# Marketeer



## Group Members

Shahum Malik (01-131222-044)

Faaiz Muzzammil (01-131222-054)

*Supervisor:* Dr. Engr. Joddat Fatima


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

May 2026

# FYP Completion Certificate

**Student Name:** Shahum Malik **Enrollment No:** 01-131222-044

**Student Name:** Faaiz Muzzammil **Enrollment No:** 01-131222-054

**Program of Study:** Bachelor of Software Engineering (BSE)

**Project Title:** Marketeer

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 _____ that is within the permissible limit set by the HEC. I have also found the thesis in a format recognised by the department.

**Supervisor's Signature: _____**

**Date: _____** **Name:** Dr. Engr. Joddat Fatima

# Certificate of Originality

This is certify that the intellectual contents of the project **Marketeer** 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:** Shahum Malik

**Signature:** _____ **Date:** _____

**Name of the Student:** Faaiz Muzzammil

**Signature:** _____ **Date:** _____

# 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 Consumption 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

Every second spent by a fintech brand designing social media posts is wasted, because the market will be long gone before the post is ready. For financial institutions, the time between a market movement happening and when branded content reaches an audience can be what keeps them relevant.

**Marketeer closes that gap entirely.**

Designed specifically for the financial industry, Marketeer takes raw live financial data and turns it into branded social media content within seconds, not hours. The tool tracks live financial feeds, uses artificial intelligence to write unique captions based on actual figures, and creates content which follows the guidelines of each brand's visual identity.

In terms of creativity, Marketeer delivers everything that a modern marketer needs when it comes to content creation. The branded **Image** posts are generated using live stock data, accurate company logos and colours which fill out the templates with a single click. **Videos** are created with real-time animations of stock charts, voiceovers, background music and much more. Professional **Newsletters** can be generated using live market data to send out to the whole subscriber list in one go.

Further, Marketeer integrates with WhatsApp via **Ticker**, a personalised bot which provides live market alerts and allows for on the fly content generation and editing through a mobile phone.

The core philosophy behind Marketeer relies on the Human-in-the-Loop process. All the generated content will go through the compliance module and then will have to get approved by humans for distribution on social media.

**The result:** time for post creation cuts down from hours to seconds, 95% brand consistency achieved on all platforms, and a fintech marketing team who is finally able to keep pace with the market.

### Keywords

Content Automation, AI, FinTech, Generative AI, Social Media Marketing, Conversational AI, WhatsApp Automation

# Dedication

*Without the endless encouragement from our families in helping us through the all-nighters and early mornings spent in creating this, Marketeer would never have become a reality.*

*A debt of thanks is owed to Dr. Joddat Fatima for her guidance and supervision, which led us in the right path to turn this idea into a practical form.*

*And lastly, to every individual working in the field of fintech, who has sacrificed many a good night's sleep worrying about their social media posts and getting them to look perfect —* ***this is built for you!***

# Acknowledgements

First and foremost, All praise is due to Allah (SWT), the Most Beneficent and the Most Merciful. He has provided us with the favour and determination to complete this project successfully.

It is our pleasure to acknowledge the efforts of **Dr. Joddat Fatima**, our project supervisor and Senior Assistant Professor of the Department of Software Engineering at Bahria University, Islamabad, who assisted us throughout this project. Thanks to her extreme involvement in our education, we were able to do much better than expected of us.

We are immensely thankful to the **Department of Software Engineering at Bahria University, Islamabad,** for the guidance, laboratories, and support provided by them that allowed this complex project to be executed successfully in our undergraduate studies.

Last but not least, we are thankful to **Mr. Saad Siddique**, CEO of MarketVerse Inc., who was our industrial supervisor and had faith in this idea from day one. Without his financial and any other resourceful assistance, it would not have been possible for us to make **Marketeer** a product meeting the market standards.

There are no words that can possibly express our gratitude towards the creators of open-source technologies such as OpenClaw[10], Remotion[1], React.js[2], Node.js[4], Supabase[3], BannerBear[5], and ElevenLabs[6]. We owe all of this to the wonderful tools that they created.

Above all, we would like to give our sincerest thank you to our **parents and families**. Every single line of code was typed with prayers and endless support from them. Nothing would have been achieved without them.

