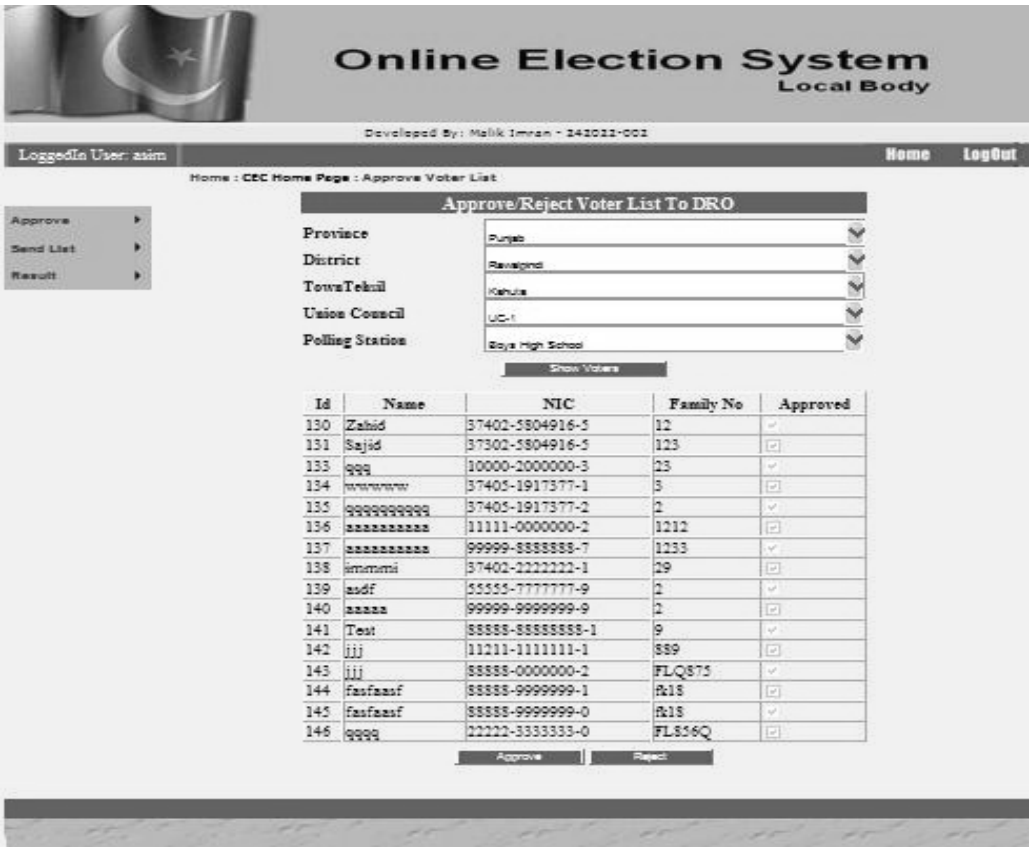


Figure: Approve Reject Voter List to DRO

- CEC send Voter list to DRO of selected province & district

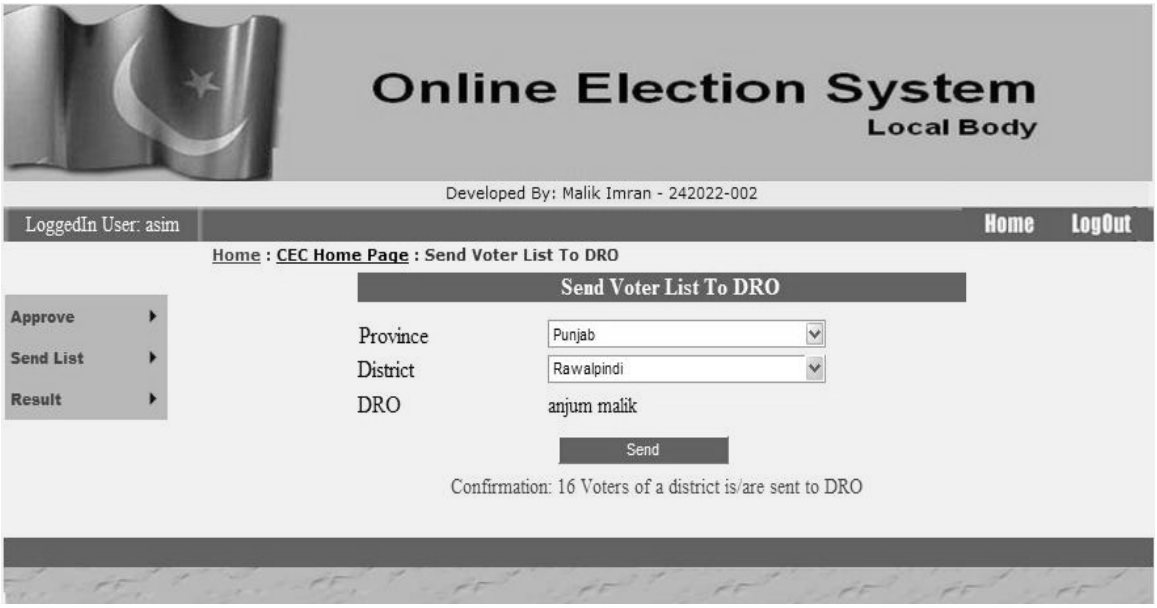


Figure: Send Voter List to DRO

- CEC can see results of all Province, Districts & Tehsils & Union councils

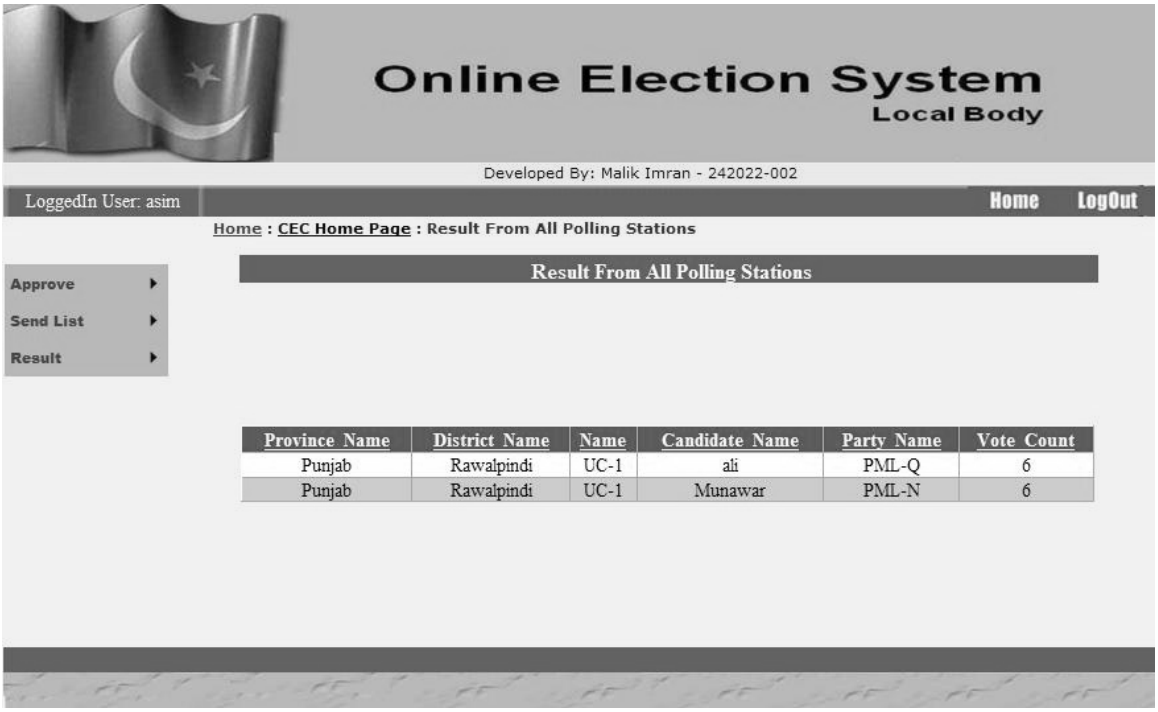


Figure: Result

ACEC

- Prepare voters of the province Verifying eligibility by NADRA web service
- Should enter NIC & Family No in valid format

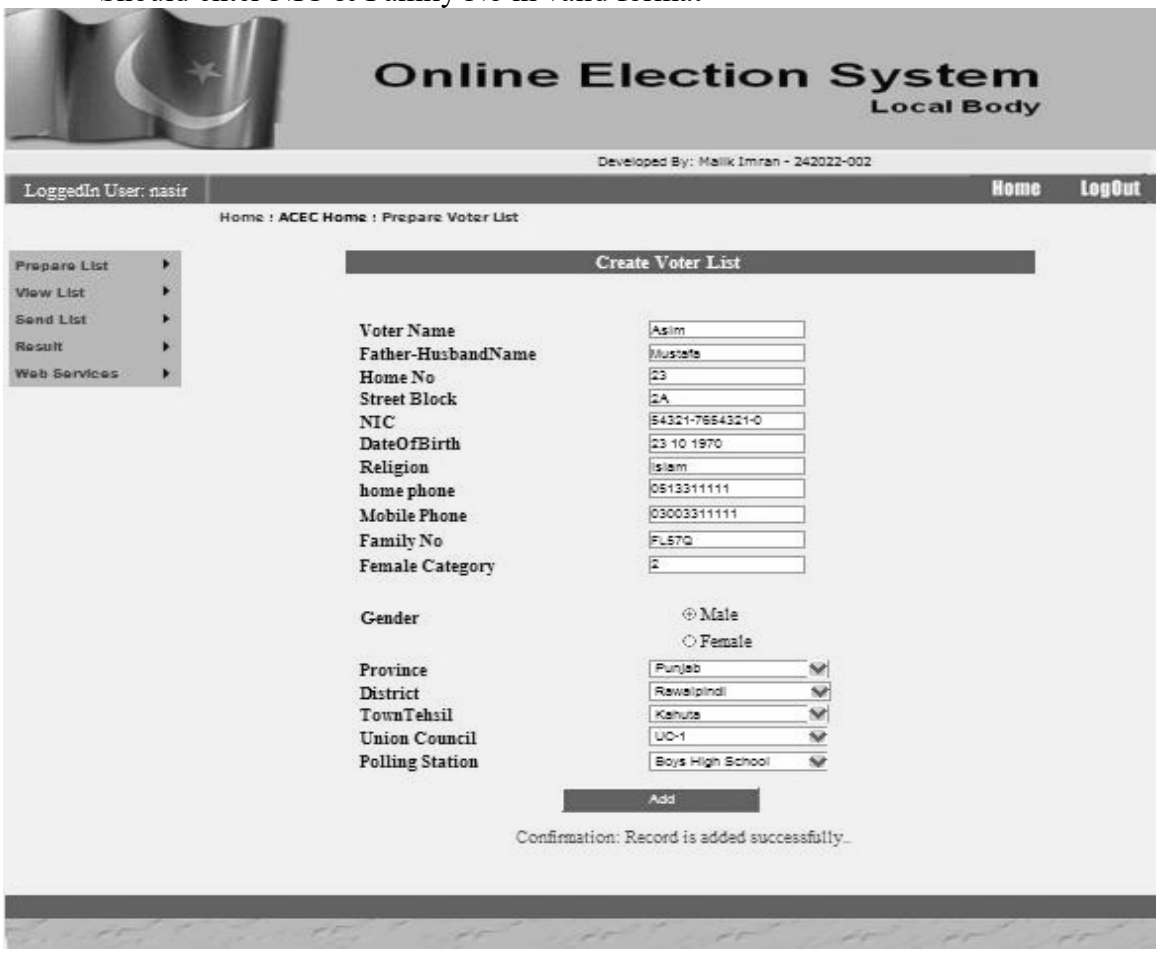


Figure: Prepare Voter

- Already created employees are assigned DRO role by selecting for cretin Province & District

