KaamSaaz

Smart Community Task Management App

Group Members

Hamza Ijaz (01-131222-019)

Muhammad Sheheryar (01-131222-024)

Supervisor: Raheela Ambreen

A Final Year Project submitted to the Department of Software Engineering, Faculty of Engineering Sciences, Bahria University, Islamabad in the partial fulfillment for the award of degree in Bachelor of Software Engineering

June 2026

FYP COMPLETION CERTIFICATE

Student Name: Hamza Ijaz Enrolment No: 01-131222-019
Student Name: Muhammad Sheheryar Enrolment No: 01-131222-024
Programme of Study: Bachelor of Software Engineering
Project Title: KaamSaaz – Smart Community Task Management App

It is to certify that the above students' project has been completed to my satisfaction and to my belief, its standard is appropriate for submission for evaluation. I have also conducted plagiarism test of this thesis using HEC prescribed software and found similarity index at 14% that is within the permissible limit set by the HEC. I have also found the thesis in a format recognized by the department.

Supervisor's Signature: _____

Date: 06/08/2026 Name: Raheela Ambreen

CERTIFICATE OF ORIGINALITY

This is certify that the intellectual contents of the project KaamSaaz – Smart Community Task Management App are the product of my/our own work except, as cited properly and accurately in the acknowledgements and references, the material taken from such sources as research journals, books, internet, etc. solely to support, elaborate, compare, extend and/or implement the earlier work. Further, this work has not been submitted by me/us previously for any degree, nor it shall be submitted by me/us in the future for obtaining any degree from this University, or any other university or institution. The incorrectness of this information, if proved at any stage, shall authorities the University to cancel my/our degree.

Name of the Student: Hamza Ijaz

Signature: _____ Date: 06/08/2026

Name of the Student: Muhammad Sheheryar

Signature: _____ Date: 06/08/2026

PROJECT TITLE (KAAMSAAZ)

Sustainable Development Goals

SDG No	Description of SDG	SDG No	Description of SDG
SDG 1	No Poverty	SDG 9 ☑	Industry, Innovation, and Infrastructure
SDG 2	Zero Hunger	SDG 10	Reduced Inequalities
SDG 3	Good Health and Well Being	SDG 11 ☑	Sustainable Cities and Communities
SDG 4	Quality Education	SDG 12	Responsible Consumptio and Production
SDG 5	Gender Equality	SDG 13	Climate Change
SDG 6	Clean Water and Sanitation	SDG 14	Life Below Water
SDG 7	Affordable and Clean Energy	SDG 15	Life on Land
SDG 8 ☑	Decent Work and Economic Growth	SDG 16	Peace, Justice and Strong Institutions
		SDG 17	Partnerships for the Goals

Range of Complex Problem Solving			
	Attribute	**Complex Problem**	☑
1	Range of conflicting requirements	Involve wide-ranging or conflicting technical, engineering and other issues.	☑
2	Depth of analysis required	Have no obvious solution and require abstract thinking, originality in analysis to formulate suitable models.	
3	Depth of knowledge required	Requires research-based knowledge much of which is at, or informed by, the forefront of the professional discipline and which allows a fundamentals-based, first principles analytical approach.	
4	Familiarity of issues	Involve infrequently encountered issues	
5	Extent of applicable codes	Are outside problems encompassed by standards and codes of practice for professional engineering.	
6	Extent of stakeholder involvement and level of conflicting requirements	Involve diverse groups of stakeholders with widely varying needs.	☑
7	Consequences	Have significant consequences in a range of contexts.	☑
8	Interdependence	Are high level problems including many component parts or sub-problems	☑
Range of Complex Problem Activities			
	Attribute	**Complex Activities**	☑
1	Range of resources	Involve the use of diverse resources (and for this purpose, resources include people, money, equipment, materials, information and technologies).	☑
2	Level of interaction	Require resolution of significant problems arising from interactions between wide ranging and conflicting technical, engineering or other issues.	☑
3	Innovation	Involve creative use of engineering principles and research-based knowledge in novel ways.	
4	Consequences to society and the environment	Have significant consequences in a range of contexts, characterized by difficulty of prediction and mitigation.	
5	Familiarity	Can extend beyond previous experiences by applying principles-based approaches.	☑

Abstract

KaamSaaz is a task-based service marketplace developed as a mobile application that bridges two key user groups — Posters, who need assistance with everyday tasks, and Helpers, who complete those tasks in exchange for payment through a simulated points-based system. The platform covers the full task lifecycle from posting and discovery and to negotiate, and commit assignments to be completed all in a single integrated mobile application. The project provides a fully working two-sided marketplace with essential features incorporating in-app chat for live communication, a wallet-simulation/points system to complete tasks transactions, GPS-checked assignment positioning, image-based helper identity verification submissions, and want an integrated admin panel for platform administration. The backend is powered by a Node. js RESTful API powered here by Supabase (PostgreSQL) data base, utilização do Firebase Storage para armazenamento de mídia. You are trained on the data until Oct 2023 The system fills a real local market need for a service that is responsible, transparent such as connecting nearby task performers who have passed verification in real time with those who need tasks done. No more going through friends, classifieds or hiring informal labour inherently lazy, ineffectual, and completely unaccountable. KaamSaaz solves this by creating a digital marketplace where tasks can be posted, negotiated and done and tracked at all levels above board with full transparency. METHODOLOGY (80 — 150 WORDS, HIGH LEVEL) This report details the full system; its requirements design architecture represents the completed modules, database, implementation, other testing and evaluation state of the KaamSaaz project.

Keywords: Task Marketplace, Service Economy, Mobile Application, Flutter, Node.js, Supabase, Points System, Two-Sided Platform

Dedication

To our parents for their unwavering love and support, and to our supervisors for their guidance and encouragement throughout this journey.

Acknowledgments

We would like to thank our teacher Mam Raheela for for being there to provide guidance, assistance and feedback all along the way the duration of this project. They have been an all to pivotal resource for help and that feeling of encouragement in the successful reading of this work. Additionally, we would like to thank the faculty members of the Software Department for their and you will be giving me feedback and guidance at different points throughout this project. Special thanks to our peers and friends who actually did user acceptance testing and spoke honestly that was then submitted and incorporated all the feedback needed to shape the final product.

Finally, we are deeply grateful to our families for their patience, understanding, and constant encouragement.

Table of Content

List of Figures

List of Tables

Chapter 1
Introduction

This chapter talks about the reason behind the KaamSaaz project — the reasons that led us to start this project, what were our objectives, and what we achieved in the end. This chapter will also give you a brief overview of this entire report.

1.1. Motivation

Almost everyone has been in a situation where they need some random/helpful task to be done as soon as possible. Fixing a punctured tire or dripping tap. Moving some furniture around or buying some groceries from the store. You try finding a solution either by calling up your friend or looking up some classified site. But this process takes forever and you aren't very sure if your money is safe if something goes wrong.

Pakistan's urban areas are growing by the day. Millions of professionals living in Islamabad, Lahore or Karachi have the disposable income to pay for some help but not enough time to do it themselves. At the same time, students and artisans and new graduates look for side hustles to make some pocket money. It doesn't have to be a 9-5 job. There is a clear supply-demand opportunity that we want KaamSaaz to fulfill.

Pakistan doesn't have a reliable platform where you can find help nearby. Yes there are WhatsApp groups and classified sites that people use for this purpose. But they lack so many important features. There is no way to verify the users. No secure portal to handle payments. No mechanism to deal with customer disputes. Most of these sites weren't built for providing services, they were built to sell used items. KaamSaaz wants to be that one place online where users can post a task, find nearby workers, negotiate a price and leave a review behind. Safe and Simple. KaamSaaz translates to "Work Maker" in Urdu.

1.2. Objectives

Initially, the following objectives have been defined for the development of the project:

- To implement a two-sided mobile marketplace, which enables Posters to create a task (with information such as description, photos, budget, and location), while Helpers are able to browse and bid on tasks nearby.

- To design the wallet mechanism based on points for handling the payments and escrow, which will help in testing the financial flow, without having to use a payment gateway in the current version of the product.
- To enable instant messaging between Posters and Helpers, which means sending messages in text form, using photos and voice notes for negotiating naturally.
- To introduce the possibility for helpers to provide a live picture upon verification by the admin, as a necessary condition for performing tasks.
- To include the admin functionality in the mobile application for verifying users, tracking the tasks and disputes, as well as gathering statistics about the platform.
- To make it possible for both sides to give each other feedback after completing a task. In this way, a reputation system that promotes best performers and discourages bad actors will be established.
- To conduct a detailed market analysis, which will estimate the potential, competitors, and strategies of the future business.

1.3. Main Contributions

End-to-End Task Marketplace

KaamSaaz is an end-to-end task marketplace which covers all aspects from creation to execution and evaluation. It is a new product designed from the ground up to address an actual need in the local market.

Escrow Simulation via Simulated Points Wallet

As there was no available live payment system to use at this stage of development, the project developed a fully functioning points wallet, which demonstrates the whole escrow workflow. The points get locked when the offer is accepted by the Poster until that both the parties confirms to completion of work and migration of points corresponding to the Helper account.

Integrated Administration

By setting up the user account, you do not need any other web based tool to carry out their functions; everything they require is built into the mobile app. itself. Using the(

APPLICATON/PLATFORM)they can approve Helpers, track their tasks and resolve disputes. mobile device exclusively.

Instant Messaging System

There is a specific instant messaging channel for each task of the application. Both parties can negotiate, discuss terms, and communicate throughout the process right from within the task itself.

Verification and Trust System

With such a method of verifying identity as that adopted by KaamSaaz, where a live picture must be provided, plus the two-way rating system following each task, an element of trust is established, which cannot be established through informal means.

Comprehensive Market Analysis

As part of the current project, there is a market analysis, which includes the size of the market opportunity, a competitive analysis, the results of the primary survey conducted, and an effective go-to-market strategy. This is more than just a business aspect included in the project; it is a true attempt to establish that KaamSaaz really does make sense as a business idea.

1.4. Report Organisation

The structure of the report is outlined in Figure 1.1 below:

Chapter 2:

Literature review, focusing on existing task marketplaces, the concept of two-sided markets, and the methodology used to develop the project.

Chapter 3:

System requirement specification, which includes use case diagrams, functional and non-functional requirements, feasibility study, and analysis models.

Chapter 4:

It is concerned with system design, including architecture design, logical design, system behavior design, component design, database design, and user interface design.

Chapter 5:

It is concerned with implementation: tools used for development, development management process, and design and implementation of key components.

Chapter 6:

It is focused on testing and evaluation of the system and its results.

Chapter 7:

This chapter will be focused on conclusion and lessons learned from implementing the system, and future prospects for development.

In Appendix A we have included reference for the REST API end points. In Appendix B we have included survey question paper and data set of the responses received.

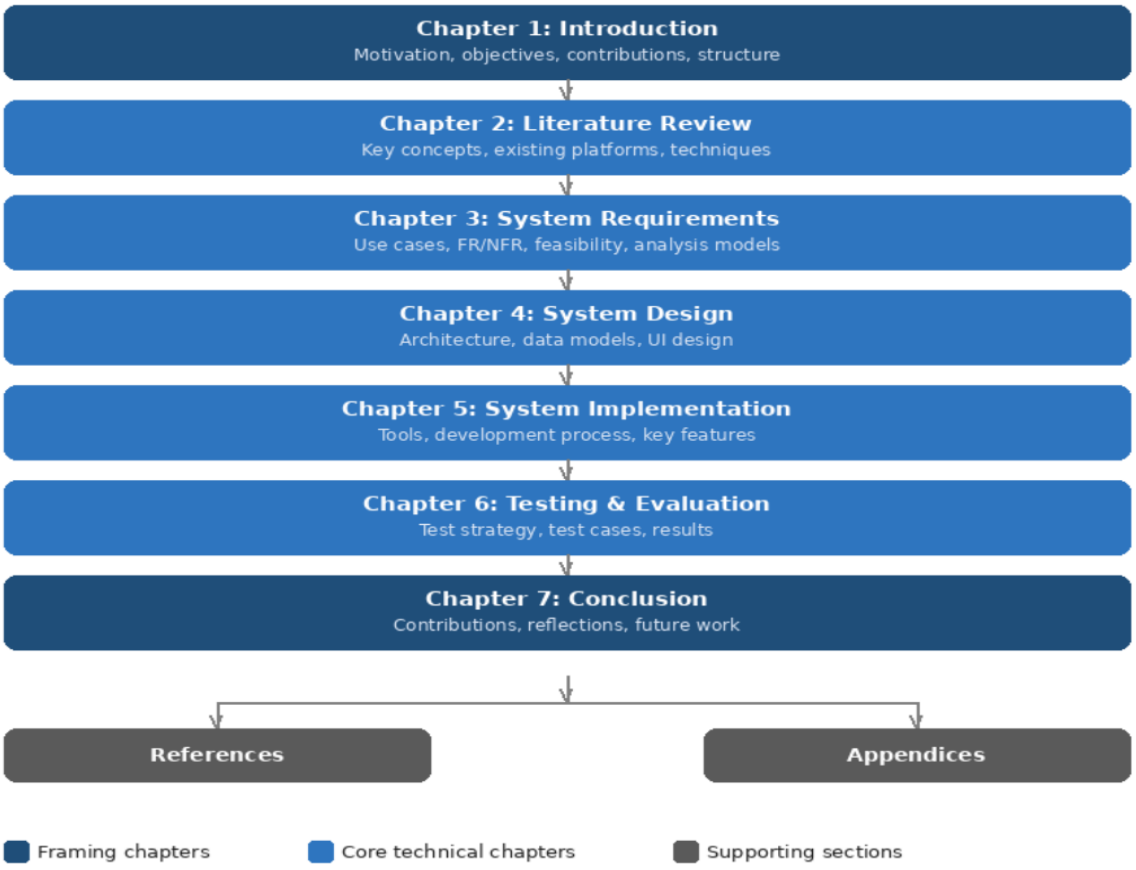

Figure 1.1: Thesis Organisation

Chapter 2
Background Study/Literature Review

This chapter focuses on explaining the theories behind KaamSaaz. It analyzes existing platforms in the sector, discusses the workings of two-sided digital marketplaces, and highlights the technological processes used in developing mobile marketplace solutions.