**Shahum Malik & Faaiz Muzzammil**

*Bahria University, Islamabad*

*2026*

# List of Figures

# List of Tables

# Table of Contents

# Chapter 1

# Introduction

## 1.1. Motivation

There are difficulties faced by the developers and businesses that are working in the area of fintech in terms of generating a well-coordinated post on the social media site that can be posted just at the right time. The process of manually creating contents slows down the chance of giving valuable inputs to its audience in a timely manner. Thus, any kind of change that occurs in the financial environment, like stock prices, changes in regulations, cannot be considered due to the lack of ability to publish the information immediately and with a special design. It is why we have developed our own tool which can solve this issue and can work in connection with the dynamics of the financial market.

## 1.2. Objectives

Our product's major objectives include:

1. Production of branded posts based on the live data generated in under 60 seconds.
2. Use of at least three different live financial APIs to get live stats.
3. Designs will strictly adhere to brand guidelines (95% logo, font, colour accuracy).
4. Posting and scheduling of content on X, Instagram, and LinkedIn.

## 1.3. Main contributions

- **Marketeer** is an automation tool that leverages AI to transform real-time market data into social media content. This allows fintech brands to automate their content creation and eliminate the need for manual designing and posting schedules.
- With Marketeer, posts can be created within seconds rather than hours while ensuring consistent brand identity across all social platforms, eliminating the repetitive manual effort typically required by design teams.
- With **Ticker,** an AI bot built using OpenClaw[10], users can access the entire Marketeer platform via WhatsApp. The bot keeps track of market activity, sends out alerts in real-time, and can create branded content and reels on the go.
- With all of this in place, Marketeer keeps the final decision and control with the user through Human-in-the-loop (HITL) model at its core.

## 1.4. Report organisation

Figure 1.1 illustrates the outline of the report.

**Chapter 2** contains an additional discussion of the literature review section, such that comparisons may be made between the performance of competing applications and the real-time automation technology utilised by Marketeer.

**Chapter 3** contains a detailed discussion on the capabilities of the system with specific mention of use cases, functionality limitations and non-functionality features of the system.

**Chapter 4** focuses on the system design phase, wherein the design components of the system such as the architecture, logical and dynamic deployments and the data models are presented.

**Chapter 5** deals with the implementation phase wherein there is a discussion regarding the software framework of the application along with the APIs used by the system.

**Chapter 6** presents an overview of the system testing phase which includes discussions regarding testing methodology and test cases.

**Chapter 7** wraps up the entire thesis, summarising everything we accomplished, sharing our final thoughts, and looking ahead to what could be added in the future.

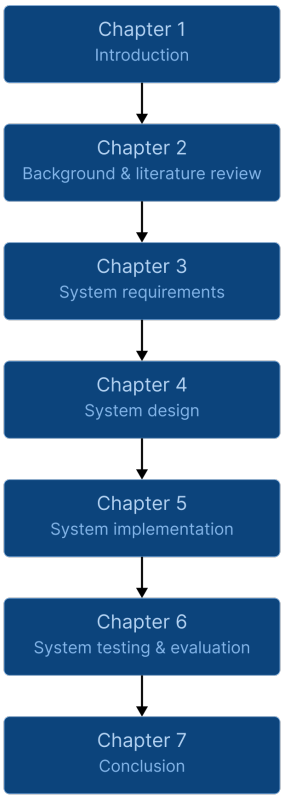


*Figure 1.1: Thesis Organization*

# Chapter 2

# Background Study / Literature Review

Currently, the implementation of social media marketing for use in the financial sector requires repetitive actions from marketers and the involvement of many people. Although special applications help schedule posts in advance, they cannot analyze information in **real-time** nor make decisions themselves. This chapter presents important advancements in terms of automating content creation. We have discussed several key aspects concerning the ways businesses automate content creation and the technology behind it. The last topic concerns human delay while creating Fintech content through commonly used programs, namely Canva, Buffer and Hootsuite.

## 2.1. Key Concepts

**Marketeer** performs a task through three operations: receiving information from the stock exchange, analysis of this information aimed at creating captions (with the help of Agent Orchestration) and developing visual content through automation (BannerBear[5], Remotion[1]).