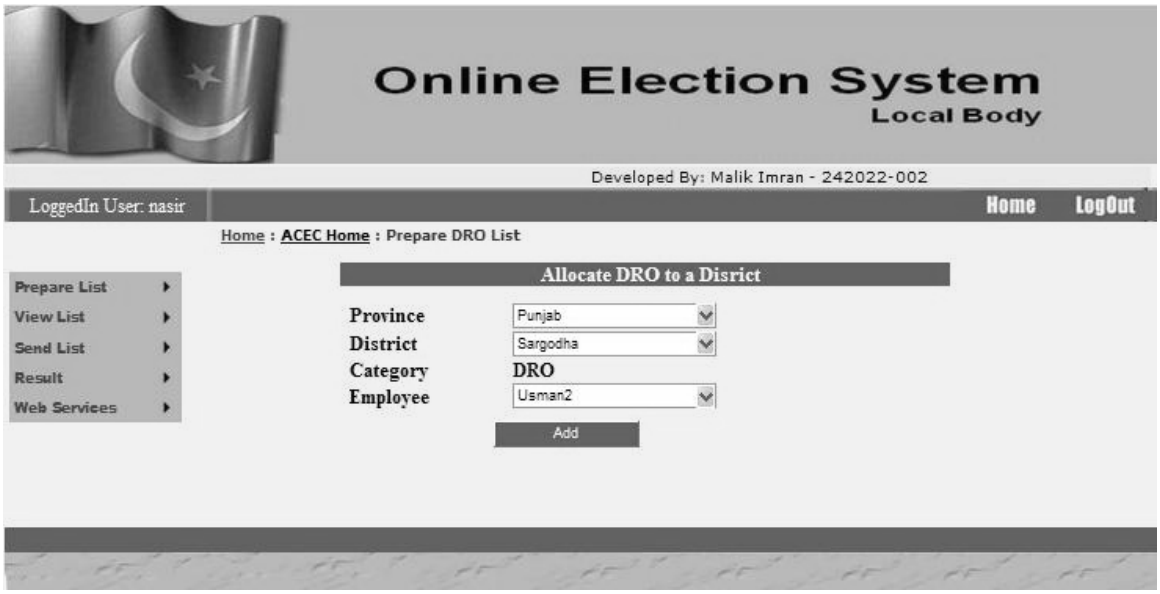


Figure: Prepare DRO

- Already created employees are assigned RO role

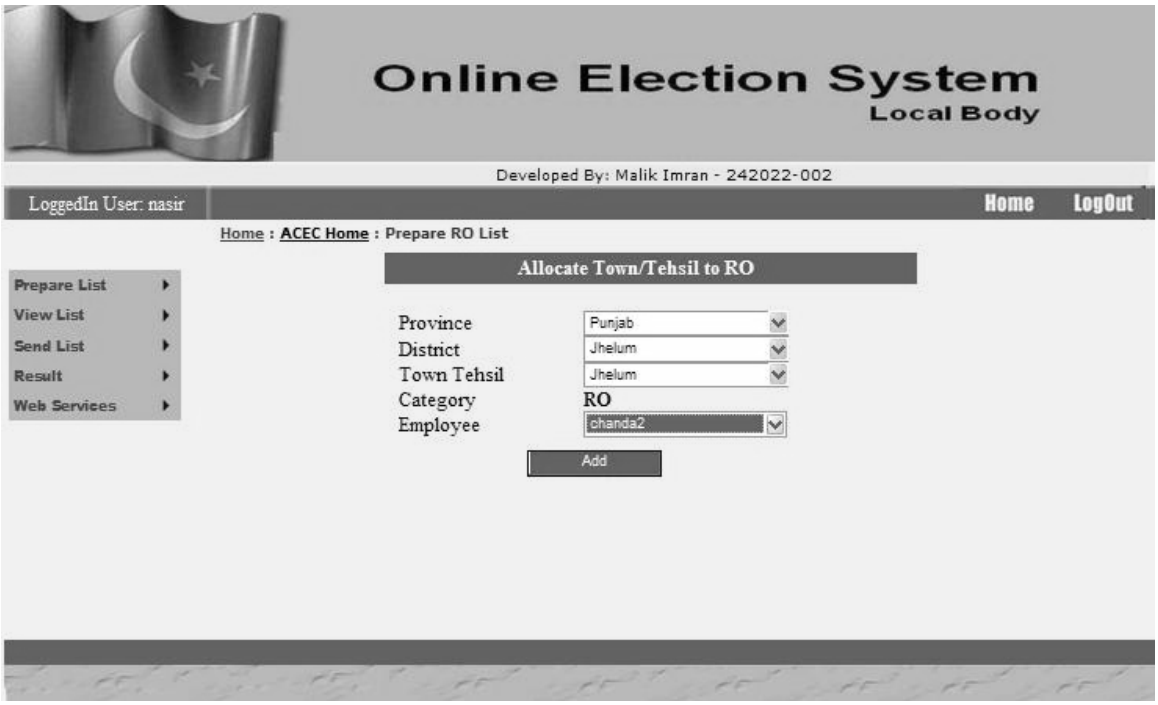


Figure: Prepare RO list

- Create new Employees for DRO & RO roles

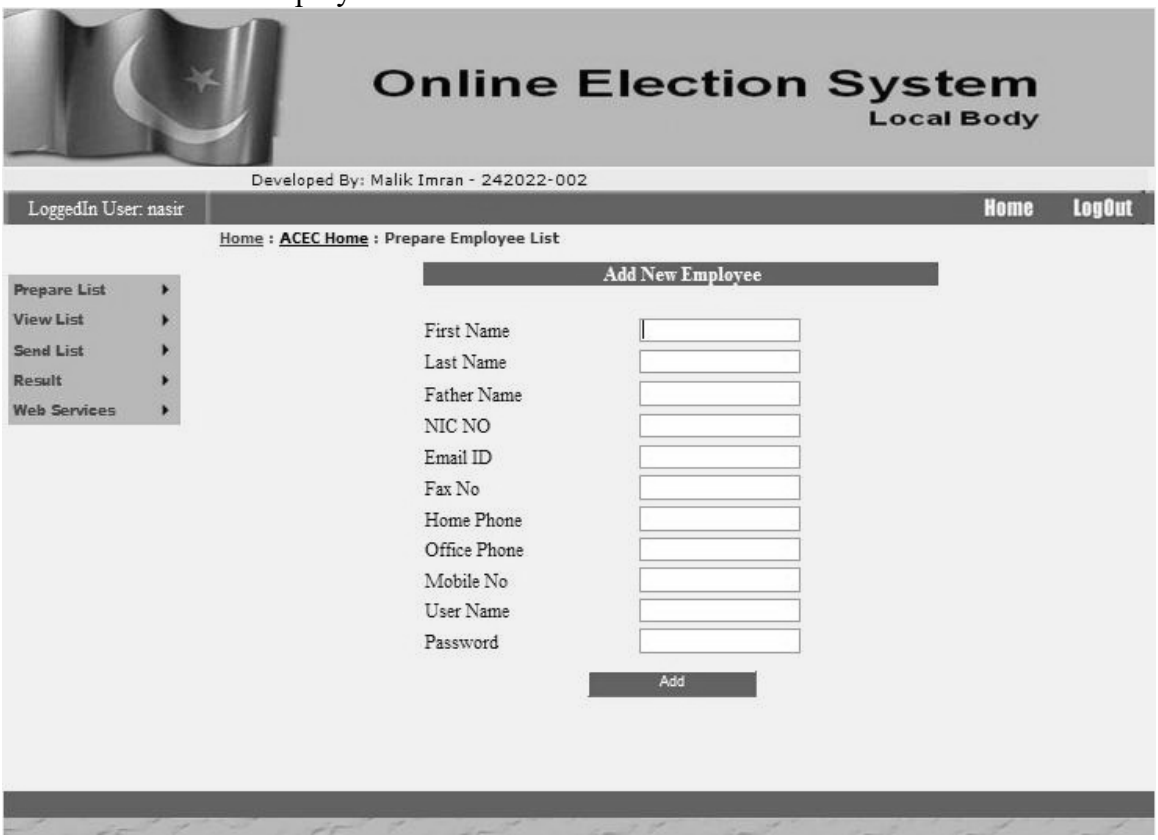


Figure: Prepare Employee

- View List > View Voter list
- Show all voters created for certain district, town tehsil & union council