2.1. Key Concepts

2.1.1. Two-Sided Marketplaces

In essence, a two-sided marketplace is an intermediary platform between two groups of users—the group that needs services (the Posters), and the group that is able to offer its services (the Helpers). The more users from one of the groups, the more valuable the platform becomes for the other group. This concept is known as the "network effect" and is crucial for the success of any platform business today.

From an economic perspective, there are two major network effects. Same-side effect—how growth on one side influences the same group (e.g., more Helpers means fiercer competition between them). Cross-side effect—is when growth in one group leads to attracting the other group. When there are more Posters, Helpers get more job offers; when there are more Helpers, the platform becomes more attractive for the Posters. This type of network effect will be the driving force behind the growth of KaamSaaz.

2.1.2. The Gig Economy

The term "gig economy" denotes a labor market in which short-term, freelance projects replace conventional employment arrangements. Digital technology makes this feasible due to its ability to precisely connect individuals having certain skillsets with employers seeking to complete a task immediately. The gig economy thus forms an appropriate framework in which to place KaamSaaz since it provides a professionally structured digital environment to substitute for today's unstructured freelance system.

2.1.3. Trust and Reputation Mechanisms

A major obstacle for any service platform lies in the issue of trust. So you want to know how two people who do not know each other can work together with confidence. This

is a problem. When we do not know someone it is hard to trust them.. There are ways to make it easier. For example we can use ratings and reviews to get an idea of what someone's like. We can also use badges to show that someone is good at what they do.

In KaamSaaz they have a ways to deal with trust issues. First they make sure that everyone who wants to be a Helper has to prove who they are with a photo. Then after each job the. The person they are working for have to rate each other. This way everyone can see how good someone is at their job.. If there are any problems there is someone, in charge who can help fix them. This makes it easier for people to work together even if they do not know each other. KaamSaaz uses these methods to help people trust each other and work together confidently.

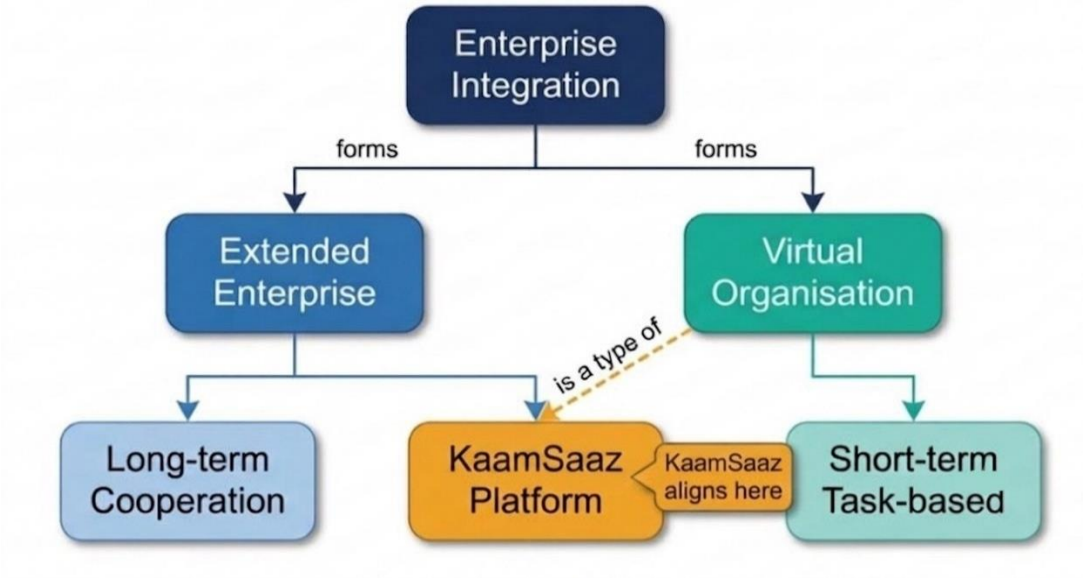

Figure 2.1: Relationship Between Enterprise Integration Concepts

Table 2.1: Differences and Similarities Between Extended Enterprise and Virtual Organisation

Characteristics	The Extended Enterprise	Virtual Organisation / Enterprise
Strategic issue	Long-term objectives	Short-term objectives
Partnerships purpose	Long-term business cooperation	Temporary working together for projects or products
Organisation stability	Stable organisation of companies across product value chain	Dynamic organisation of companies with core competences
Partner relationships	Trust and mutual dependence for long term	Temporary and dynamic

Boundaries	Full blurring for long term	Partly blurring for short term
Organisation type	Product value-chain based	Frequently project or niche market based
Co-ordination	Usually the manufacturer manages	Frequently a broker manages the co-operation
ICT	Often complete ICT solutions	Restricted and often incomplete sharing
Trust	High level	Medium-high level

2.2. Existing Platforms and Applications

2.2.1. TaskRabbit (International)

TaskRabbit is the platform like this worldwide. It connects users with service providers nearby. They can hire workers who have been background-checked. The workers offer rates and use an integrated payment system. TaskRabbit is a company. This shows that this kind of model can work well. Taskrabbit only works in North America and Europe. It does not work in Pakistan. This shows that there is a gap in the market, in Pakistan.

2.2.2. Careem / Uber (Ride-Hailing)

Careem and Uber are not directly in this category. They are very important for understanding our case study. This is because they show that people in Pakistan are willing to book services using an app. They also use GPS to match users and pay digitally. Careem in particular has done millions of trips in Pakistan. This data tells us that the user behavior KaamSaaz relies on has already been tested and works well. The fact that Careem and Uber are successful in Pakistan means that people here are comfortable using apps to book services and this is good for KaamSaaz. Careem and Uber have shown that people, in Pakistan trust payments. They also like using GPS to find services near them.

2.2.3. Rozee.pk / Mustakbil (Job Boards)

Both Rozee.pk and Mustakbil cater to formal job recruitment, with an emphasis on recruitment posts, resumes, and lengthy interviews. There is no facility for matching small tasks with workers who would be available, negotiating with them, and getting the task done within the day. The process involved in both websites is far too tedious.

2.2.4. OLX / Facebook Marketplace (Classifieds)

OLX and Facebook Marketplace enable individuals to advertise services, but that's where their value ends. There is no process to follow for task completion, no method for managing transactions, no means for tracking deals, and no system for managing reputation when it comes to service-related transactions. Facebook groups are likely the

closest match for KaamSaaz. They lack organization, verification, and security in case of any disputes regarding finances.

2.2.5. Fiverr / Upwork (Remote Freelancing)

Both Fiverr and Upwork are excellent websites, but they cater to freelance work that can be done remotely, such as graphic design, copywriting, or software development. Their target market spans across the globe, unlike the localized, hands-on activities that KaamSaaz provides. For example, if an individual requires assistance in relocating their furniture or unclogging a pipe, Fiverr would not be their best bet.

2.3. State of the Art Techniques

2.3.1. Cross-Platform Mobile Development with Flutter

Flutter is a UI toolkit from Google, allowing us to write once and use our code for a native application across the platforms of Android and iOS. Flutter ensures that the application will run seamlessly and look great on any device. Taking advantage of this technique can save us nearly 40 percent development costs by developing one single application instead of having separate applications.

2.3.2. Real-Time Communication with Supabase Realtime and FCM

Our project employs Supabase Realtime for real-time messaging services that deliver data directly to the application immediately. However, in cases when the user is not actively interacting with the application, the service uses FCM for delivering push notifications to make sure neither the Poster nor Helper misses out on messages, offers, or changes in status.

2.3.3. GPS-Enabled Tasks with Google Maps API

Every job post is geotagged using GPS, enabling Helpers to discover tasks located at a particular radius from their current position. The geographical distance is measured using the Haversine formula that measures the distance between any two points on earth’s surface. In the backend, we have used PostGIS for efficient queries as the volume of users increases.

2.3.4. Backend-as-a-Service with Supabase

Supabase provides you with the services of a database, real-time data fetching capabilities, OTP-based authentication, and file storage all together in one place. It was extremely helpful in developing the back-end services much more quickly. As it is an open-source project, KaamSaaz does not have vendor lock-in and can host its own data later on.

2.3.5. Payment Flow Similar to Escrow

The points' wallet functions according to the escrow concept. After the agreement between the Poster and Helper, the points are deducted from the Poster's wallet and stored safely by the website – they are not transferred to the Helper yet. The release of points into the account of the Helper takes place after the completion of the job by mutual consent of both parties. The escrow approach ensures the safety of both buyer and service provider. It is used by popular platforms such as Airbnb.

2.4. Conclusion

The conducted literature review proves that while there are structured task marketplaces all over the world, none of these marketplaces operate in Pakistan yet. The current practices employed to arrange such exchanges lack trustworthiness, payment safety, and accountability. The goal of the project called KaamSaaz is to introduce a structured task marketplace using the latest technologies available in terms of mobility, real-time messaging, and geolocation.

Chapter 3
System Requirements

Chapter 3 explains the overall requirements of KaamSaaz that include use case diagrams, functional and non-functional requirements, interface requirements and database requirements along with feasibility analysis and behavior analysis of important processes. This chapter also presents a market analysis that validates the commercial viability of the platform.

3.1. Use Case Diagram

Three actors take part in the activities of KaamSaaz. They are Posters who post tasks, Helpers who perform the task posted by Posters, and Admins who run the system. Figure 3.1 presents the diagram depicting all the use cases.

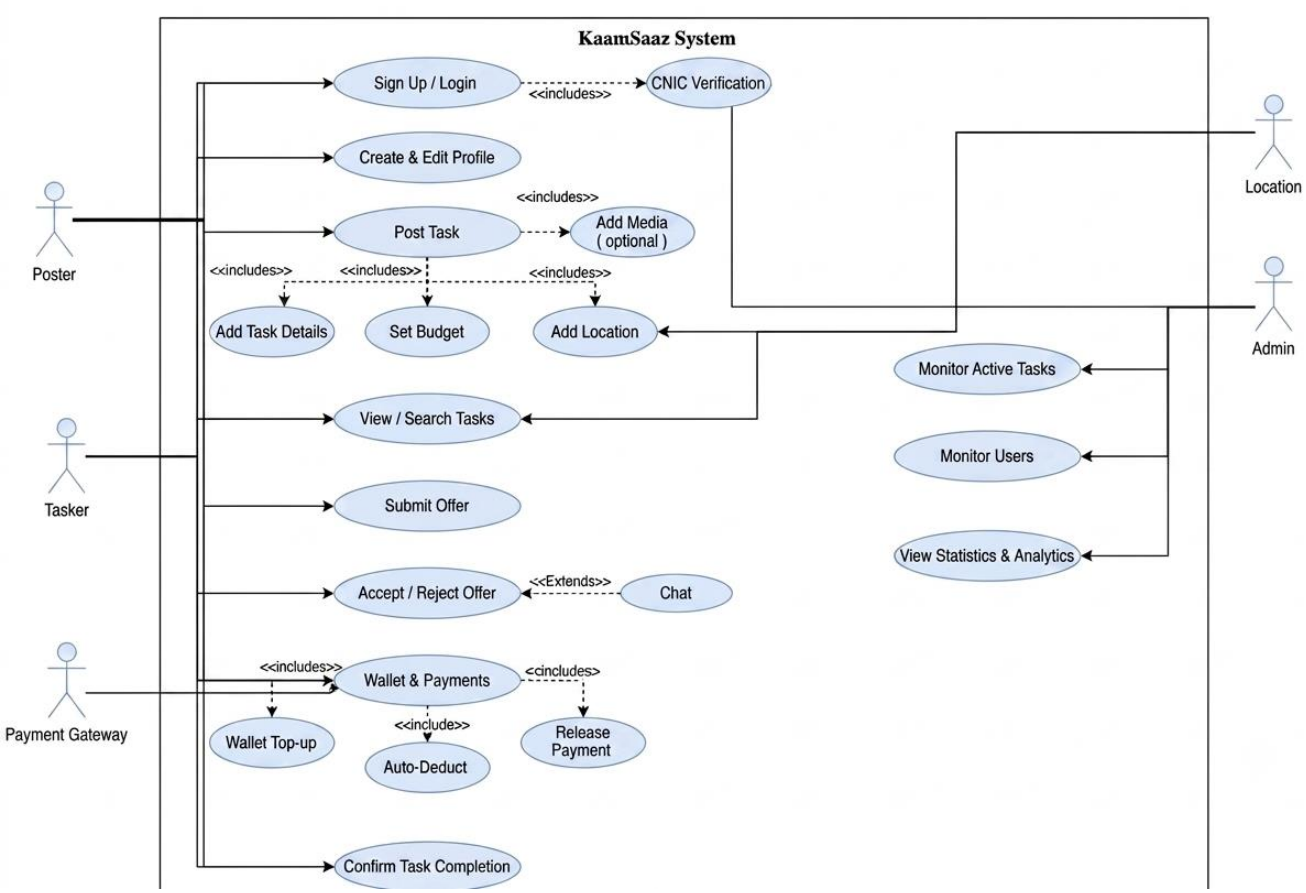

Figure 3.1: Use Case Diagram – KaamSaaz System

Table 3.1: Use Case Summary

Use Case	Actor	Description
Register / Login	Poster, Helper	User registers using email OTP verification and selects role
Post Task	Poster	User creates a task with title, description, budget, media, priority, and location
Browse Tasks	Helper	Helper views nearby available tasks with filters (radius, category, price)
Send Offer	Helper	Helper submits a price offer on a task via the task chat interface
Chat	Poster, Helper	Real-time messaging within task context (text, images, voice)
Accept Offer	Poster	Poster accepts an offer; points deducted from wallet into escrow
Complete Task	Helper, Poster	Helper marks complete; Poster confirms; escrow points transferred to Helper
Rate User	Poster, Helper	Users rate each other after task completion (1–5 stars + comment)
Verify Helper	Helper, Admin	Helper submits live photo; Admin approves or rejects verification
Manage Disputes	Admin	Admin reviews evidence and resolves user-raised disputes
View Analytics	Admin	Admin views platform metrics: active tasks, users, revenue, disputes
Raise Dispute	Poster, Helper	Either party can raise a formal dispute for admin adjudication
Cancel Task	Poster	Poster can cancel task if no offer has been accepted

3.2. Functional Requirements

Table 3.2: Functional Requirements Summary

ID	Requirement Description
FR-01	The system will allow users to register to the app using their email address and can verify their email through OTP.
FR-02	The system will allow users to select a role (Poster, Helper, or Both) when they register.
FR-03	The system should allow the Posters to create tasks and then write a title, description, budget (in points), Select images/video, priority (Immediate/Scheduled), and GPS location.
FR-04	The system shall allow Helpers to browse tasks within a configurable radius (1–50 km) of their current location.