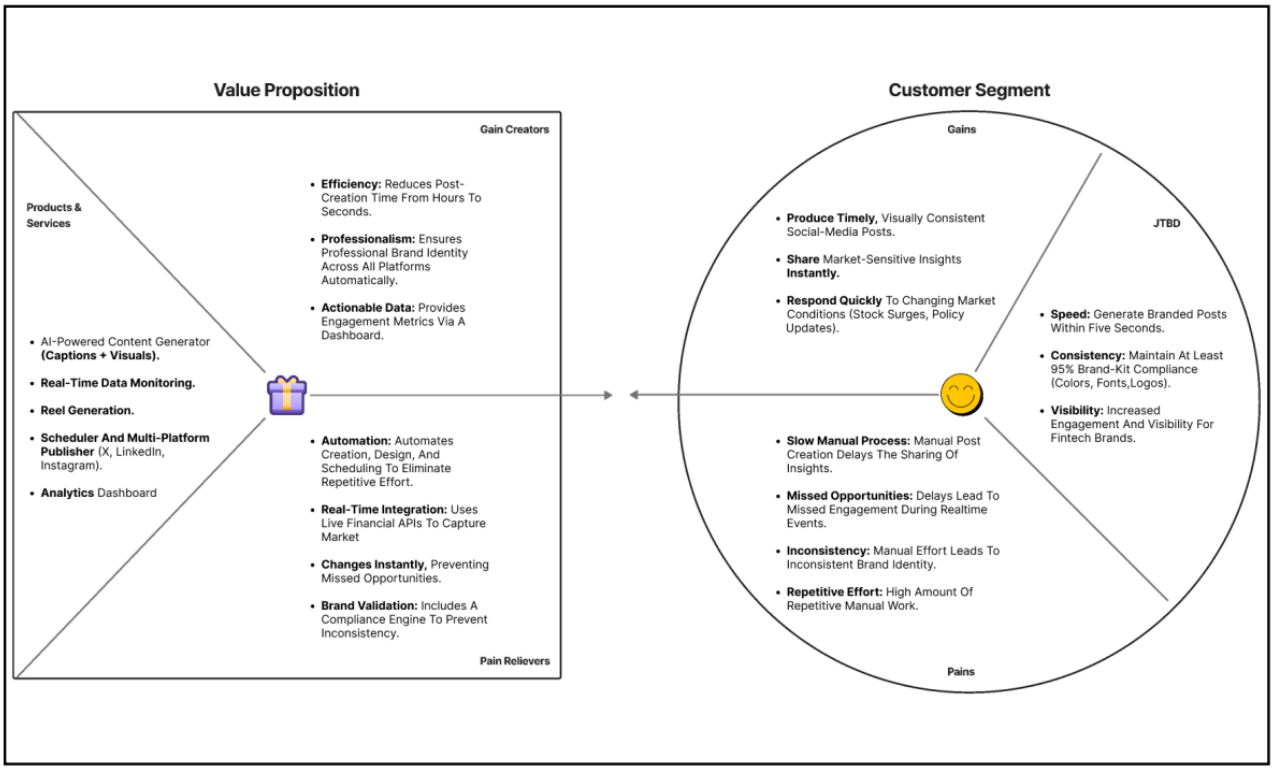


*Figure 2.1: Value Proposition Canvas*

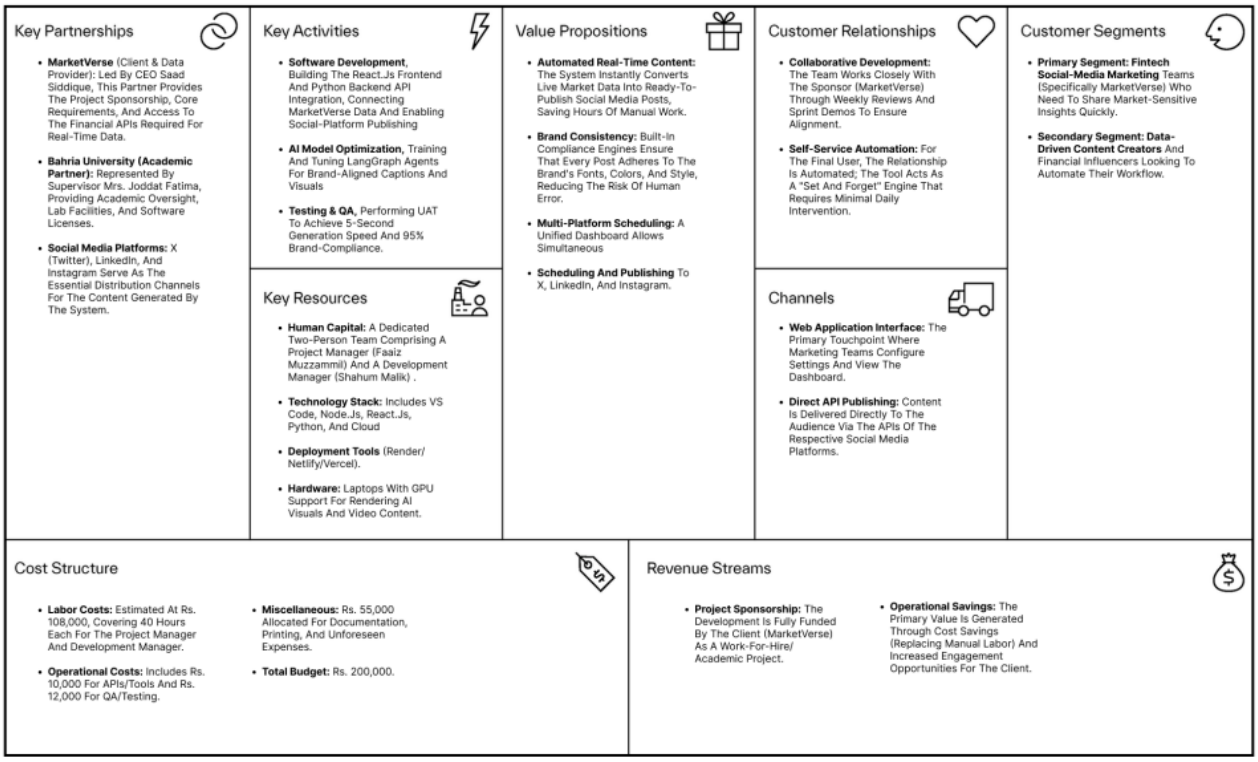


*Figure 2.2: Business Model Canvas*

## 2.2. Evolution of Content Automation

The adoption of artificial intelligence-driven marketing has brought about a transformation in communication within the banking industry. It was once fragmented, but in the digital era, all the pieces have finally come together.

*Table 2.1: Comparison of Traditional Schedulers and Marketeer*

| **Characteristics** | **Traditional Schedulers (Hootsuite, Buffer)** | **AI Automation Engine (Marketeer)** |
|---|---|---|
| Problem to be Solved | Planning posts already prepared. | Planning posts right away due to some unforeseen changes in the market. |
| Purpose of Collaboration | Planning posts ahead in order to share weeks or months after. | Quick reaction on any changes occurring in the market. |

| Characteristics | Traditional Schedulers (Hootsuite, Buffer) | AI Automation Engine (Marketeer) |
|---|---|---|
| Data Integration | Entirely depends on the data input manually entered by the user. | Automated process of pulling data from financial APIs, with the option of manual editing. |
| Brand alignment | Requires people to manually check each image. | Instantly verifies all graphics against your specific brand kit. |
| Workflow Gaps | Designing and scheduling are handled as separate tasks. | Combines the design, writing, and publishing phases into one smooth process. |