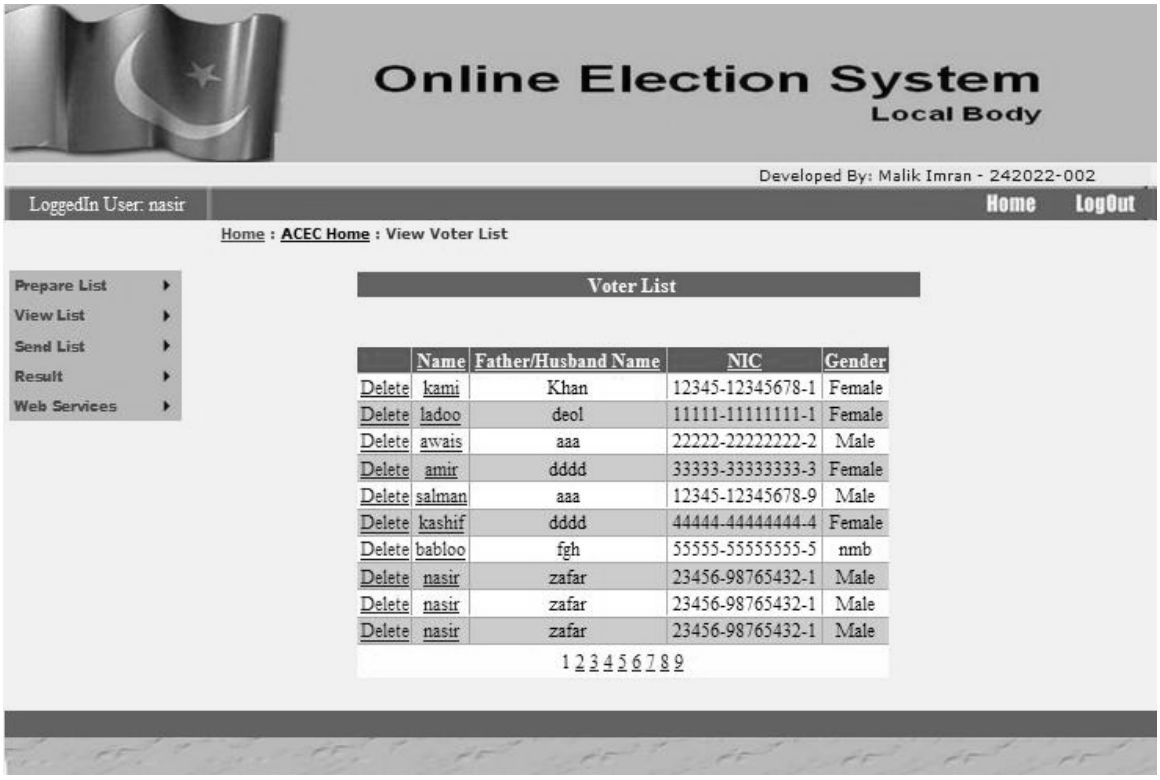


Figure: View voter list

- View List > View DRO list
- Show all DRO's created for certain district.

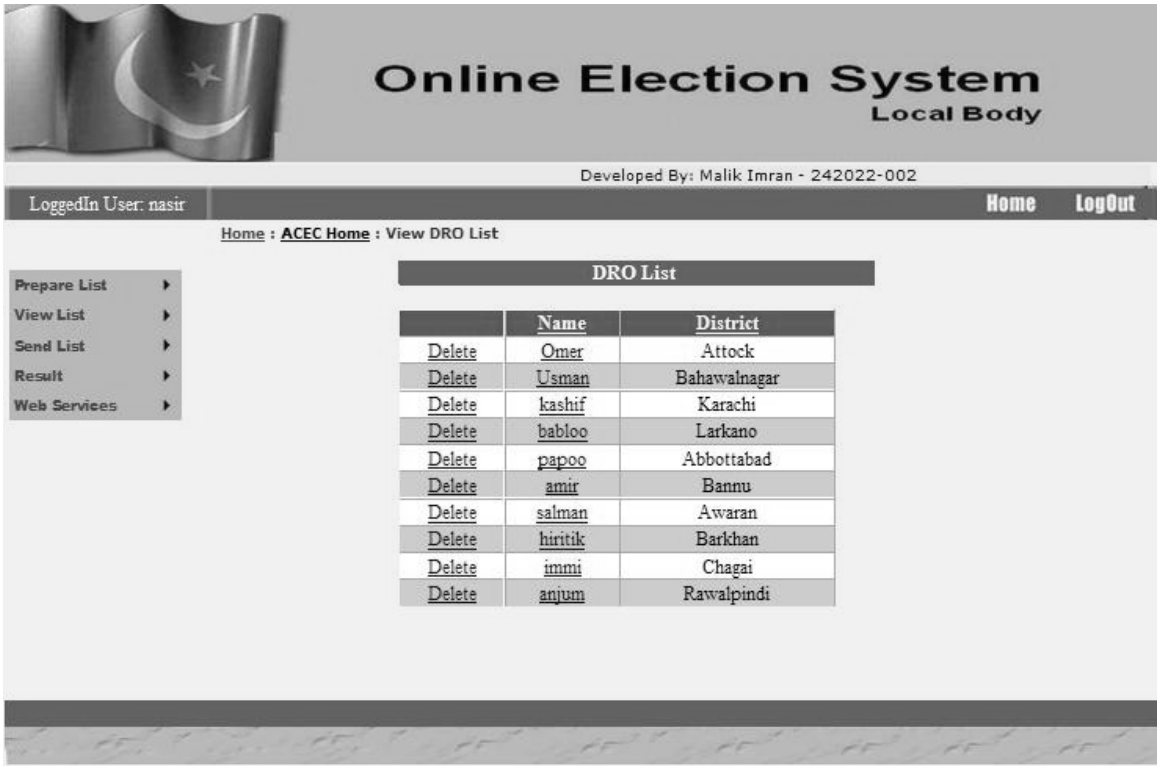


Figure: View DRO list

- View List > View Employee list
- Show all Employees created for certain for all levels starting from CEC, ACEC, DRO, RO & PO
- Can Delete & update users as well

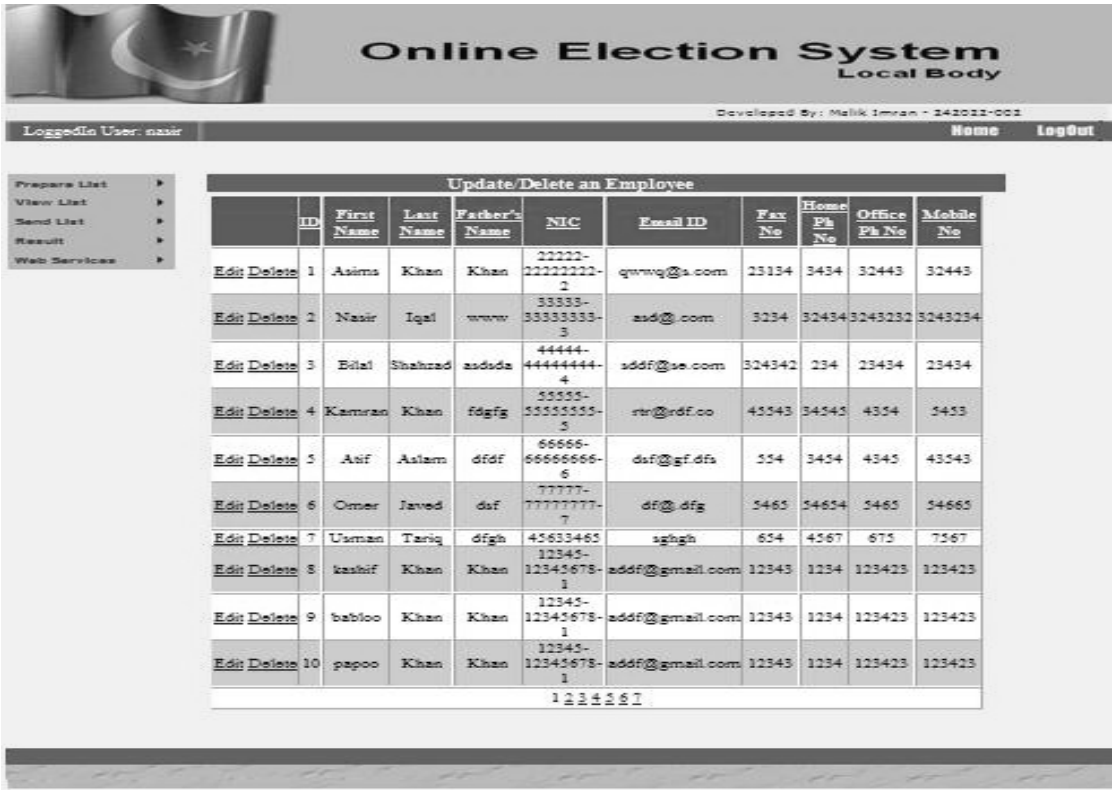


Figure: View Employee List

- After voter list preparation it is send to CEC for approval

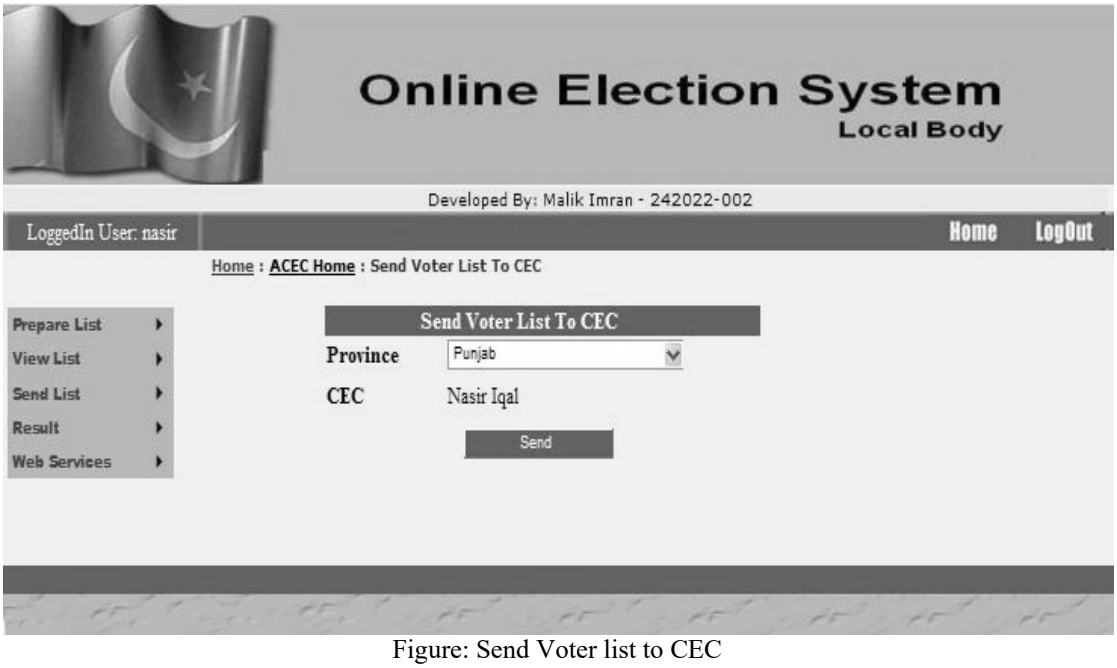


Figure: Send Voter list to CEC

- CEC can see results of all District's & Tehsil's & Union council's of certin province