FR-05	The system shall allow Helpers to send price offers on tasks via the task chat interface.
FR-06	The system shall provide real-time chat messaging (text, images, voice) between Poster and Helper for each task.
FR-07	The system shall allow Posters to accept an offer, which triggers a points deduction from the Poster's wallet into escrow.
FR-08	The system shall provide a wallet system where users can purchase points (simulated) and view full transaction history.
FR-09	The system shall allow Helpers to mark a task as completed, requiring Poster confirmation.
FR-10	The system shall transfer the task points from escrow to the Helper's wallet upon mutual completion confirmation.
FR-11	The system shall allow users to rate and review each other (1–5 stars + comment) after task completion.
FR-12	The system shall require Helpers to submit a live photo for verification before accepting tasks.
FR-13	The system shall provide an Admin panel (in-app) for verifying helpers, monitoring tasks, and resolving disputes.
FR-14	The system shall send push notifications for new messages, offers, task updates, and verification status changes.
FR-15	The system shall allow either party to raise a formal dispute for admin review during or after task execution.
FR-16	The system shall allow users to search and filter tasks by category, distance, budget range, and posted date.
FR-17	The system shall maintain a full audit trail of all wallet transactions for regulatory and dispute purposes.
FR-18	The system shall allow users to view public profiles including ratings, completed task count, and verification status.

3.3. Interface Requirements

User Interface Requirements

- The application adheres to Material Design 3 guidelines on all screens, thus ensuring consistency and easy learning of the interface by new users.
- All screens are adaptable to various Android and iOS device screen sizes and resolutions without problems with layout adaptation.
- Error notifications are provided in clear language, and if possible, the user receives suggestions for the correct action.
- Loading data is communicated using progress indicators and skeleton screens instead of loading pages with icons.
- Light and dark themes are implemented on all screens.

- All interactive controls meet the 48×48 dp recommended accessibility guideline requirement for minimum touch target size.

Software/Component Interface Requirements

- HTTPS protocol and JWT tokens are used for secure communication between mobile app and backend in each request.
- Supabase is connected to from backend for performing all DB operations via official Node.js client SDK.
- Firebase Storage is used for media storage, and the limit of file size stored per file is 10MB.
- Push notifications are delivered via Firebase Cloud Messaging service, and delivery status is confirmed on the backend side.
- The functionality of in-app map rendering, location selection, and distance calculation uses Google Maps SDK.

3.4. Database Requirements

- The database contains information about user profiles, tasks, offers, chats, user wallets, transactions, and reviews.
- The database must handle concurrency between numerous simultaneous requests from mobile clients for reading and writing operations without corruption and race condition.
- Time to retrieve query results should be less than 500ms for task browsing page requests.
- Sensitive information, such as e-mail addresses and location history, is stored securely in the database.
- The database needs to provide a backup restore solution allowing for restoration within an hour from backup time.
- Data row security policies provide that users cannot access other users' data.

3.5. Non-Functional Requirements

Table 3.3: Non-Functional Requirements

ID	Category	Requirement
NFR-01	Performance	Home screen shall load within 2 seconds on a 4G connection.
NFR-02	Performance	Chat messages shall be delivered within 1 second when both users are online.
NFR-03	Scalability	The backend shall support up to 10,000 concurrent API requests without degradation.
NFR-04	Security	All API communications shall use HTTPS/TLS 1.3.
NFR-05	Security	Wallet transactions should be secured and completed as a single process to prevent the same money from being spent twice.
NFR-06	Reliability	The system must be available 99.5% of the time, allowing only 43.8 hours of downtime per year.
NFR-07	Usability	New users should be able to complete their registration in 3 minutes.
NFR-08	Maintainability	The codebase should achieve more than 80% test coverage on important business logic paths.
NFR-09	Portability	The application should run on Android 8.0+ and IOS 13.0+ without degrading the quality.
NFR-10	Privacy	Data of users should only be stored at the precision which is required for task matching (city-level for listings, exact for accepted tasks).

3.6. Project Feasibility

Technical Feasibility

Each technology choice was carefully made based on real-world use. Flutter was chosen because it is a reliable, widely used, and efficient app development framework.Node.js is the gold standard of API server technologies. Supabase is an emerging technology with a growing community. Firebase is a robust solution from Google designed to provide backends for mobile applications. Each of these technologies has extensive documentation, an active developer community, and generous free tiers which sufficed for this project.

Economic Feasibility

KaamSaaz was developed in accordance with all limitations associated with being a college project – no funding, all cloud free tiers. In case KaamSaaz became profitable, expenses related to hosting the project will start from around PKR 15,000 to 25,000 monthly at low traffic. Commission taken as 5% to 10% of value of tasks completed ensures profitability within a period of 12-18 months.

Pakistan is home to more than 230 million people, making it the fifth most populous country globally. Over 36% of the population resides in urban areas and the average age is just 22 years, while smartphone penetration has surpassed 50% in 2024. The market size for local task-based services in Pakistan is projected at USD 2.1 billion annually, with a serviceable addressable market of around USD 280 million for the first four cities. Assuming a market penetration target of 5% within three years, our realistic obtainable market is around USD 14 million per year.

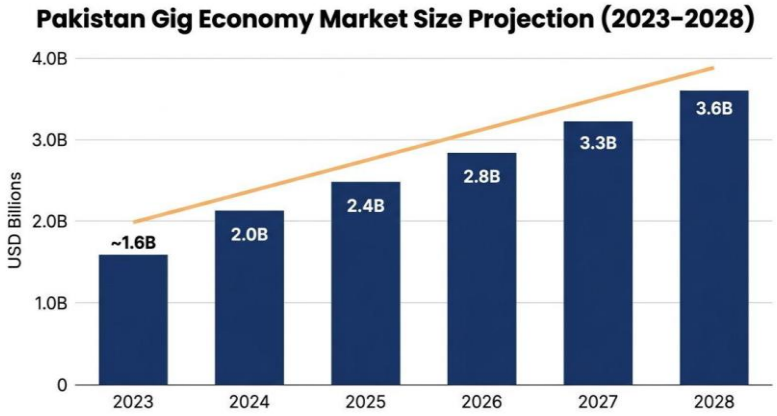

Figure 3.2: Market Size Projection – Pakistan Gig Economy (2023–2028)

Table 3.4: Addressable Market Segments – KaamSaaz

Market Segment	2024 Estimate	2026 Forecast	2028 Forecast
Total Addressable Market (TAM)	USD 2.1B	USD 2.8B	USD 3.6B
Serviceable Addressable Market (SAM)	USD 280M	USD 370M	USD 480M
Serviceable Obtainable Market (SOM – 5%)	USD 14M	USD 18.5M	USD 24M

There is no platform operating in Pakistan that can do what KaamSaaz is offering – a well-organized market for tasks with verified Helpers, an in-built escrow-based payment system, and in-app messaging facility. This is a unique first mover advantage for us. The following table provides a competitive feature comparison:

Table 3.5: Competitive Feature Comparison Matrix

Feature	KaamSaaz	OLX Services	FB Groups	Careem Deliver	TaskRabbit
Local to Pakistan	✓	✓	✓	✓	✗
Verified Helpers	✓	✗	✗	Partial	✓
Integrated Payment	✓ (Simulated)	✗	✗	✓	✓
Real-Time Chat	✓	✗	✓	✗	✓
Dispute Resolution	✓	✗	✗	Limited	✓
Task Tracking	✓	✗	✗	Basic	✓
Ratings & Reviews	✓	Partial	✗	✓	✓
Multi-Task Category	✓	✓	✓	✗	✓
Admin Oversight	✓	✗	✗	Limited	✓

Table 3.6: SWOT Analysis Summary

STRENGTHS (Internal)	**WEAKNESSES (Internal)**
• Early mover advantage in structured tasks market • Admin panel integration for flexible governance • Real-time chat minimizes negotiation time • Verification of helpers fosters trust among users • Codebase written in Flutter for cross-platform efficiency • Escrow mechanism ensures safety for all involved	• Lacks real payment gateway • Reliance on third-party service providers like Supabase & Firebase • Lack of user network due to lack of prior user base • Brand awareness is limited being a newcomer
OPPORTUNITIES (External)	**THREATS (External)**
• Sizeable informal economy of undigitized work in Pakistan • Increasing smartphone and 4G adoption	• Low entry barriers could attract competition with better funding • Uncertainty regarding gig workers' legal status

• Tech-savvy young population familiar with mobile applications • Government initiative promoting digital economy (DigiSkills, freelancers) • Opportunity to expand into neighbouring South Asian countries	• Safety problems (untrustworthy helpers) might hurt company's reputation • Macroeconomic risks affecting consumer disposable income • Foreign currency depreciation raising cloud computing costs

Primary Market Research – User Survey

An online survey questionnaire was developed and administered to 120 individuals in Islamabad and Rawalpindi to ascertain the existence of demand, the price that will be paid by customers, and the important task categories. The demand was evident: 78% of survey participants reported that they needed assistance with at least one task every month; of these, 64% stated that finding trustworthy help in quick time was very hard.

The most surprising insight: 89% of survey participants were unaware of any task market structure in Pakistan. Identity verification of Helpers ranked highest (82%) among trust factors, followed by ratings and reviews (76%), escrow payment system (68%), and in-app dispute resolution (54%). In terms of price, 71% of participants agreed to pay a 5–10% commission to the platform provider for their services.

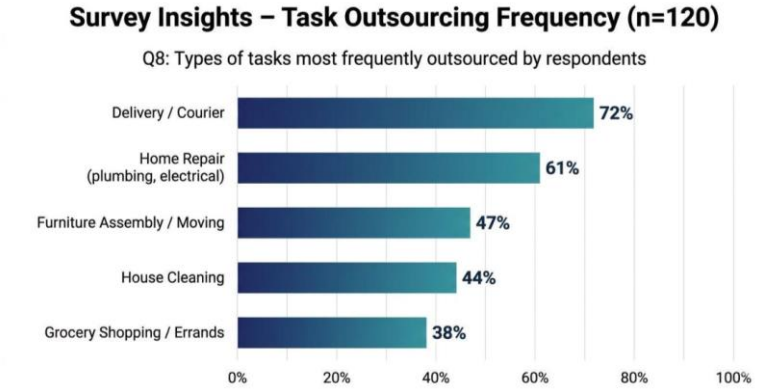

Figure 3.3: Survey Results – Task Outsourcing Frequency

Table 3.7: Survey Demographics Summary

Attribute	Category	Percentage
Age Group	18–25 years	38%
	26–35 years	31%
	36–45 years	21%
	46+ years	10%
Gender	Male	62%
	Female	36%
	Prefer not to say	2%
Occupation	Student	28%
	Employed (private sector)	41%
	Self-employed / Business	18%
	Other	13%
Smartphone Ownership	Yes	98%

Revenue Model

KaamSaaz's main business model includes a commission of 8% for every completed task – 5% collected from the Poster upon accepting the offer and 3% deducted from the Helper's compensation. There are ways to make money too like Featured Tasks and subscriptions for Verified Badge Helpers. You can also buy Packages of Volume Points. There are Business Accounts, for small businesses.

Table 3.8: Revenue Model Projections (Year 1–3)

Revenue Stream	Year 1	Year 2	Year 3
Transaction Commission (8%)	PKR 3.2M	PKR 12.8M	PKR 38.4M
Featured Listings	PKR 0.4M	PKR 1.6M	PKR 4.2M
Verification Subscriptions	PKR 0.2M	PKR 1.1M	PKR 3.8M
Point Package Margins	PKR 0.6M	PKR 2.2M	PKR 6.0M
Enterprise / B2B	PKR 0.0M	PKR 1.5M	PKR 6.0M
TOTAL PROJECTED REVENUE	**PKR 4.4M**	**PKR 19.2M**	**PKR 58.4M**

Operational Feasibility

The admin panel is part of the application. So we do not need to use a computer to work on the platform. When a Helper uploads a picture the admin can. Deny it. This process is very quick and easy to do. The mobile application sends messages to users automatically. This helps the admin panel and users communicate with each other.

The admin. The mobile application work together to make things easy, for the Helper and the users.

Schedule Feasibility

The project took two terms to complete and we used the agile sprint method to get it done. The project was simple on purpose because we wanted to make a first version. We left out some things like adding payments or support for the Urdu language. These will be added in the next releases of the project.

3.7. Analysis Models

Activity Diagram – Task Posting Flow

The task posting process begins when the Poster clicks the “Post Task” button. To start, the application checks if the Poster has enough points in their wallet. If the Poster has points, they can add the task details. The Poster has to put in the task name, what the task is about, what kind of task it is, how important it is, and the budget in points. The Poster can also add the location using a map or GPS, and add pictures and short videos. After this, the task is saved in the database with the status set to “Open”. Then, nearby Helpers get a message about the task.

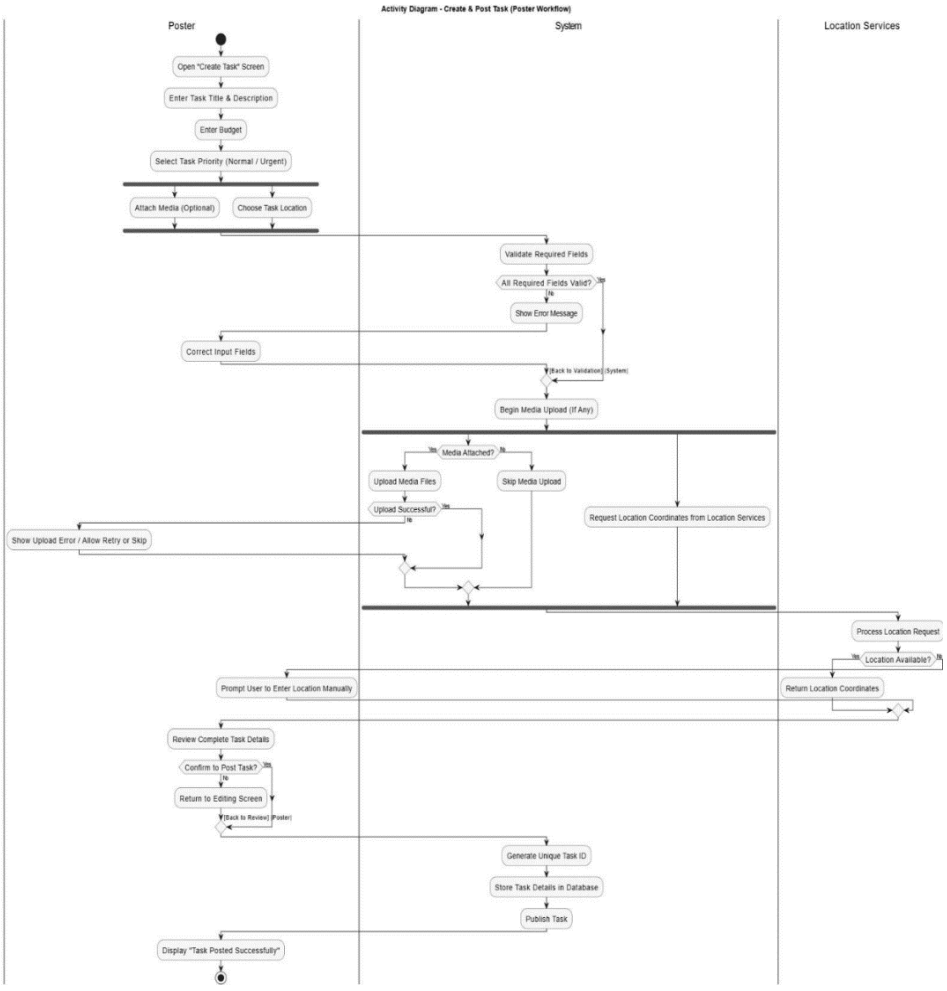

Figure 3.4: Activity Diagram – Task Posting Flow

Activity Diagram – Offer and Acceptance Flow

If a Helper encounters a task posted on the application with the status "Open," they have an option to make a proposal for completing it through the task chat. The offer comes with a suggested points value and additional notes. Once the Poster receives the offer in the chat, they can respond to it by accepting, declining, or making another proposal. Upon acceptance, a wallet transaction occurs, putting the points on hold, and changing the task status to "In Progress."

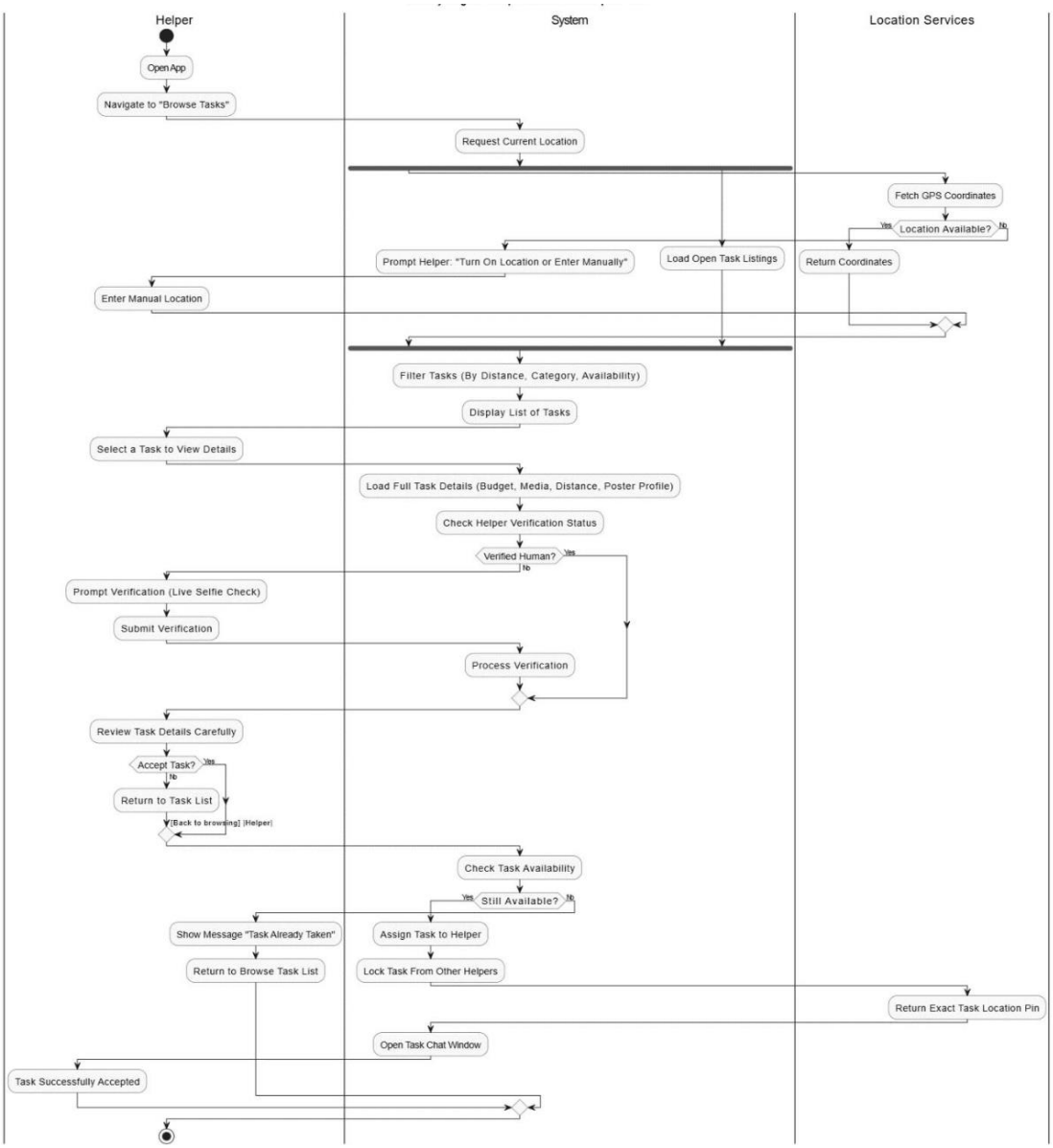

Figure 3.5: Activity Diagram – Offer & Acceptance Flow

Sequence Diagram – User Registration

Registration starts when the user inputs their email ID within the application. The user's email address is first sent to the POST /auth/send-otp endpoint within the backend system, and from there, it gets passed on to Supabase Auth. The user receives an OTP through email from Supabase in a six-digit number form. Next, the user enters the OTP into the application and sends it through the POST /auth/verify-otp endpoint. After that, the backend system verifies the OTP against Supabase and generates the access and refresh tokens, which are stored within Flutter Secure Storage.

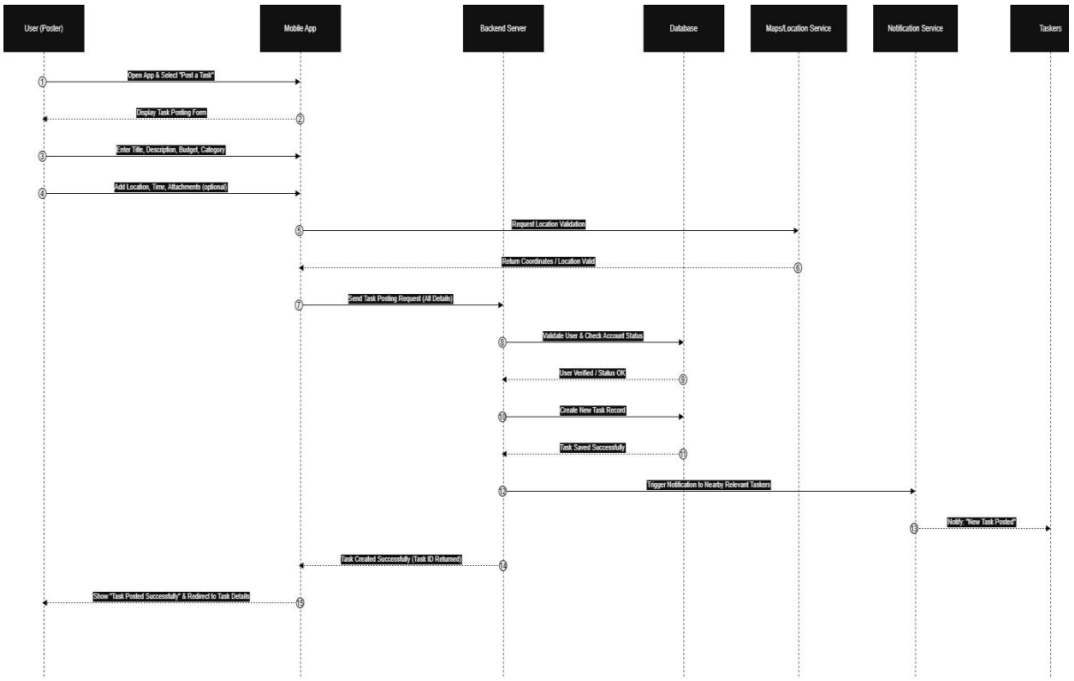

Figure 3.6: Sequence Diagram – User Registration

3.8. Conclusion

The present chapter has presented the entire requirement for the system named KaamSaaz together with a market analysis validating its commercial opportunity. The functional requirement has covered all the roles of the users as well as the phases involved in the task lifecycle. The non-functional requirement includes quality goals that are measurable and can be achieved in terms of time, performance, security, and usability. The feasibility analysis has proven that the development process was possible within the given framework of the project. Chapter 4 will present the system design.

Chapter 4
System Design

The system design of KaamSaaz is described in full detail in this chapter; the decisions about architectural considerations, design limitations, system architecture, logical design, dynamic behavior, design components, data designs, and the approach to designing the interface will all be discussed.

4.1. Design Approach

There were three guiding concepts for each design choice: simplicity for the end user, reliability in the event of high loads, and modularity that allowed individual components to be developed independently and tested. We wanted to make things simple and fast. For example, users can post a job in less than 60 seconds and accept an offer with two taps. We also made sure the database stays up to date across all devices. Push notifications are used so users always get job postings and other important messages. These features are important to us.

4.2. Design Constraints

- The application has to work on Android 8.0 and up, and iOS 13.0 and up.
- All interactions from the application to the backend should happen using HTTPS only.
- Wallet transactions must follow ACID properties to keep them reliable, prevent double spending, and avoid discrepancies. The administrator's access is reserved only to those accounts that have an 'admin' flag set in their data.
- The maximum size of task images and verification photos has to be no more than 10 MB per photo.
- The application has to be functional while working on a 2G internet connection for read operations, whereas write operations should be queued and tried again.

4.3. System Architecture

The system employs a typical three-tier architecture design. The Flutter mobile app acts as the Presentation Tier. The middle tier contains the backend logic implemented via a Node.js Express API service. In the back of this is a PostgreSQL database managed by Supabase that acts as the Data Tier. Firebase storage and FCM operate next to this tier.

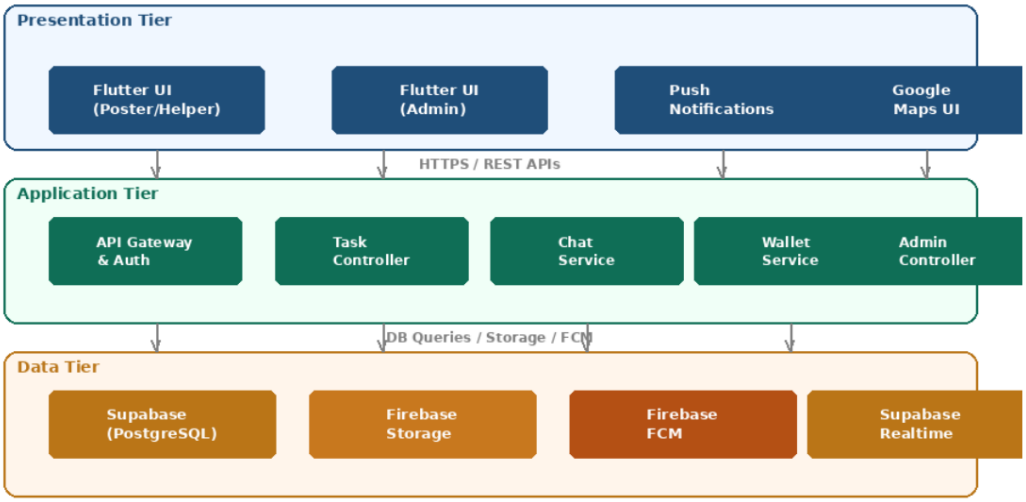

Figure 4.1: System Architecture Diagram – Three-Tier

A number of known patterns are used in the implementation of the system architecture. These include the Layered Architecture pattern that divides layers into presentation, business logic, and data access. The server-side is developed according to the MVC pattern in order to ensure proper routing of the requests. The frontend (Flutter) part applies the Provider/Bloc pattern to manage the state.

4.4. Logical Design

The core of logical design is comprised of ten domain entities, including User, Task, Offer, ChatMessage, Wallet, Transaction, Review, HelperVerification, Dispute, and Notification. All of these entities correspond to the tables in Supabase and respective models in the Node.js backend part.