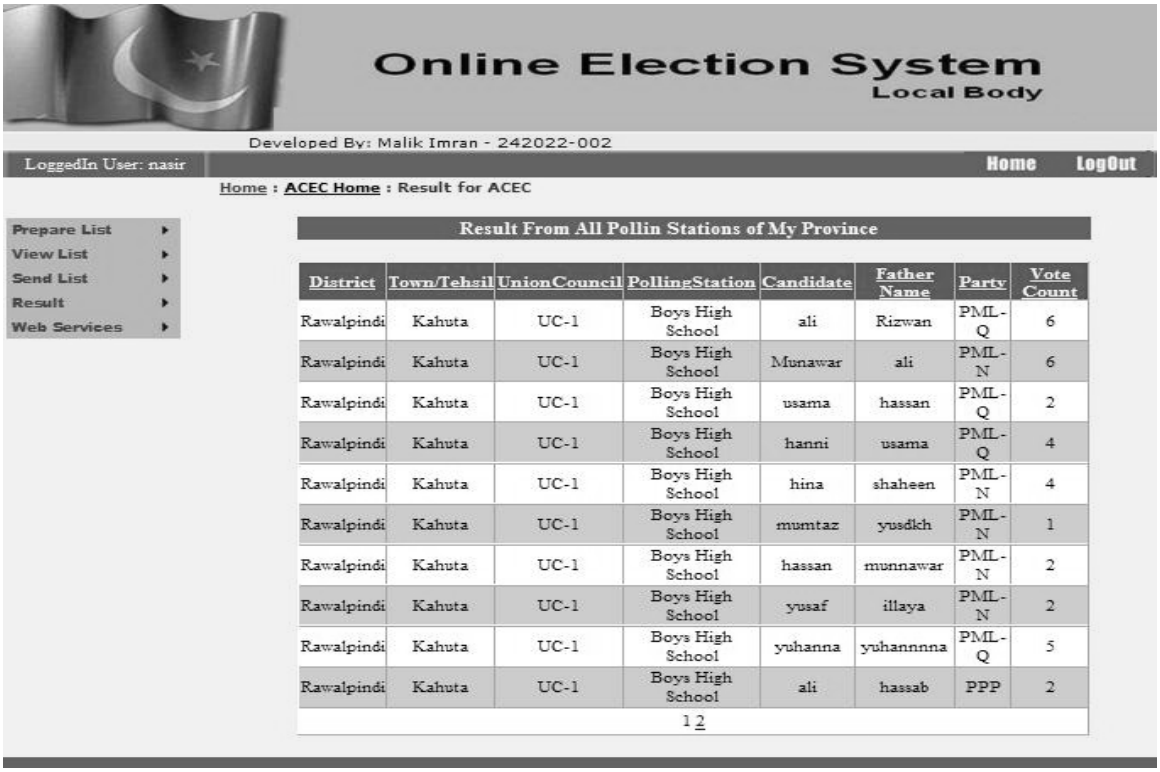


Figure: View Result

- Voter eligibility is checked & further helping for preparing the voter

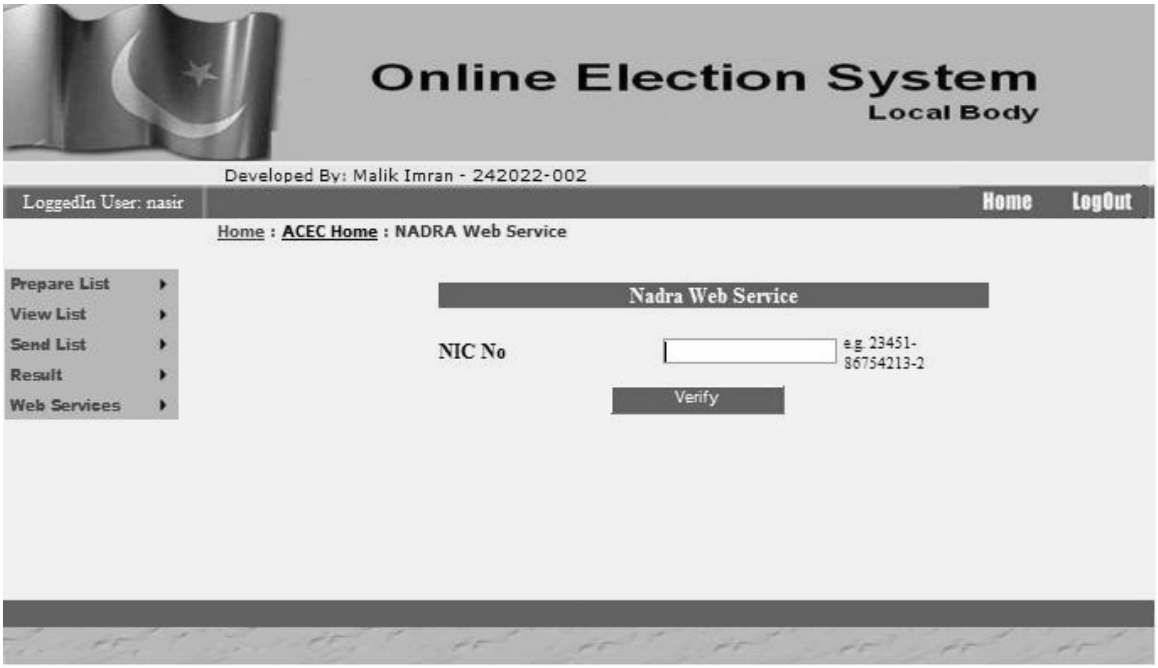


Figure: NADRA webservice

DRO

- Approve Polling station list assigned by RO

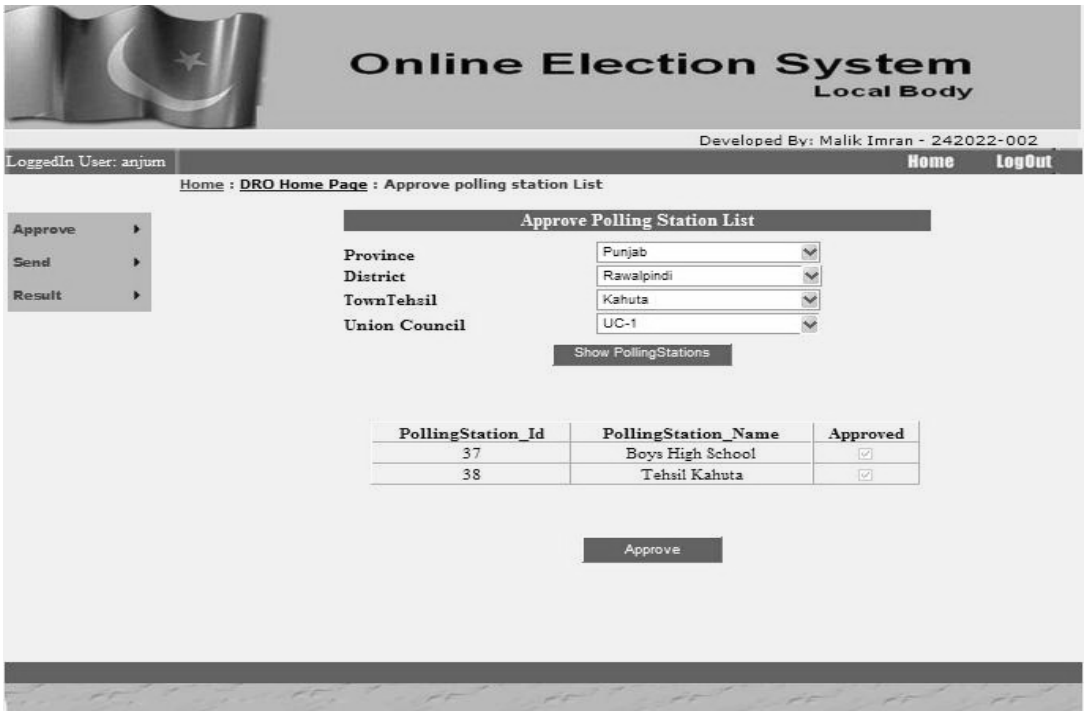


Figure: Approve Polling station list

- Approve Polling station staff list assigned by RO

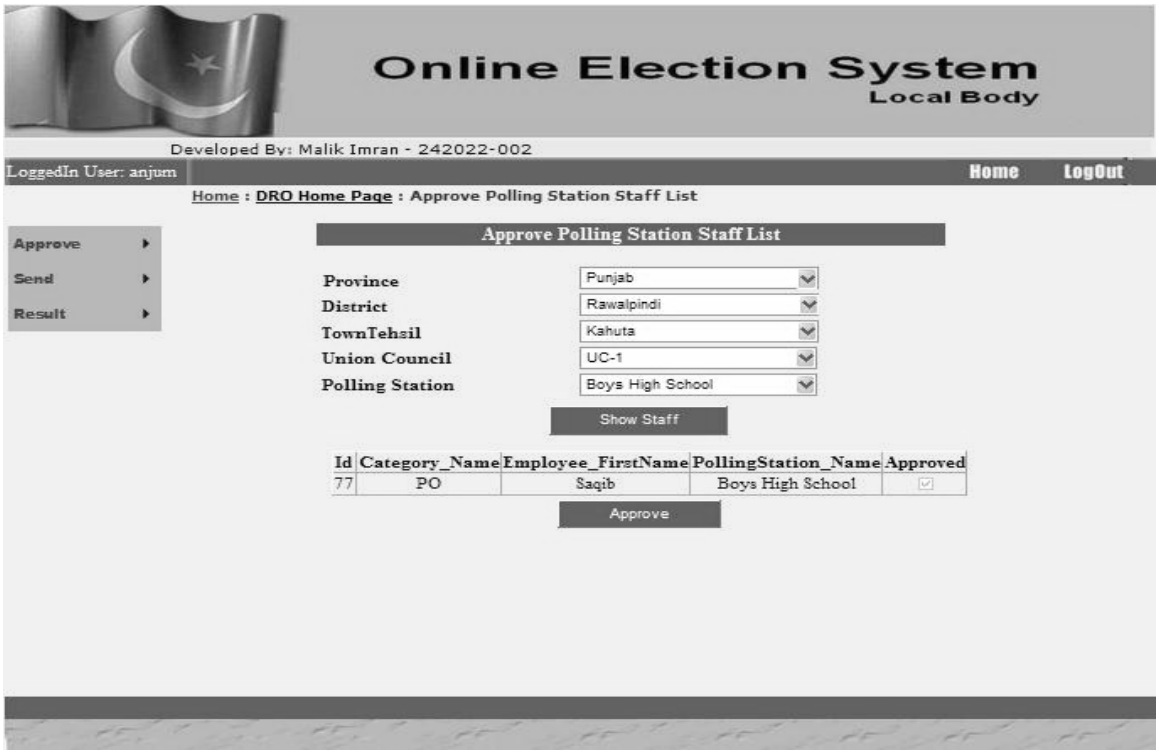


Figure: Approve Polling station staff List

- Send voter list approved by CEC to concern Town/Tehsil RO

Figure: Send Voter list to DRO

- DRO can see results of all Town/Tehsil's & Union council's of certain province

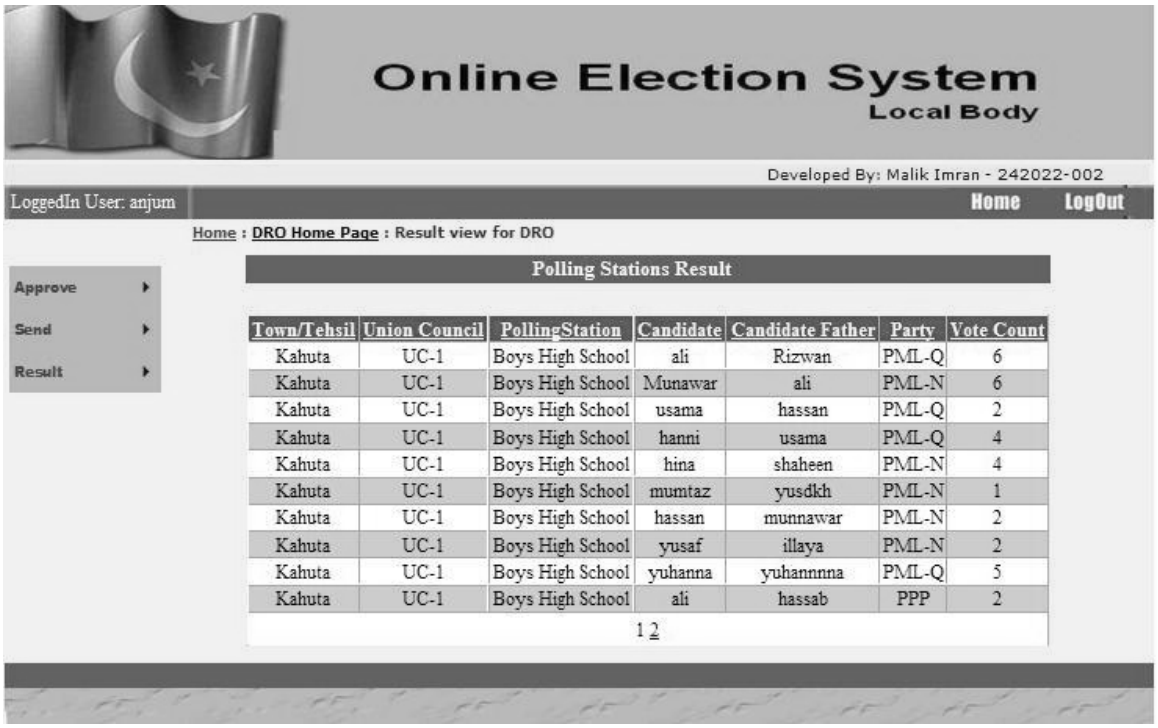


Figure: View Result DRO

RO

- Prepare list of candidates category, symbols, parties & assigning it to all candidates
- Upload Candidate form provided by Election Commission

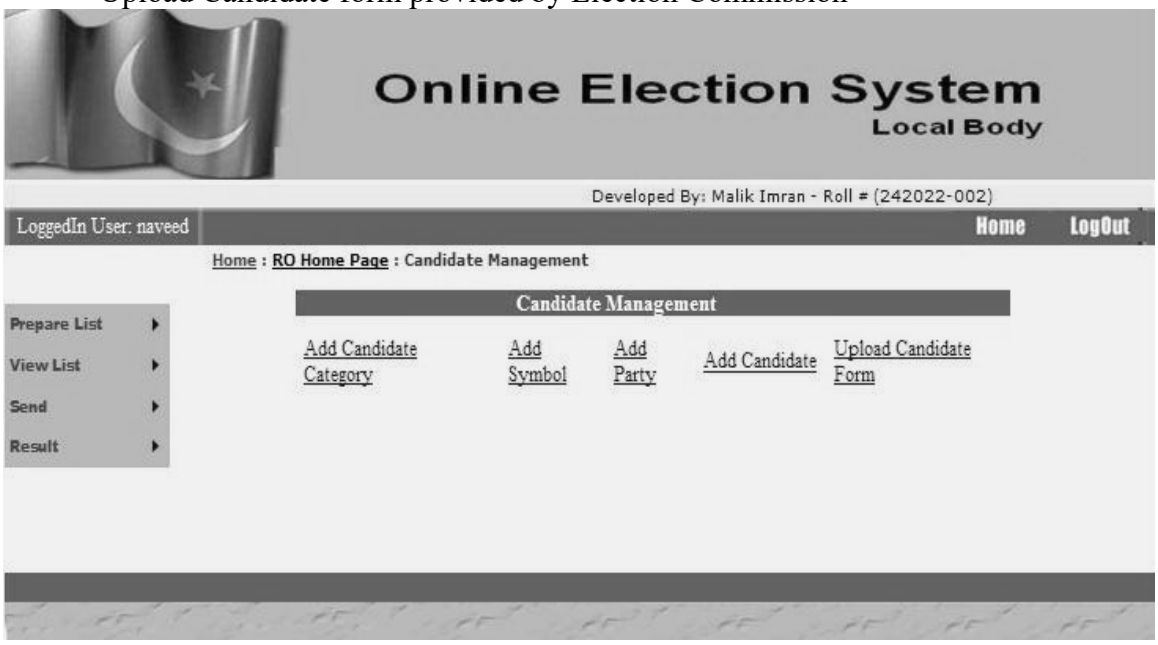


Figure: Candidate Management

- Assign PO role to already created Employee list by ACEC

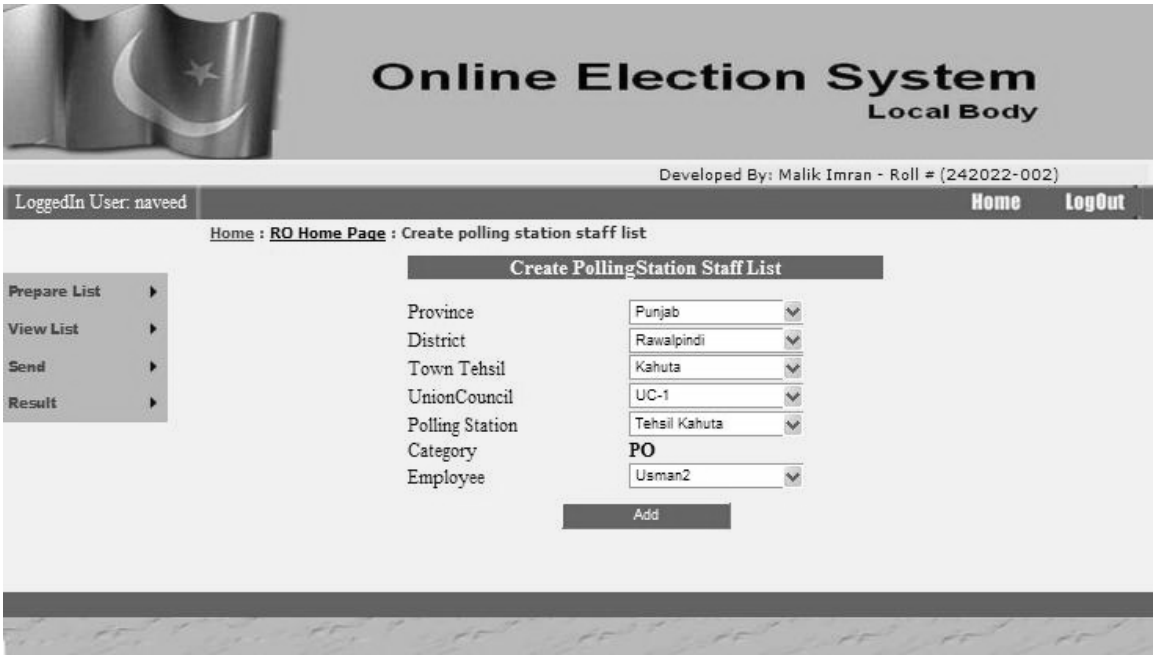


Figure: Prepare Polling station staff list

- Creates polling station for Union Council of selected Town/Tehsil

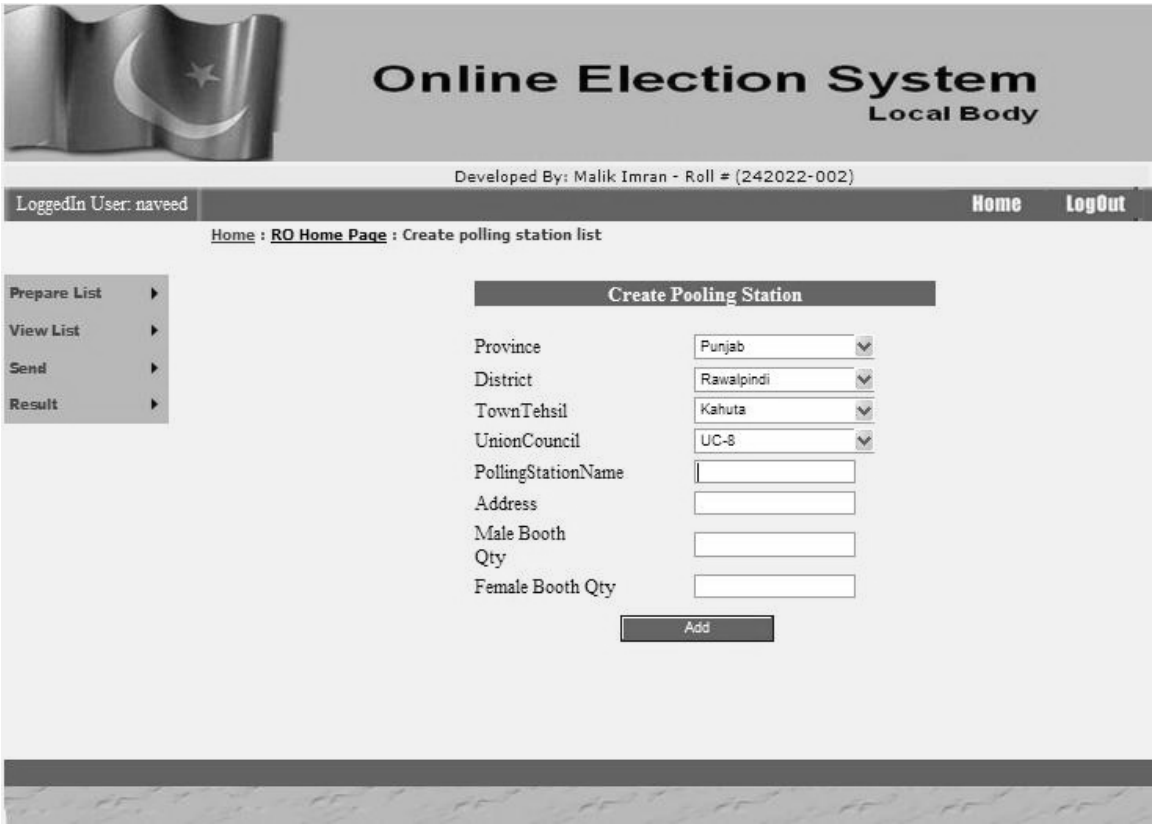


Figure: Prepare Polling station List

- After creating the Polling station Booth is created for vote casting

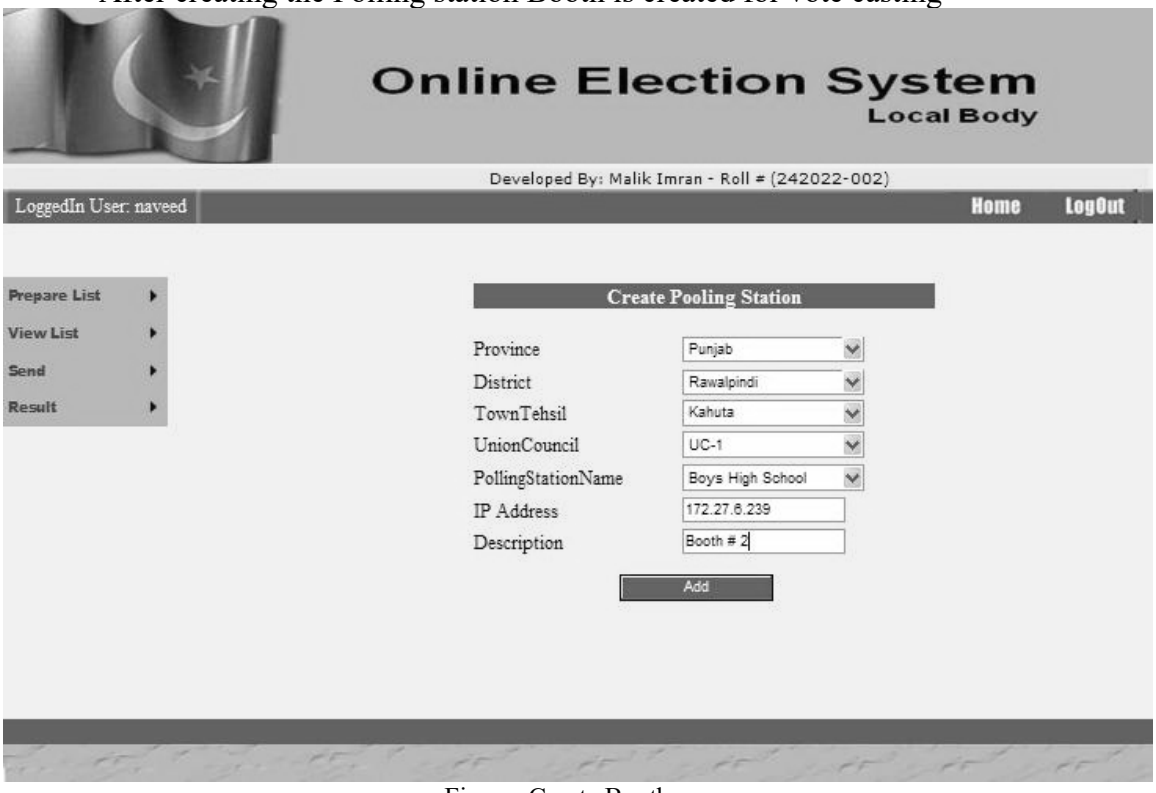


Figure: Create Booth

- View List for all created candidates
- Can be updated or deleted if required

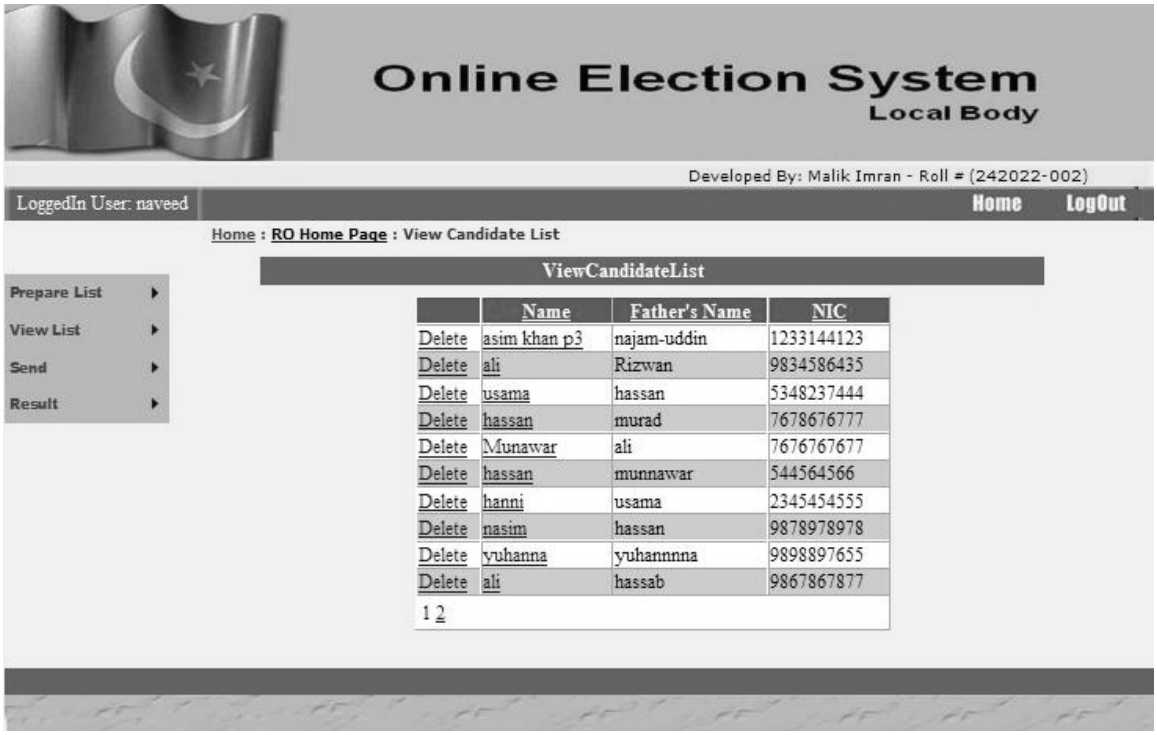


Figure: View Candidate List

- View List for all created Polling station staff list
- Can be updated or deleted if required