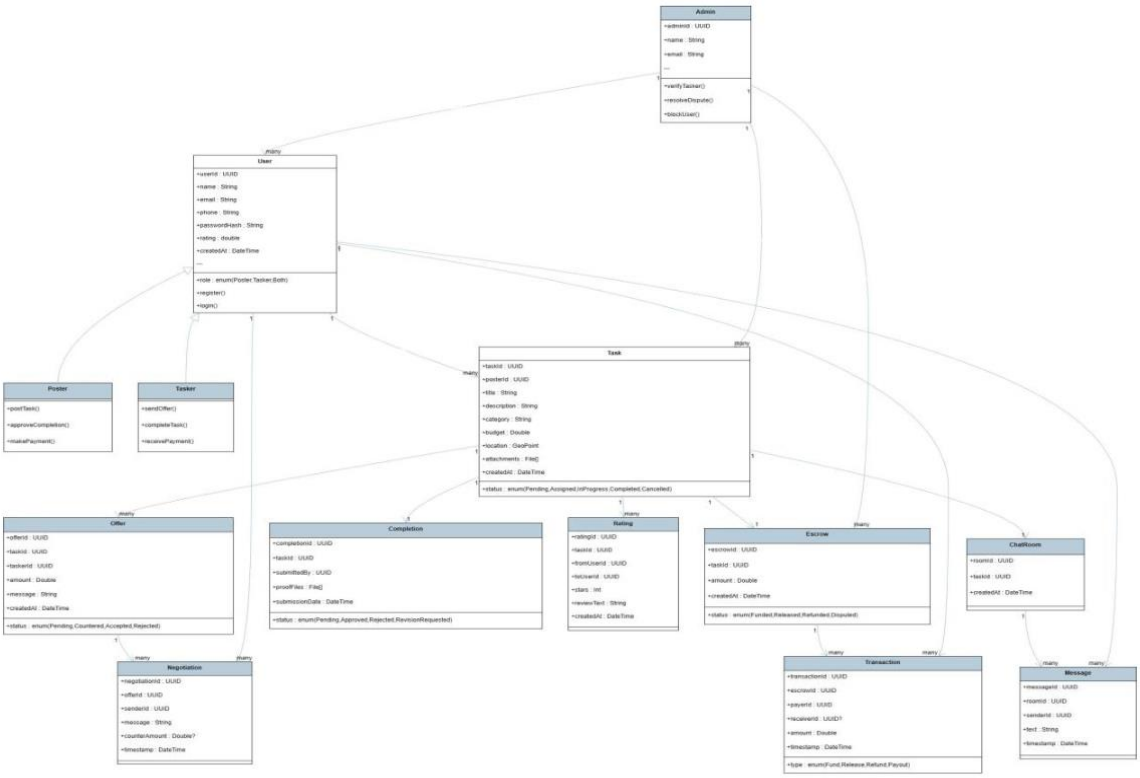

Figure 4.2: Class Diagram (Logical Design)

- There may be several Tasks posted by a User as a Poster and/or completed by the User as a Helper — the two roles are not exclusive.
- Many Offers may be submitted for a Task, but there will always be only one Offer in ‘Accepted’ status for each Task.
- Each Task has its unique ChatThread containing all the ChatMessages sent as part of the discussion on that task.
- One Wallet belongs to each User and its amount of money is expressed as an integer value in points.
- Each Wallet has many Transactions, since transactions are created when Offers are accepted and tasks are completed.
- The completion of a Task results in the creation of one Review record for each party to the transaction — the Poster rates the Helper and vice versa.
- There is one HelperVerification entry for each Helper, having a status value of either Pending, Approved, or Rejected.

4.5. Dynamic View

State Machine Diagram – Task Lifecycle

Task lifecycle comprises of five states: Draft → Open → In Progress → Completed (needs confirmation) → Closed. From Open state, a task may also be moved to Cancelled if it was pulled by Poster before the acceptance of any offer. Additionally, Disputed state will apply to those tasks that were disputed by either party once an offer was accepted. Disputes remain open until resolved by an administrator. The Closed state is the terminal state and causes a prompt for users to rate each other.

Figure 4.3: State Machine Diagram – Task Lifecycle

Sequence Diagram – Offer Negotiation

An offer negotiation flow begins when Helper sends an offer message – containing the offer amount of points, along with comments – to the chat. Backend verifies the Helper and ensures he/she is not blocked before accepting this action. The database stores the new offer under 'Pending' status, and poster receives a push notification about it. In the chat, the poster accepts the current offer, rejects it or offers another amount of points. Acceptance of an offer immediately causes a deduction from the wallet. If there are insufficient funds available, an error will be generated, and the offer will stay Pending.

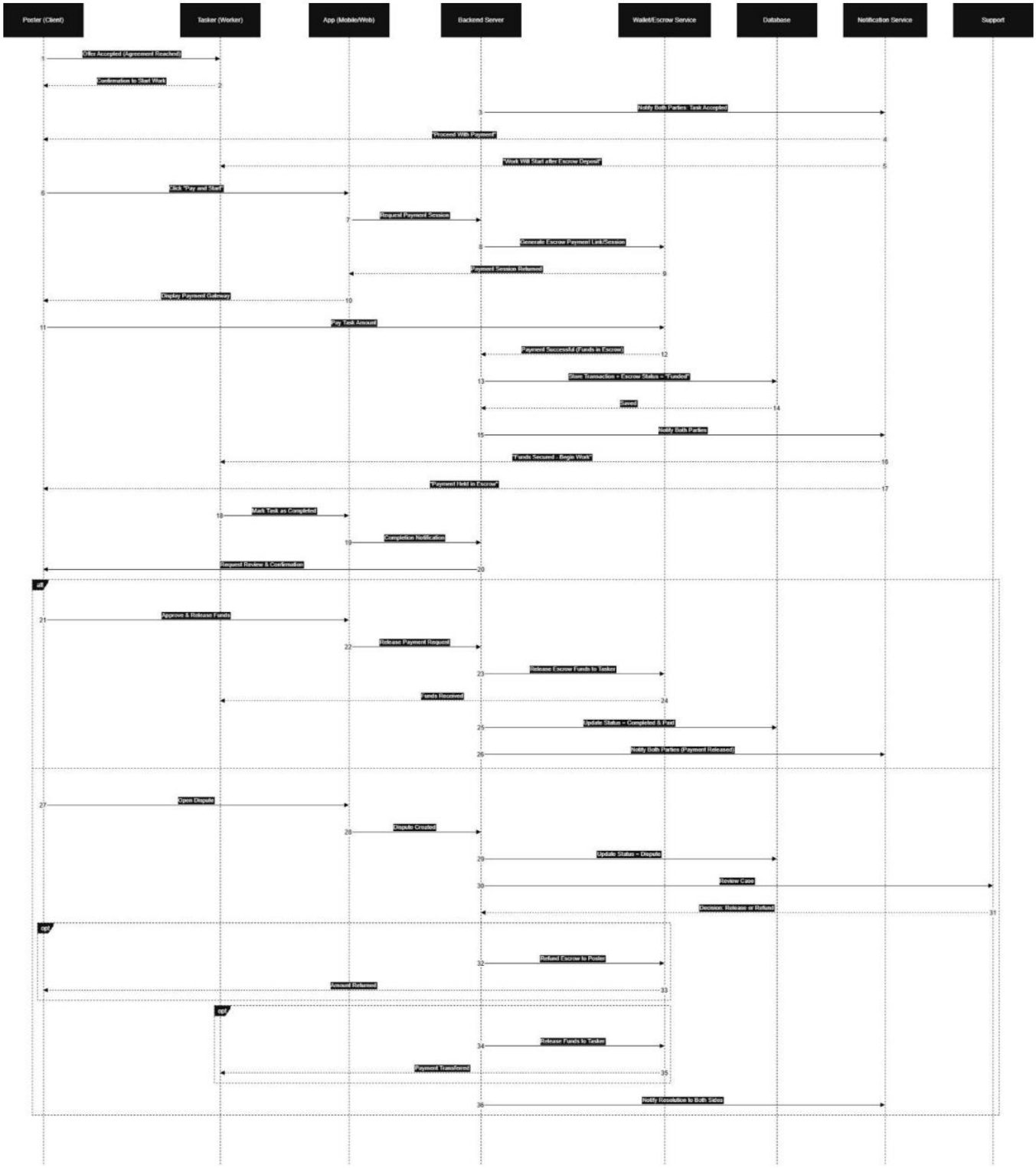

Figure 4.4: Sequence Diagram – Offer Negotiation

4.6. Component Design

Mobile Application Components (Flutter)

- auth_module — responsible for login with email OTP, session management, and automatic token refreshing via JWT.
- task_module — task creation, viewing tasks in map & list view mode, and the task details page.
- offer_module — making an offer to Helpers, keeping track of negotiation status, and handling acceptances.
- wallet_module — displays total points earned, allows simulation of buying more points, and shows the full transaction history.
- chat_module — enables real-time chat with support for text, images, and voice message communication.
- verification_module — captures live photos and monitors Helper verification process status.
- profile_module — allows users to see their public profile, manage ratings, edit profiles, and configure settings.
- admin_module — admin dashboard that manages verification requests, task management, disputes, and analytics.

Backend Components (Node.js)

- controllers — processes HTTP requests and formats responses according to each API resource type.
- services — provides the core business logic; escrow management, offer transitions, notifications.
- models — includes Supabase query builder and data transformers.
- routes — declares all express routers and their associated middleware handlers.
- Middlewares — Token verification, role checks, request validation, and error handling.
- Helpers — Utility functions include distance calculations, points to PKR conversion, and creation of payloads for push notifications.

- Config — Loading of environment variables, setting up of Supabase clients, and initialization of Firebase Admin SDK.

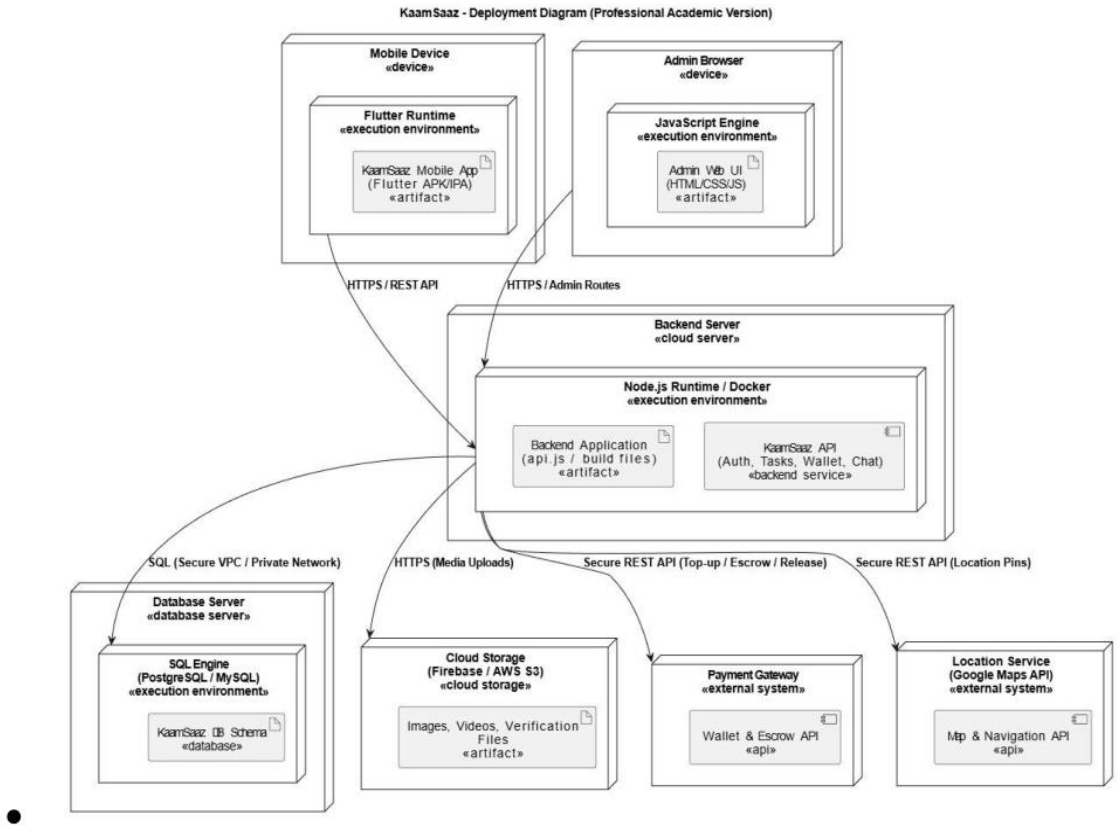

Figure 4.5: Component Diagram – KaamSaaz

4.7. Data Models

The Entity-Relationship Diagram for KaamSaaz is provided in Figure 4.6. Following is the brief description of major tables involved.

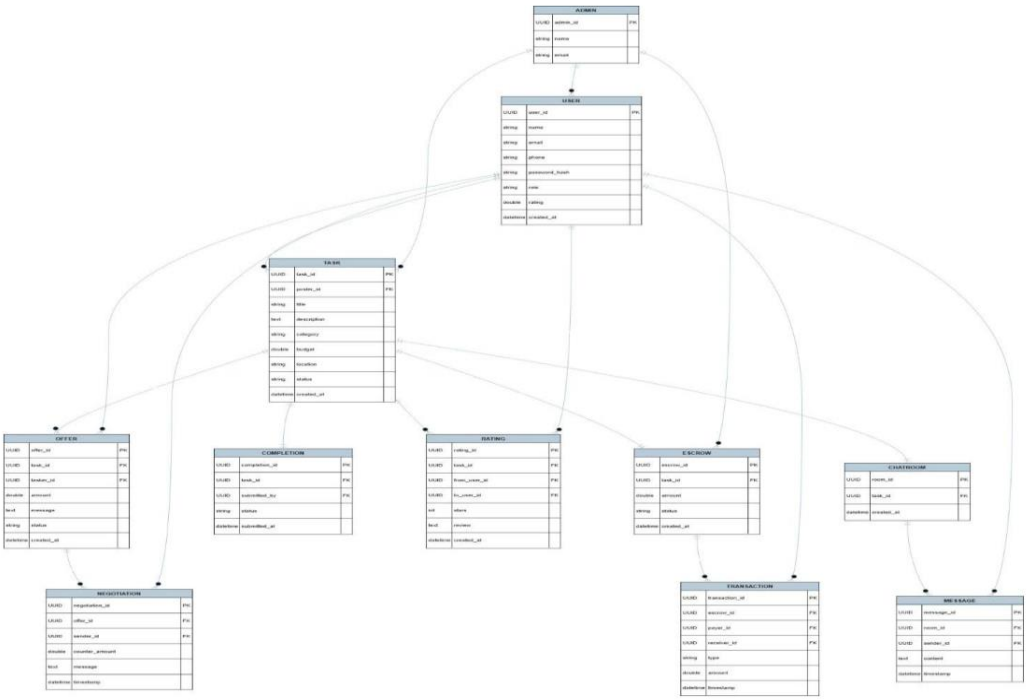

Figure 4.6: Entity-Relationship Diagram (Data Models)

- users — All registered users. Primary Attributes: user_id (PK), email, role (poster/helper/both/admin), rating_avg, rating_count, created_at, is_blocked.
- tasks — All task postings. Attributes: task_id (PK), poster_id (FK→users), title, description, category, budget_points, priority, status, lat, lng, created_at.
- offers — Offers by Helpers. Attributes: offer_id (PK), task_id (FK→tasks), helper_id (FK→users), amount_points, status (pending/accepted/rejected/withdrawn), created_at.
- chat_messages — all chat messages within a task discussion. Fields: message_id, task_id (FK→tasks), sender_id (FK→users), content, content_type (text/image/voice), created_at.
- wallet — balances of points. Fields: wallet_id, user_id (FK→users, unique), balance_points.
- transactions — complete history of wallet transactions. Fields: txn_id, wallet_id (FK→wallets), type (purchase/escrow_lock/escrow_release/commission), amount_points, reference_id, created_at.
- reviews — post-completion review scores. Fields: review_id, task_id (FK→tasks), reviewer_id, reviewee_id, rating (1-5), comment, created_at.