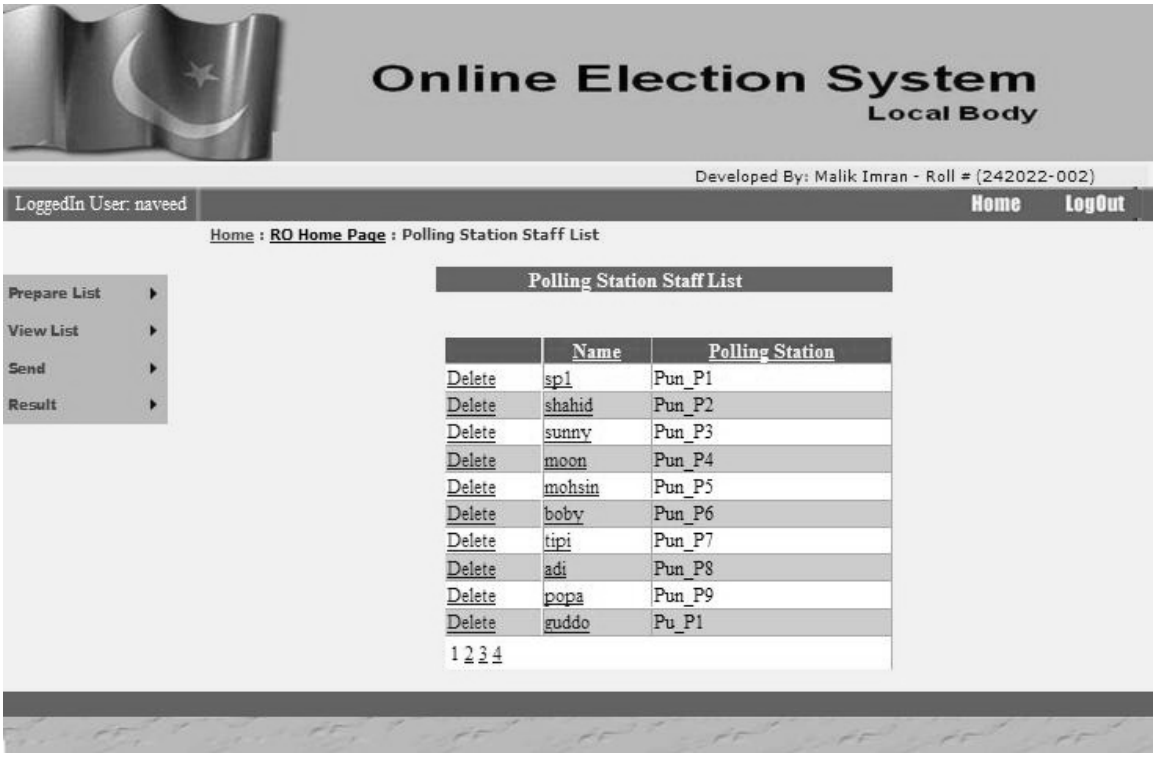


Figure: Polling station staff list

- View List for all Polling station list
- Can be updated or deleted if required

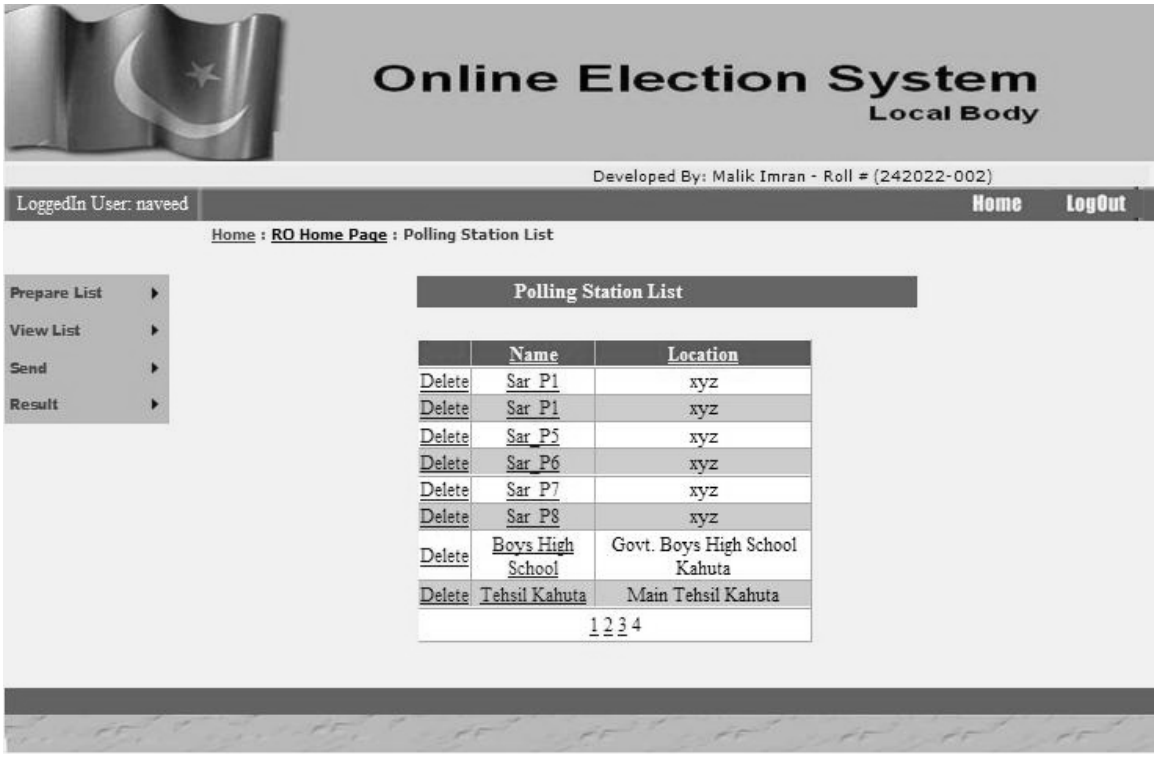


Figure: Polling station list

- Send created list of candidates to PO

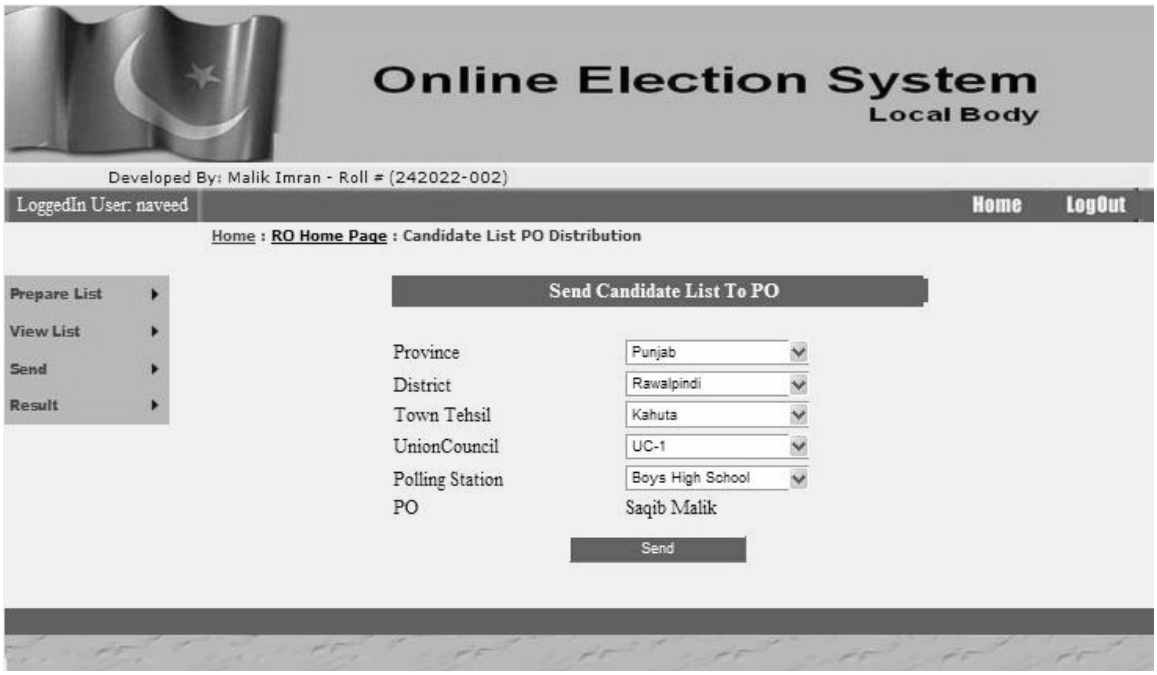


Figure: Send candidate list to PO

- Send Polling station list to DRO for approval

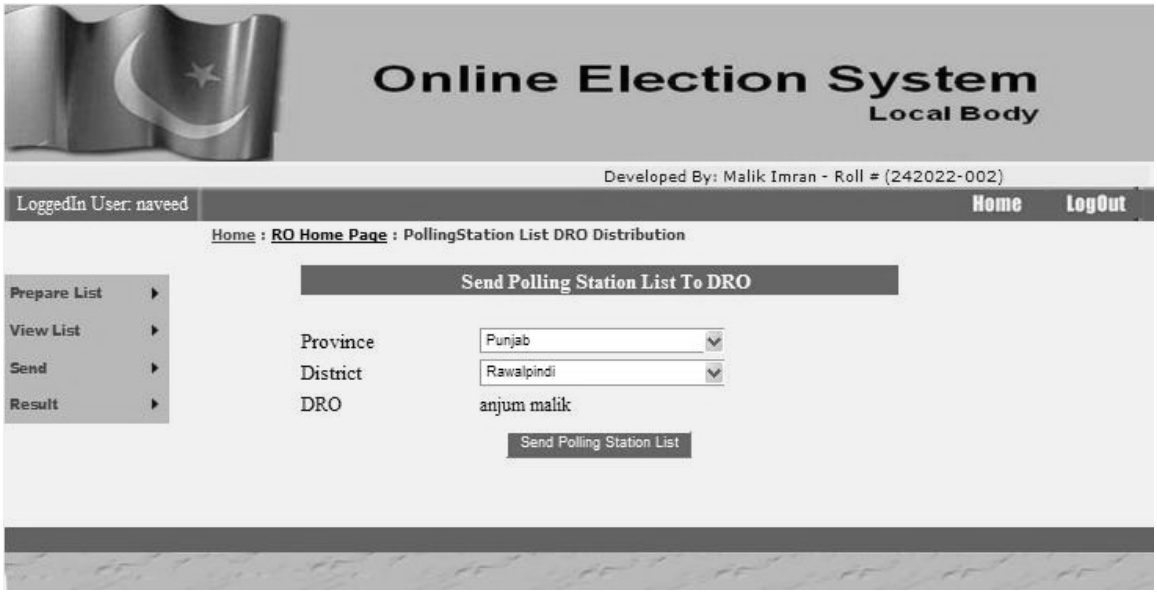
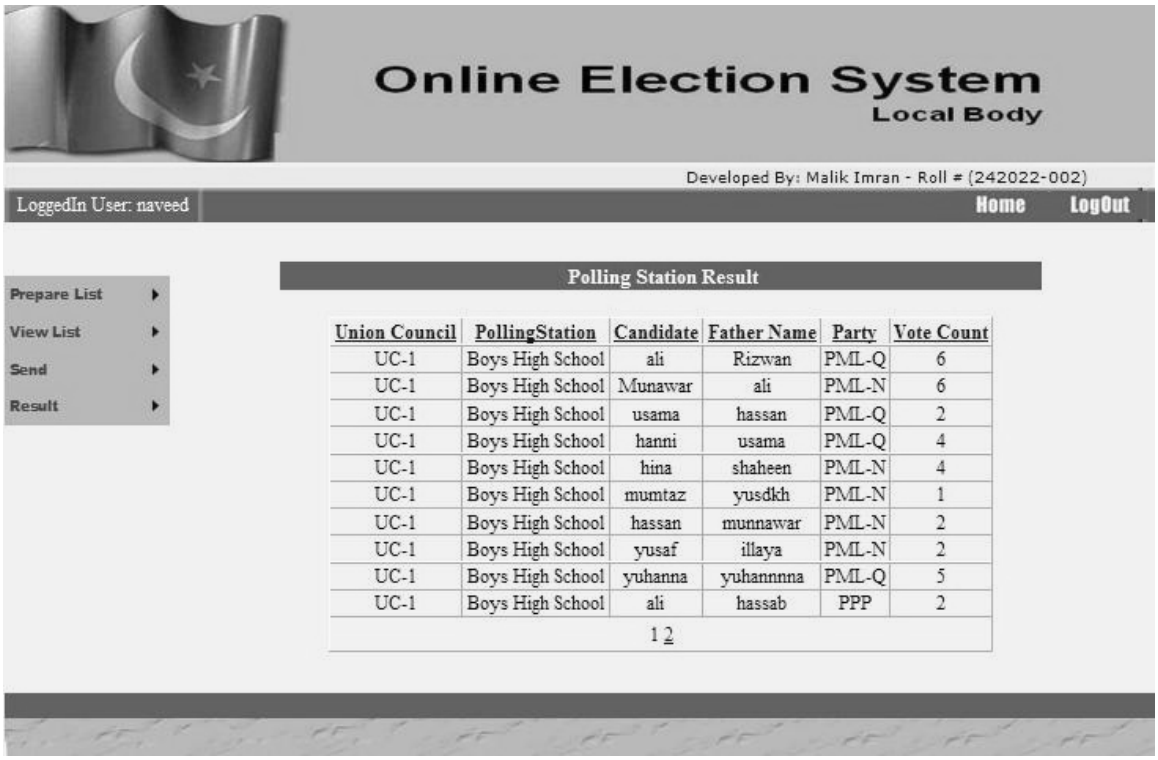


Figure: Send polling station list to DRO

- View Result for Union Council & Polling stations



Union Council	PollingStation	Candidate	Father Name	Party	Vote Count
UC-1	Boys High School	ali	Rizwan	PML-Q	6
UC-1	Boys High School	Munawar	ali	PML-N	6
UC-1	Boys High School	usama	hassan	PML-Q	2
UC-1	Boys High School	hanni	usama	PML-Q	4
UC-1	Boys High School	hina	shaheen	PML-N	4
UC-1	Boys High School	mumtaz	yusdkh	PML-N	1
UC-1	Boys High School	hassan	munnawar	PML-N	2
UC-1	Boys High School	yusaf	illaya	PML-N	2
UC-1	Boys High School	yuhanna	yuhannnna	PML-Q	5
UC-1	Boys High School	ali	hassab	PPP	2

1 2

Figure: Result RO

PO

- Ask voter for NIC & enter its number

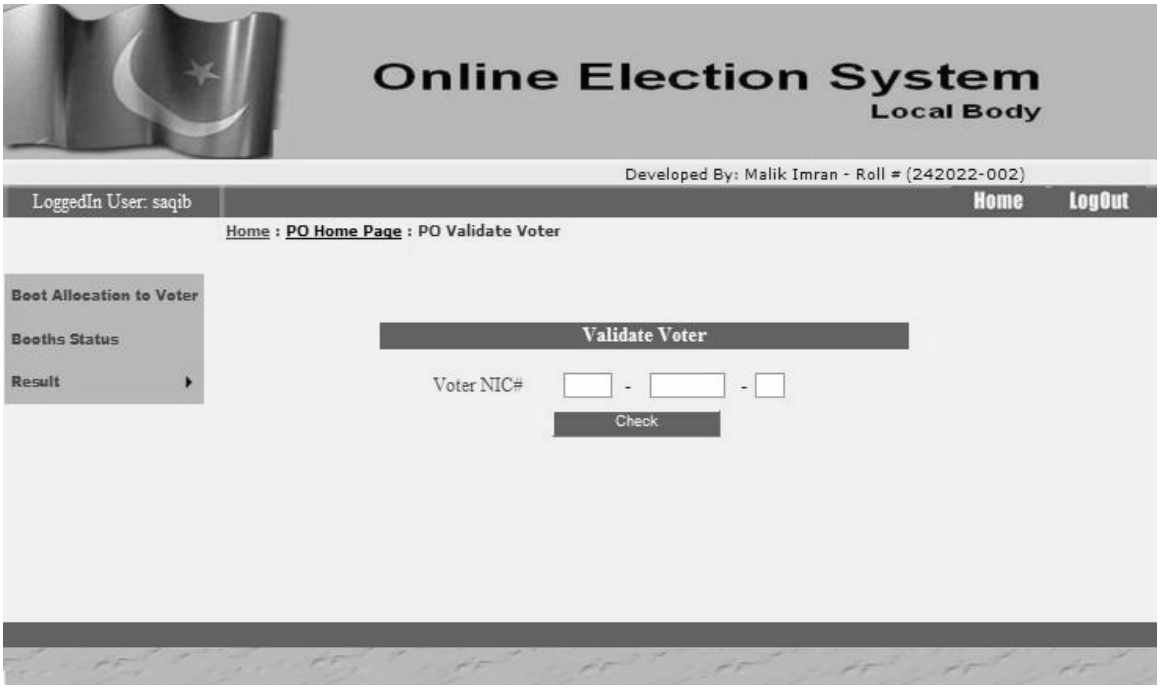


Figure: Booth Allocation to Voter

- Can view the Result of Polling station

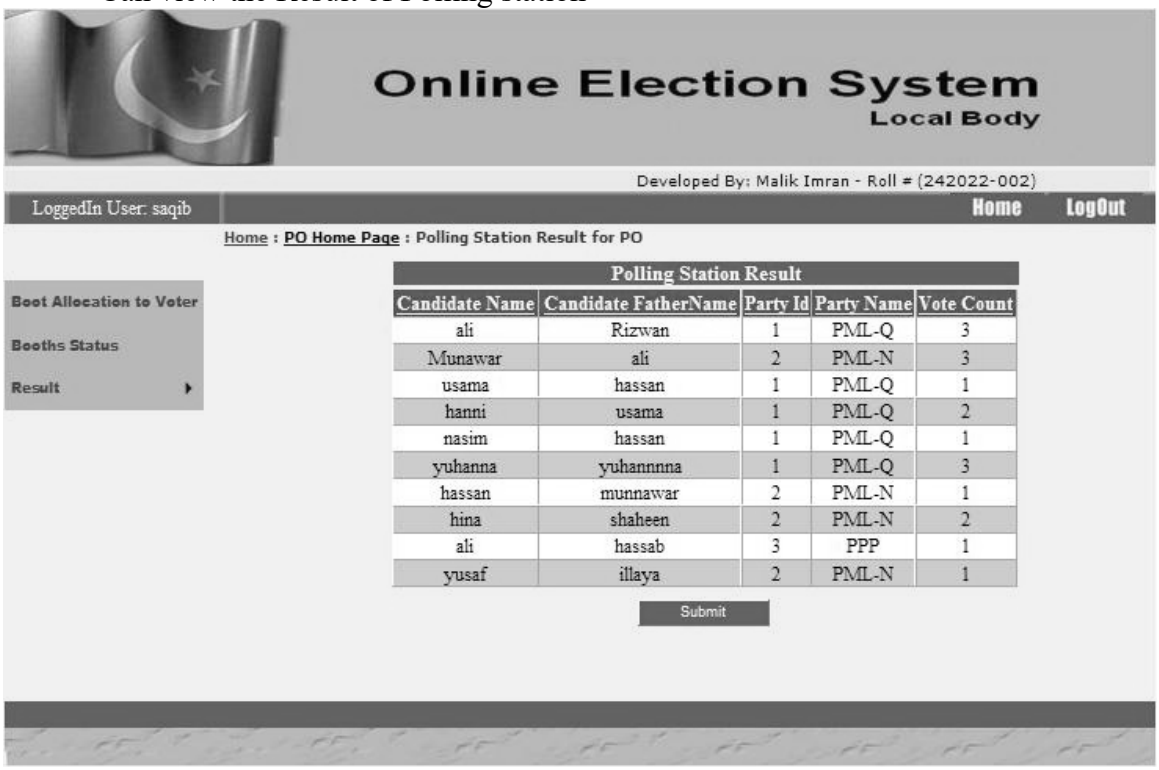


Figure: View Result PO

- Screen shown on the Booth for voter to continue after NIC validation from PO

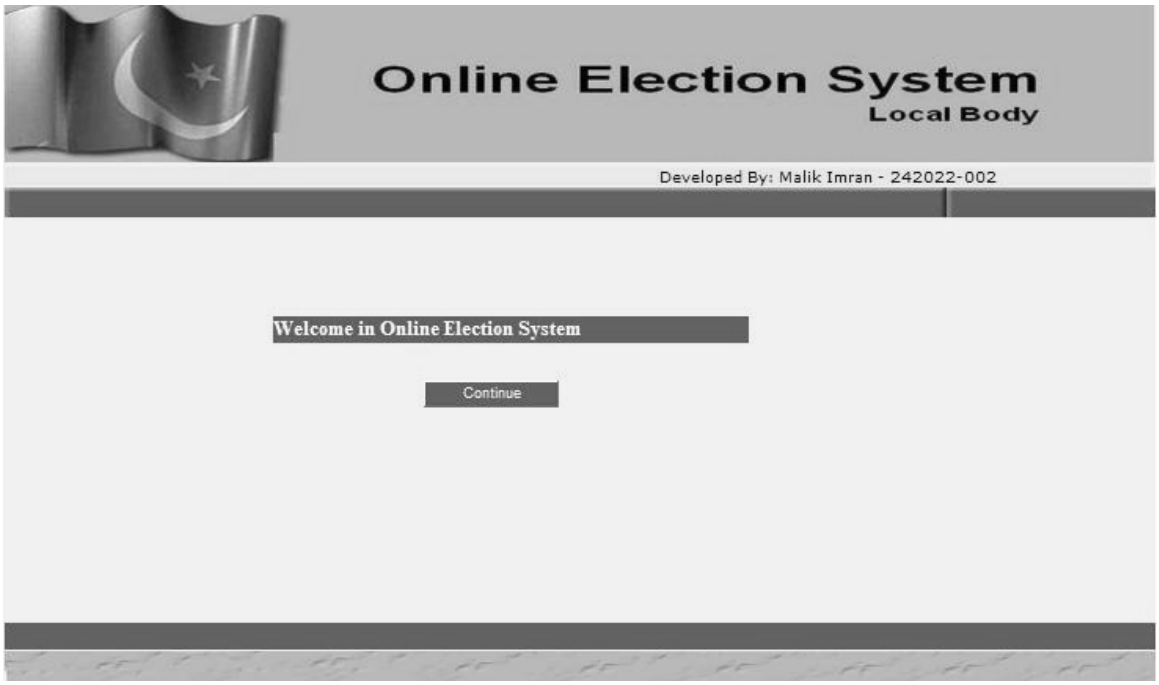


Figure: Booth welcome screen

- Contact Us Screen

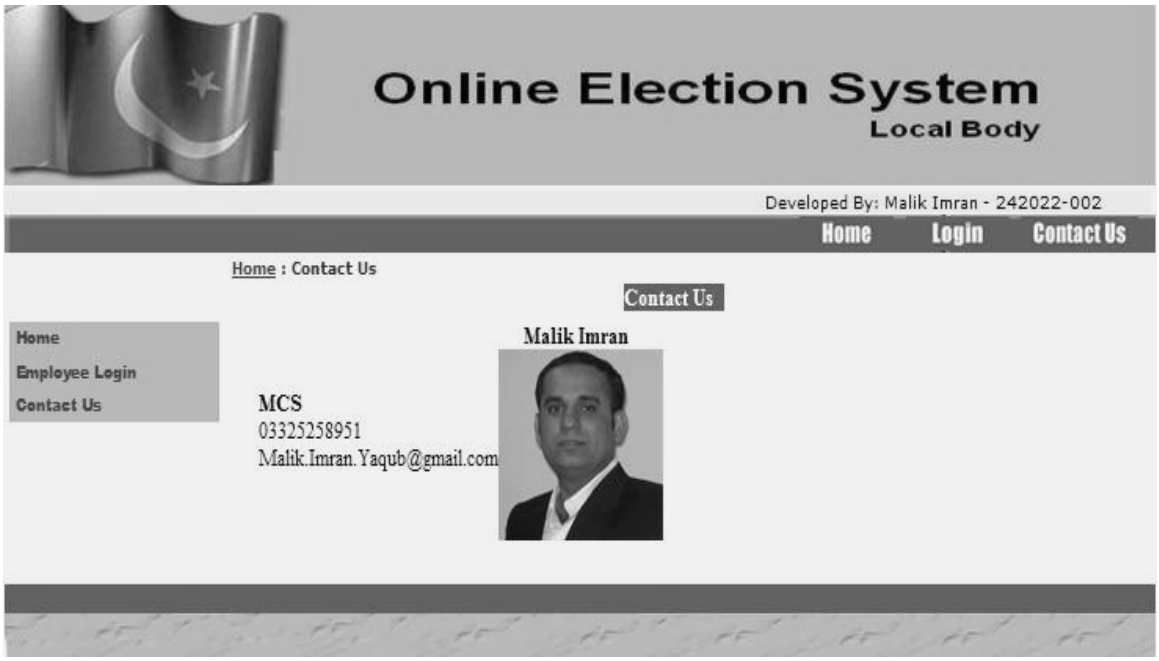


Figure: Contact Us details

APPENDIX E

REFERENCES

References

Books

1. Bernd Bruegge & Allen H. Dutoit, Object-Oriented Software Engineering Using UML, Patterns, and Java. Second Edition

2. Mark Priestley, Practical Object-Oriented design with UML, Second Edition, TATA McGraw HILL

3. Craig Larman, Applying UML and patterns, Third Edition, PEARSON Education

4. Charles Wright, C# TIPS & TCHNIQUES, TATA McGraw HILL

5. Art Gittleman, Computing with C# and .NET Framework, Jones & Bartlett Publishers.

Web

1. www.elections.com.pk
2. www.ecp.gov.pk
3. www.dotnetspider.com
4. www.wikipedia.org
5. www.aspfree.com
6. msdn.microsoft.com