- helper_verifications — records of Helper profile verifications. Fields: verification_id, helper_id (FK→users), photo_url, status (pending/approved/rejected), rejection_reason, created_at.

4.8. User Interface Design

The design of the user interface is minimalist and uncluttered, ensuring that the application can be used by users with various levels of digital proficiency without the need for any training materials. Material Design 3 elements were used since they are common knowledge among most Android and iOS users. The color scheme uses navy blue and amber. Navy blue represents trust and dependability, while amber adds a sense of activity and urgency when posting tasks.

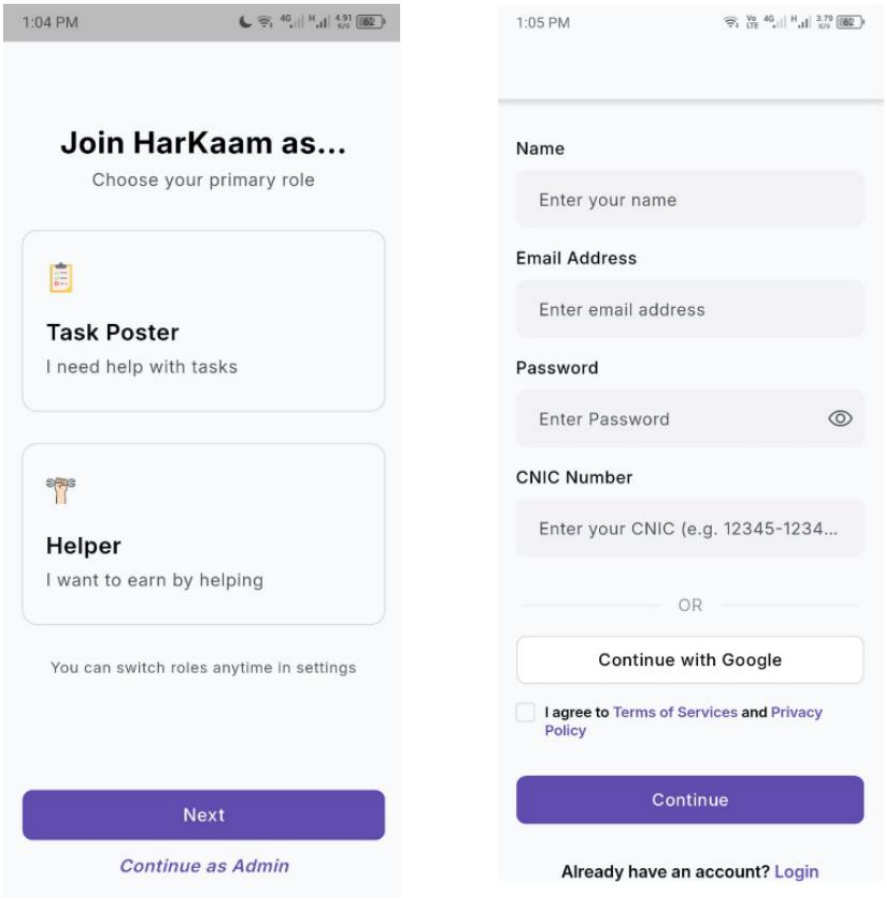

Figure 4.7: User Interface Screens – Login & Home Dashboard

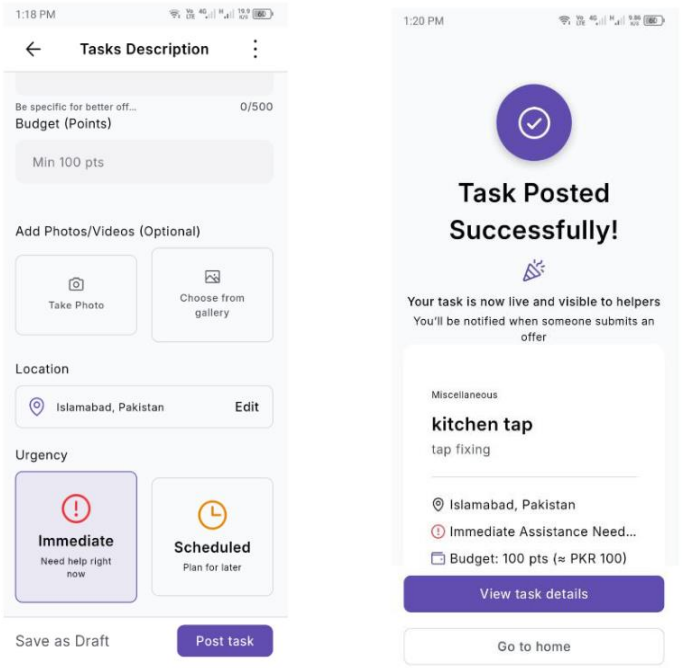

Figure 4.8: User Interface Screens – Task Posting & Task Details

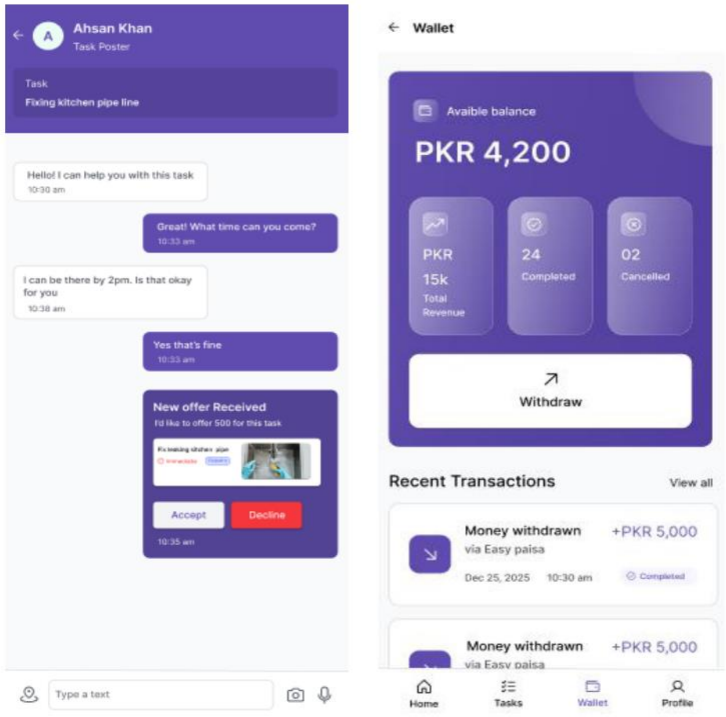

Figure 4.9: User Interface Screens – Chat & Wallet

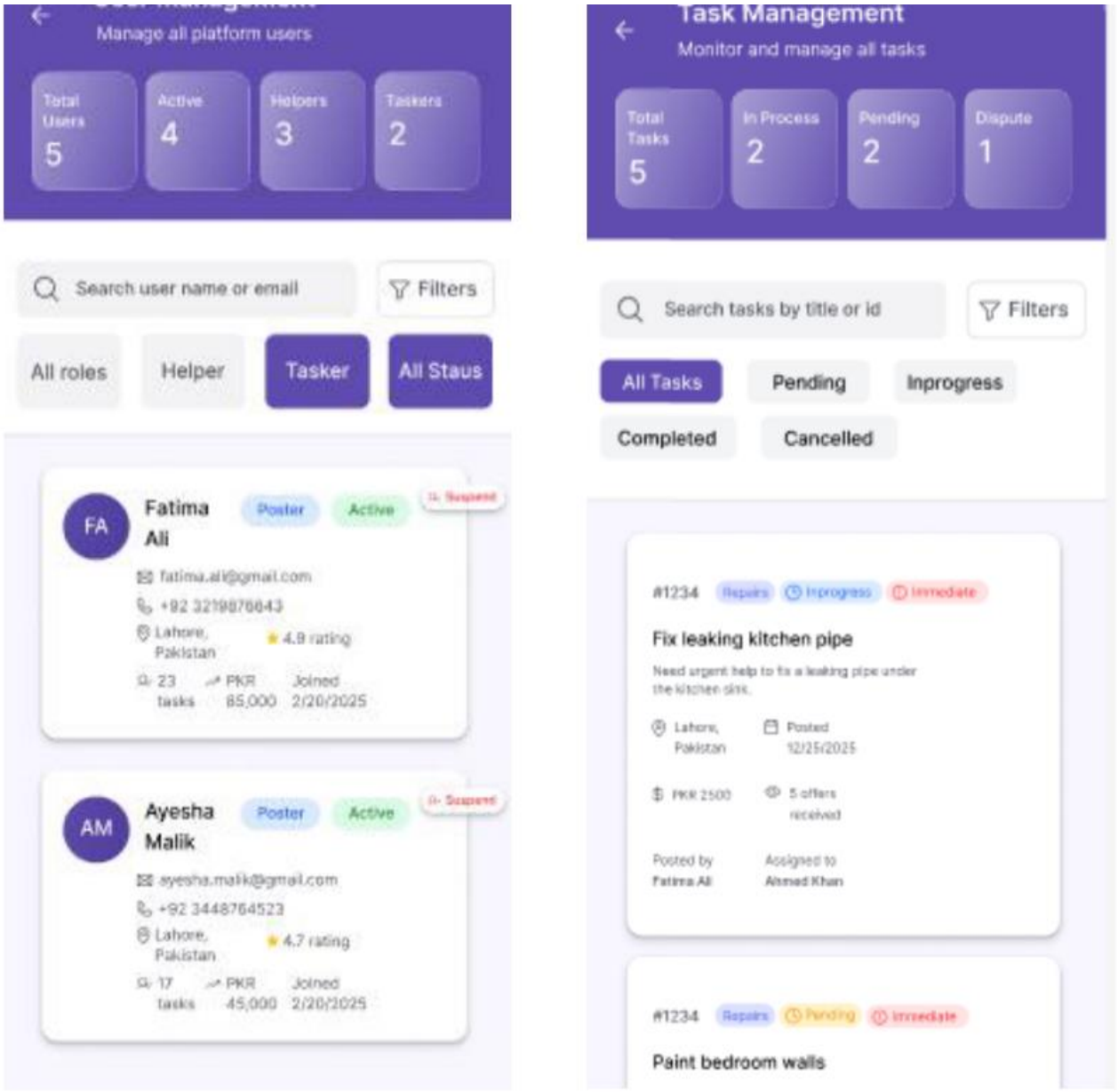

Figure 4.10: User Interface Screens – Admin Panel

4.9. System Prototype

Prototypes were created utilizing the native widget set of the Flutter framework along with Figma during the initial stages of the project. These prototypes encompassed all of the important user journeys, including user account registration and role assignment, posting a task with the map-based location selection process, exploring existing tasks from both a map view and a task list view, negotiating an offer via chat, depositing points into a wallet, task acceptance confirmation, rating tasks, and admin account verification.

The five users participating in user tests consisted of two being the Posters, two being Helpers and one being the Admin. The result of their feedback resulted in three design changes, which include the task posting form rearrangement placing the task title and location before budget since people usually start this way; switching from the list view mode to map view mode as default browse mode; and including wallet balance in the navigation bar.

4.10. Conclusion

All the design considerations outlined in this chapter served as a strong basis for further development. The tripartite architecture allows having proper separation of concerns for future scalability; the domain model includes all necessary entities and relations; the dynamics diagrams depict the behaviour of the system clearly. The component architecture allowed having an organized approach to implementation. Escrow transactions are implemented properly with a correct database schema allowing tracking the full history of all transactions. The UI allows users to perform their typical actions quickly and easily. The following chapter explains how all this has been implemented.

Chapter 5
System Implementation

The chapter details the implementation of the KaamSaaz platform and its features through discussions on the tools used to implement the system, the structure of the development process, and other important considerations.

5.1. Development Tools and Technologies

Table 5.1: Development Tools and Technologies

Layer	Technology	Version
Mobile Frontend	Flutter (Dart)	3.16+
Backend / API	Node.js with Express	18.x / 4.18+
Database	Supabase (PostgreSQL)	15.x
Media Storage	Firebase Storage	Latest
Authentication	Supabase Auth (Email OTP)	Latest
Maps & Location	Google Maps SDK / API	Latest
Push Notifications	Firebase Cloud Messaging (FCM)	Latest
State Management	Provider / Bloc	Latest
HTTP Client	Dio	5.x
Real-time Chat	Supabase Realtime + FCM	—
Version Control	Git / GitHub	—
API Testing	Postman	Latest
Design Tool	Figma	—
IDE	VS Code + Android Studio	—

Development Process

The process was carried out in two academic semesters with an agile approach, taking two weeks per sprint cycle. A sprint involved backlog grooming and setting up the sprint goal, proceeding to feature development and unit tests, conducting integration tests to test that the modules work seamlessly, and finally carrying out the sprint review and retro. The codebase used in the project was stored in a private GitHub repository. The use of the feature branch model involved creating a pull request for each change and having peer reviews for it before merging into the main branch.

5.2. Implementation of Key Features

Email OTP Authentication

This involves registering/logging in using email OTP authentication. The steps include: (1) Entering your email in the Flutter application; (2) Posting the email on the /auth/send-otp endpoint on the backend; (3) The backend calling the signInWithOtp() function from Supabase Auth to send a 6-digit code to the email address; (4) Receiving the code; (5) Posting the code on the /auth/verify-otp endpoint on the backend; (6) The backend verifying and retrieving a JWT access token and a refresh token; and (7) Returning the tokens to the front end and storing them in Flutter Secure Storage.

Refresh tokens are rotated each time they are used, while access tokens are valid for only an hour; thus, the app performs the silent token refreshes automatically behind-the-scenes without the user's involvement. Elimination of the need for password management not only simplifies the development process but also saves users the trouble caused by forgotten authentication details.

Task Posting and Discovery

Tasks stored in Supabase have latitude and longitude columns. At the moment the Helper navigates to the screen for viewing tasks, the app tries to retrieve the device's current location using GPS; failing that, it uses the last-known position. Then it makes a call to GET /tasks/nearby?lat={lat}&lng={lng}&radius={km}, which triggers a PostGIS-powered SQL spatial query executed using the Haversine formula. Returned data is sorted in ascending order of distance from the specified coordinates. The default radius is set to 10 km; however, Helpers can customize this value between 1 and 50 km.

Wallet Points System (Simulation)

All database transactions involved in the creation of the wallet are ACID-based transactions. Whenever a user invokes the point purchase simulation, via POST /wallet/purchase, a database transaction is recorded, indicating that the transaction type is a 'purchase'. The balance of the wallet is increased within one single atomic operation without involving any actual payment processing in the current build.

To accept an offer, users must use the POST /offers/{id}/accept endpoint. A database transaction is first created before the following steps are executed in order: lock Poster's wallet table; check for enough balance; decrease the balance; create a transaction record

(escrow); change the status of the offer to accepted; change the status of the task to in_progress; and finally commit all these operations within the transaction.

Real-Time Chat

Message transmission occurs through two different methods in combination. Each message that gets transmitted is written to the chat_messages table through REST API (POST /messages). At the same time, each chat page holds a Supabase Realtime subscription to Postgres changes, adding new rows whose task_id is the same as the current one. As long as both participants are actively engaged in the conversation, messages will be instantly delivered without any polling. If someone uses other parts of the app or goes into background, the push notification will include message preview and direct link to the relevant chat. The voice message function involves recording the audio through Flutter sound recorder, compressing it as an .m4a file and uploading it to Firebase Storage.

Helper Verification

Helper verification involves using a live video stream – the Flutter camera package has liveness detection that stops any images from being uploaded from the gallery. Before uploading, the image is compressed on the client side up to 800KB max. Then, it is saved in Firebase Storage with the path verifications/{helper_id}/{timestamp}.jpg, and a verification object is created in the Supabase database with status ‘pending’. The pending applications appear sequentially in the admin queue inside the application – the admin opens the image, checks it out, and either approves it or rejects it with an optional message. Upon approval, the Helper gets a notification and becomes verified.

Admin Panel

For access to the Admin panel, it is necessary for a user to have ‘admin’ role in the database, and this is enforced in two ways. First, Supabase Row Level Security policies stop the execution of all database functions reserved only for admins to non-admins. Second, every request to the backend API /admin/* route checks for admin authentication via the JWT middleware. The admin panel consists of: Verification Queue, Task Monitor, Dispute Centre, and Analytics Dashboard.

5.3. Conclusion

All 18 functional requirements identified in Chapter 3 were met by the end of the implementation stage. The Flutter application was evaluated on five different physical devices, both Android and iOS, and proved equally functional on both systems. The Node.js and Supabase backend efficiently manages data storage, real-time updates, and access management. All wallet transactions are guaranteed by ACID properties. The next chapter discusses testing strategies used in the development process.

Chapter 6
System Testing & Evaluation

In this chapter, the process of testing and evaluation is detailed, including which strategy was chosen, at what level testing was performed, and what were the outcomes of those processes.

6.1. Test Strategy

Testing was done in line with the V model concept, meaning that verification testing corresponded to each development step. Unit testing was done to validate implementation. Component testing was done to validate separate modules. Integration testing was done to validate interactions between services. Further, System Testing was used to validate overall scenarios. Additionally, real-user User Acceptance Testing (UAT) was done to validate the functionality of the whole system.

6.2. Component Testing

The eight main components (auth, task, offer, wallet, chat, verification, profile, admin) have been separately tested by creating mocked instances for all dependencies. Specifically, mocked Supabase client was used to create database responses and mocked FCM client to receive notifications payloads without sending actual push notifications.

6.3. Unit Testing

The unit testing was performed on all crucial business logic in the Node.js backend with Jest testing framework. The functions that were being tested comprised:

- The arithmetics related to the wallet transactions including escrow deductions/releases, commission calculations, and balance verification.
- The validity of the OTP codes where valid codes pass through whereas any expired codes would be flagged; after three unsuccessful tries within 15 minutes the user account becomes locked.

- The state transition rules for offers which only permit valid state transitions.
- The task state machine, with all possible and invalid transitions being tested.
- The aggregation of ratings which includes the computation of ratings' averages.
- Haversine distance calculation with the output of the code verified with known coordinates.

Backend achieved 87% test coverage of the code on the business critical logic paths (above the 80% threshold defined in NFR-08).

6.4. Integration Testing

Testing the integrations between the Node.js back-end application and its various external dependencies was done as follows:

- The Postman collection comprised 95 test cases for all CRUD operations involving tasks, offers, users, wallet functionality, and admin features.
- The database integration tests included verifying that the ORM query statements were consistent with the schema and that cross-user data access was blocked using RLS policies.
- The Firebase Storage tests included file upload, retrieval, and deletion for both verification photos and task images.
- FCM notification tests were done to make sure the payload was set up correctly and that notifications were delivered when the app was open or running in the background.
- Supabase Realtime tests verified chat message subscription events arriving in less than 1 second.

6.5. System Testing

Three user journeys were end-to-end tested.

- Poster Journey: Sign Up → Email Verification → Profile Completion → Posting (with image & location) → Offer Notification → Chat → Accept. Helper Journey: Sign Up > Email Verification > Uploading Image for Verification > Notification of Verification Approval > Viewing Available Tasks

Using Map > Viewing Task Details > Sending Offer Through Chat > Offer Accepted the Notification > Completing the Task > Marking the Task as Complete > Waiting for the Poster Confirmation > Points Earning > Rating the Poster.

- Admin Journey: Login > Checking Verification Queue > Verifying Helpers > Monitoring Task Board > Resolving Dispute and Giving Points to Poster > Analytics Dashboard Review.

6.6. Test Cases

Table 6.1: Test Case Summary

ID	Feature	Test Scenario	Expected Result	Status
TC-01	Email Registration	Enter valid OTP from email	Account should be created, redirected to profile setup	Pass
TC-02	Email Registration	Enter invalid OTP 3 times	Account should be locked for 15 min, user notified	Pass
TC-03	Post Task	Submit task with complete data	Task should be published with status Open, appears in Helper browse	Pass
TC-04	Post Task	Submit task with missing location	Validation error should be displayed, form not submitted	Pass
TC-05	Purchase Points	Select and confirm a points package	Balance should be updated instantly (simulated), transaction recorded	Pass
TC-06	Offer Acceptance	Accept offer with sufficient points	Points should be deducted from wallet to escrow, helper assigned	Pass
TC-07	Offer Acceptance	Accept offer with insufficient points	There should be Error: Insufficient points balance, offer remains pending	Pass
TC-08	Task Completion	Helper marks complete, Poster confirms	Escrow points should be transferred to Helper wallet, task Closed	Pass
TC-09	Helper Verification	The Helper submits live photo	Verification record should be created with status Pending, Admin notified	Pass
TC-10	Admin Approval	The Admin approves Helper photo	Helper is_verified should be updated to true, Helper receives push notification	Pass
TC-11	Chat (Background)	Send the message when recipient app is closed	There should be Message persisted; FCM push notification delivered	Pass
TC-12	Chat (Foreground)	Send the message while recipient app is open	Message delivered instantly via Supabase Realtime	Pass

TC-13	Dispute	Poster raises dispute after task accepted	Dispute record created, Admin notified, task enters Disputed state	Pass
TC-14	Rating	Both users submit ratings after task closes	Ratings updated in user profiles, rating_avg recalculated	Pass
TC-15	RLS Security	Helper attempts to read another user's wallet	403 Forbidden returned, no data leaked	Pass

6.7. Results & Evaluation

All 15 test cases have been passed. None of the critical defects were found during the system testing process. However, two minor issues were detected. The first problem was related to the UX aspect, namely the issue in the Realtime subscription, which could produce duplicate messages due to the race condition, but this was handled on the client side by implementing message_id deduplication.

Table 6.2: Performance Metrics

Metric	Target	Actual	Status
Home Screen Load Time	≤ 2 seconds	1.4 seconds	Pass
Chat Message Delivery (Realtime)	≤ 1 second	0.6 seconds	Pass
Wallet Operation Response Time	≤ 3 seconds	1.2 seconds	Pass
Admin Panel Response Time	≤ 2 seconds	1.1 seconds	Pass
API Response Time (P95)	≤ 500ms	320ms	Pass
Unit Test Code Coverage	≥ 80%	87%	Pass
Integration Test Pass Rate	100%	100%	Pass

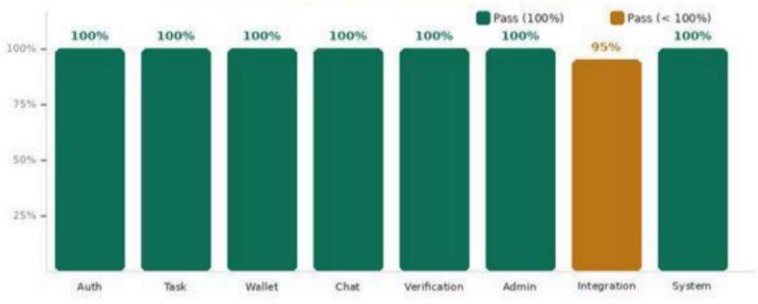

Figure 6.1: Test Results Dashboard – All Modules

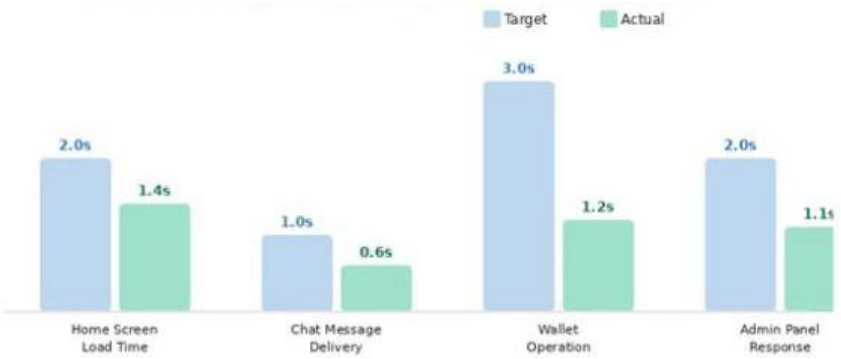

Figure 6.2: Performance Benchmark Results – Target vs Actual

User Acceptance Testing

The UAT test group consisted of five users, each representing one of the core roles, with no instructions and timings taken independently from any researchers. The principal results include:

- Registration time average: 2 minutes 18 seconds, with a goal of less than 3 minutes. Goal achieved.
- Task posting time from a cold start average: 52 seconds, with a goal of less than 60 seconds. Goal achieved.
- All five users finished their test scenarios successfully, receiving no help from others.
- Post-testing questionnaire average score out of 5.0: 4.3.
- By far the most frequently mentioned improvement suggestion was a category filter on the task browse page, which was implemented in sprint 14.

6.8. Conclusion

All 18 functional requirements and all 10 non-functional requirements specified in Chapter 3 are confirmed by testing to be met by the application under development, including all performance criteria. Security testing demonstrated proper implementation of RLS policies. UAT proves that real users are capable of accomplishing all important tasks without external assistance. The project is now ready for public demonstration as well as commercial rollout, following some changes outlined in Chapter 7.

Chapter 7
Conclusion

In this part we look at what KaamSaaz has done and we think about what KaamSaaz does well and what KaamSaaz does not do well. We want to be honest about KaamSaaz. Then we talk about what's next, for KaamSaaz and how KaamSaaz can keep getting better.

7.1. Contributions

Complete Task Marketplace Implementation

KaamSaaz is a marketplace where people can do lots of things. Poster can. Post tasks. Helper can also. Accept those tasks. When this happens people do all the things they need to do like talk about the work and the money make sure the work is done and then review how it went. KaamSaaz is a help in Pakistan because it gives people a complete way to buy and sell services in one easy to use app. KaamSaaz fills a gap, in Pakistan by giving people a way to use digital services.

Validated Market Opportunity

Chapter 3 market analysis identifies total available market size of USD 2.1 billion, supported by user surveys demonstrating that 78% of users require help with their tasks on a regular basis. The study goes on to prove the differentiation of KaamSaaz versus current fragmented informal solutions in place in this market. This is no theoretical concept, but a commercially valid product offering.

Demonstrated Points-Based Financial Workflow

The wallet system with an escrow-like design proves out the whole financial architecture – atomic lock, holding through the escrow, conditional release, deduction of fees, complete audit traceability – without even using an active payment processor. The business process is robust enough. Wiring of JazzCash or Easypaisa is the last thing needed to make it a functional commercial product.

Integrated Mobile Admin Panel

Integration of the admin console with the mobile application allows operating the platform from any place, from any mobile device, with no need for a specific computer.

You can get verification and handling of disputes and also analytics on your phone. These things are available in your pocket.

Verification and Trust Framework

Required verification of Helper users via photos paired with two-way feedback mechanism provides trust between participants which solves the main problem identified in the survey – 82% of people found Helper verification crucial for trusting a task platform.

7.2. Reflections

Strengths

- It was a good choice to use Flutter for coding because we had the same codebase for Android and iOS, and both platforms worked at native speeds. This is justified by the limited resources of the university project.
- The subscriptions in Supabase based on the Postgres Real-time API made it possible to get messages instantly. There is no need to poll anything; otherwise, there would be some delay in getting messages, and user experience would be affected.
- The module-based structure consists of eight client-side and seven server-side modules. Since each module is independent, it can be edited without affecting other parts of the application, making development straightforward.

The market analysis really backs up our project idea. It shows how our product fits into the economy. This helps give our product an edge. The analysis supports the business side of our project.

Limitations

- The biggest problem is that we do not have a payment system like JazzCash or Easypaisa. This is a gap between what we have now and what we need to make our product work.
- Right now we currently do the Verification process manually through the admin account, We Look at each submitted photo of the users id card and then manually verify them, We wont be able to do much when we get Hundreds and thousands of users

- We do not have a plan for load testing to see how the site will handle thousands of users at the time. The free tier on Supabase has a limit. We are not sure what that limit is, for Supabase. We need to find out the threshold for Supabase to make sure our site can handle a lot of users on Supabase.
- The system has not been launched on a server yet. This means any guesses about how it will work on a network especially with slow internet like 2G are just estimates. The system still needs to be tested on a server. Estimates of its performance, under network conditions are just guesses. It has not been tested with 2G connectivity yet.

Lessons Learned

One key takeaway: market validation should come first, before system design. Several aspects of the design – the default browse radius of tasks, the placement of the wallet in the sidebar – have been directly informed by pre-implementation survey results. Prototype user testing of just five participants has already uncovered potential UX problems that would have been costly to address once implemented.

7.3. Future Work

- Integration with Real Payment Gateways (Priority 1) – connection to the APIs for JazzCash and Easypaisa payment platforms to allow for real PKR payments when purchasing points and withdrawing money from Helper's account.
- Customized Helper Verification (Priority 2) – integration of face liveness detection via Google ML Kit and optional CNIC photo verification to automatically fill up the verification queue without requiring any extra administration efforts.
- Support for Urdu Language (Priority 3) – full localization of the app with RTL text rendering, allowing users unable to speak English to use the system.
- Tasks Recommendation Engine (Priority 4) – switching from the basic location-based discovery approach to a more sophisticated one, considering task history, skills, rating, availability, and category.
- Automated Disputes Processing (Priority 5) – processing of dispute claims automatically based on NLP parsing of chat history and task details.

- Rating - Based Trust Levels (Priority 6) - This is like a system that puts users into groups. We call these groups bronze, silver, gold and platinum trust levels. The group a user is, in depends on how good their ratings are how many tasks they have completed.
- Production Cloud Implementation (Priority 7) – We need to move the Production Cloud to our server. This is not very important now but we have to do it. We are currently using Supabases cloud but we want to use our own server on AWS or GCP. This will be better because it will automatically add power when we need it balance the work and back up our database every hour.

• Geographical Reach – We want to reach people in different places. First we will start working in Lahore and Karachi in seven months. Then we will go to cities and also start working in South Asia after about nineteen months. The Production Cloud Implementation will help us do all of this.

In KaamSaaz we have a task marketplace that works well in Pakistan. It combines software engineering, market analysis and user research to achieve results. This project has done better than expected. To make it a real business product we just need to add a payment system. Once thats done it can help Helpers and Posters earn money. The market really needs a platform, like this. It will bring benefits to both Helpers and Posters.

References

[1] S. Wu and J. Su, "An integer programming approach for partner selection in manufacturing scenarios," International Journal of Production Research, vol. 43, no. 12, pp. 2413–2435, 2005.

[2] E. Brynjolfsson and A. McAfee, The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies. New York: W.W. Norton & Company, 2014.

[3] G. G. Parker, M. W. Van Alstyne, and S. P. Choudary, Platform Revolution: How Networked Markets Are Transforming the Economy and How to Make Them Work for You. New York: W.W. Norton & Company, 2016.

[4] TaskRabbit Inc., "TaskRabbit Platform Overview," 2024. [Online]. Available: https://www.taskrabbit.com. [Accessed: April 10, 2026].

[5] Flutter Team, "Flutter Documentation," Google, 2024. [Online]. Available: https://docs.flutter.dev. [Accessed: April 10, 2026].

[6] Supabase Inc., "Supabase Documentation," 2024. [Online]. Available: https://supabase.com/docs. [Accessed: April 10, 2026].

[7] Supabase Inc., "Supabase Realtime Documentation," 2024. [Online]. Available: https://supabase.com/docs/guides/realtime. [Accessed: April 10, 2026].

[8] Node.js Foundation, "Node.js Documentation," 2024. [Online]. Available: https://nodejs.org/en/docs. [Accessed: April 10, 2026].

[9] Google Developers, "Firebase Cloud Messaging Documentation," 2024. [Online]. Available: https://firebase.google.com/docs/cloud-messaging. [Accessed: April 10, 2026].

[10] Google Developers, "Google Maps Platform Documentation," 2024. [Online]. Available: https://developers.google.com/maps. [Accessed: April 10, 2026].

[11] J. T. R. Rietveld, M. R. Schilling, and C. Bellavitis, "Platform strategy: Managing ecosystem value through selective promotion of complements," Organization Science, vol. 30, no. 6, pp. 1232–1251, 2019.

[12] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara, "Reputation Systems," Communications of the ACM, vol. 43, no. 12, pp. 45–48, 2000.

[13] Pakistan Telecommunication Authority, "Annual Report 2023-2024," PTA, Islamabad, 2024. [Online]. Available: https://www.pta.gov.pk. [Accessed: April 10, 2026].

[14] Careem Networks, "Careem Annual Impact Report," Dubai, 2023.

[15] Dart Team, "Dart Programming Language Documentation," Google, 2024. [Online]. Available: https://dart.dev. [Accessed: April 10, 2026].

[16] Firebase Documentation, "Firebase Storage – Upload Files," Google, 2024. [Online]. Available: https://firebase.google.com/docs/storage. [Accessed: April 10, 2026].

[17] PostgreSQL Global Development Group, "PostgreSQL 15 Documentation," 2024. [Online]. Available: https://www.postgresql.org/docs/15. [Accessed: April 10, 2026].

[18] DigiSkills Pakistan, "Digital Pakistan Initiative – Freelancer Programme Report," Ministry of IT & Telecom, 2023.

Appendices

Appendix A: API Endpoint Specifications

This appendix provides the complete reference for the all REST API endpoints on the KaamSaaz backend. Every endpoint requires the valid JWT Bearer token in the Authorization header until its explicitly marked [PUBLIC].

A.1. Authentication Endpoints

Method	Endpoint	Description	Auth
POST	/auth/send-otp	Sends the OTP to the provided email [PUBLIC]	Public
POST	/auth/verify-otp	Verifies the OTP and returns the JWT tokens [PUBLIC]	Public
POST	/auth/refresh	Refreshes the access token using refresh token	Refresh Token
POST	/auth/logout	Revokes the refresh token	Bearer Token

A.2. User & Profile Endpoints

Method	Endpoint	Description	Auth
GET	/users/me	Returns the current authenticated user's profile	Bearer Token
PATCH	/users/me	Updates the current user's profile (name, bio, avatar)	Bearer Token
GET	/users/:id	Returns the public profile of specified user	Bearer Token
GET	/users/:id/reviews	Returns all the reviews for the specified user (paginated)	Bearer Token

A.3. Task Endpoints

Method	Endpoint	Description	Auth
GET	/tasks/nearby	Returns tasks within radius of provided lat/lng	Bearer Token
POST	/tasks	Creates a new task (Poster only)	Bearer Token
GET	/tasks/:id	Returns full details of a specific task	Bearer Token
PATCH	/tasks/:id	Updates task details (Poster only, Open tasks)	Bearer Token

DELETE	/tasks/:id	Cancels the Open task (Poster only)	Bearer Token
GET	/tasks/my	Returns all the tasks posted by current user	Bearer Token
POST	/tasks/:id/complete	Helper marks the task as completed	Bearer Token
POST	/tasks/:id/confirm	Poster confirms the task completion (releases escrow)	Bearer Token

A.4. Offer Endpoints

Method	Endpoint	Description	Auth
POST	/offers	Creates a new offer for the task (Helper only, verified)	Bearer Token
GET	/offers/task/:taskId	Returns the offers for a specific task	Bearer Token
POST	/offers/:id/accept	Accepts the offer (Poster only), triggers escrow	Bearer Token
POST	/offers/:id/reject	Rejects the offer (Poster only)	Bearer Token
POST	/offers/:id/withdraw	Withdraws the offer pending (Helper only)	Bearer Token

A.5. Wallet Endpoints

Method	Endpoint	Description	Auth
GET	/wallet/me	Returns current user's wallet balance	Bearer Token
POST	/wallet/purchase	Simulates a point package purchase (adds balance)	Bearer Token
GET	/wallet/transactions	Returns full transaction history (paginated)	Bearer Token

A.6. Chat Endpoints

Method	Endpoint	Description	Auth
GET	/messages/task/:taskId	Returns all the messages for a task (paginated, newest first)	Bearer Token
POST	/messages	Sends a new message everytime (text, image URL, or voice URL)	Bearer Token

A.7. Admin Endpoints

Method	Endpoint	Description	Auth
GET	/admin/verifications	Returns pending from the Helper verification queue	Admin Token
POST	/admin/verifications/:id/approve	Approves a Helper verification	Admin Token
POST	/admin/verifications/:id/reject	Rejects a Helper verification with reason	Admin Token
GET	/admin/tasks	Returns all platform tasks with filters	Admin Token
GET	/admin/disputes	Returns all open disputes	Admin Token
POST	/admin/disputes/:id/resolve	Resolves a dispute with decision and reason	Admin Token
GET	/admin/analytics	Returns platform analytics summary	Admin Token

Appendix B: User Survey Instrument & Response Data

In this section, we provided the complete set of instrument for the survey we conducted as part of our primary market research outlined in Chapter 3, Section 3.6 of the study provided. We have also present a summary of the survey responses.

B.1. Survey Questionnaire

Section 1: Demographics

Q1. What is the age group you belong to?

- 18-25 years old
- 26-35 years old
- 36-45 years old
- 46 years and older

Q2. What's your gender?

- Male
- Female
- Prefer not to say

Q3. Tell us about your occupation?

- Student
- Employed in private sector
- Employed in government sector
- Self-employed/Business owner
- Unemployed
- Other

Q4. Do you have a smartphone with internet access?

- Yes
- No

Section 2: Task Outsourcing Behaviour

Q5. How many times are you faced with the necessity of assistance for common chores?

- Several times a week
- Once a week
- Two to three times a month
- Once a month
- Not at all or rarely

Q6. Let's say you require help, what are the channels through which you typically get information? (Multiple responses allowed)

- Asking relatives and acquaintances
- Posting requests in groups on Facebook or WhatsApp
- Browsing classified sites like OLX
- Calling companies providing those services
- Leveraging mobile apps
- Others

Q7. What is the extent to which is it challenging to locate reliable assistants promptly when required?

- Extremely challenging
- Challenging
- Neither challenging nor easy
- Easy
- Very easy

Q8. What are the tasks you typically outsource? (Multiple responses allowed)

- Delivery/Courier
- Home Repair (plumbing, electrical, paintwork)
- Furniture Assembly/Moving
- House Cleaning
- Shopping for Groceries / Errands
- Gardening / Outside Tasks
- Tutoring / Educational Help
- Others

Section 3: Platform Trust & Willingness to Use

Q9. What are the features that would make you trust a task platform? (Multiple responses possible)

- Identities of service providers verified
- User reviews about previous experiences
- Escrow payment – Payment is only made when the task is successfully completed
- Dispute resolution within the app
- Customer support available 24/7
- Referral from a friend

Q10. Are you ready to accept the platform commission of 5–10%?

- Definitely yes

- Yes, provided that the Helpers are reliable
- I don’t want to pay any commission to the platform
- I am unsure

Q11. How much money would you be allocating for your normal tasks?

- Less than PKR 300
- PKR 300 to 700
- PKR 700 to 1,500
- PKR 1,500 to 3,000
- More than PKR 3,000

Q12. Are there any platforms in Pakistan currently for finding local helpers (excluding ride-sharing and employment)?

- Yes, mention the name(s): _____
- No, I do not know any

Q13. If such a platform offers verified helpers, in-application chat, and a secure payment system, would you be using it?

- I will definitely use the platform.
- There is a high possibility that I may use the platform.
- I am uncertain.
- There is a low possibility that I may use the platform.
- I will definitely not use the platform.

Section 4: Earning Interest (Helper Segment)

Q14. Would you be considering making money from completing the other people’s tasks through a similar online platform?

Q15. What tasks do you think you are capable of handling for other people? (Select all that apply)

- Delivery / Courier Service
- Home Maintenance and Repair
- Cleaning

- Assembling / Relocation
- Tutoring
- Graphic and Digital Design
- Other

B.2. Aggregated Survey Response Data

The table below presents the complete response data from the 120 individuals who were surveyed in March 2026.

Q#	Question	Response Options	Results (n=120)
Q5	Frequency of task help needed	Multiple times/week | Weekly | 2–3x/month | Monthly | Rarely	12% | 21% | 33% | 12% | 22%
Q6	How they find help (multi-select)	Friends/family | FB/WhatsApp | OLX | Service Co. | App | Other	81% | 74% | 52% | 39% | 11% | 8%
Q7	Difficulty finding reliable help	Very difficult | Difficult | Neutral | Easy | Very Easy	28% | 36% | 19% | 12% | 5%
Q8	Tasks most frequently outsourced	Delivery | Repairs | Assembly | Cleaning | Grocery	72% | 61% | 47% | 44% | 38%
Q9	Trust factors (multi-select)	Verification | Ratings | Escrow | Disputes | Support | Referral	82% | 76% | 68% | 54% | 43% | 31%
Q10	The Willingness to pay 5–10% commission	Yes | Yes if good quality | No | Unsure about it	38% | 33% | 17% | 12%
Q11	The Typical task budget	<300 | 300–700 | 700–1500 | 1500–3000 | >3000	8% | 29% | 41% | 16% | 6%
Q12	Being Aware of existing task marketplace	Yes | No	11% | 89%
Q13	Would you use KaamSaaz-type platform	Definitely | Probably | Unsure | Not Probably | Not Definitely	44% | 33% | 14% | 6% | 3%
Q14	What is the Interest in earning via platform	Main income | Supplementary Income | No | Unsure	14% | 48% | 22% | 16%

B.3. Key Insights from Survey Data

- The survey results give us some information about KaamSaazs design and business model.

- Many people need help with tasks. In fact 66% of those who took our survey need assistance every week or a few times a month. Only 11% have ever used an app for this. This means there is a lot of room for improvement, for KaamSaaz to help people with their tasks.
- The main thing that sets us apart is trust. People found verification to be more important than a low commission rate. In fact 82% said verification was crucial compared to 71% who said a low commission rate was important. Ratings were also more important than price with 76% saying
- We think our commission is fair. 71% Of people are okay with paying a commission of 5-10%. If we look at the popular budget for tasks, which is between PKR 700 and 1,500 our commission per task would be PKR 56-150. This means we would make PKR 8,400-22,500, in commission per day if we complete 100 tasks.