| Trigger Mechanism | Runs on a strict calendar queue. | Kicks into action based on live market changes and programmatic code. |
| Tech Capabilities | Basically just publishes ready-made posts. | Runs entirely on advanced, end-to-end generative tech. |

### 2.3. Limitations of Existing Systems

Although almost all companies are using an automated strategy, there are only a handful that make changes based on latest dynamics. Financial personalities and companies often miss out on many opportunities owing to the reason that the stock market operates **far faster** than even the best design teams.

### 2.4. Programmatic Media Mapping

Marketeer uses a programmatic approach with tools like Remotion[1] and BannerBear[5] to auto-fill brand templates with live financial data. The process can be modeled mathematically: arrays of market data (d1...dn) multiplied by Brand Kit rule matrices (r) produce ready-to-post designs (p1...pm).

$$(p_1, p_2, \ldots, p_n) = (d_1, d_2, \ldots, d_m) \bullet \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{bmatrix}$$

*Figure 2.3: Mathematical Model*

By uniting engineering principles and graphic design, this approach views visual elements as lines of code rather than mere files. The system facilitates the algorithmic manipulation of visual hierarchy, mapping real-time API content into design variables. For instance, interactive stock graphs or colour schemes that adapt to changing market dynamics enables the system to ensure 95% brand kit compliance while significantly reducing processing time.


*Figure 2.4: Template Tagging by Marketeer*

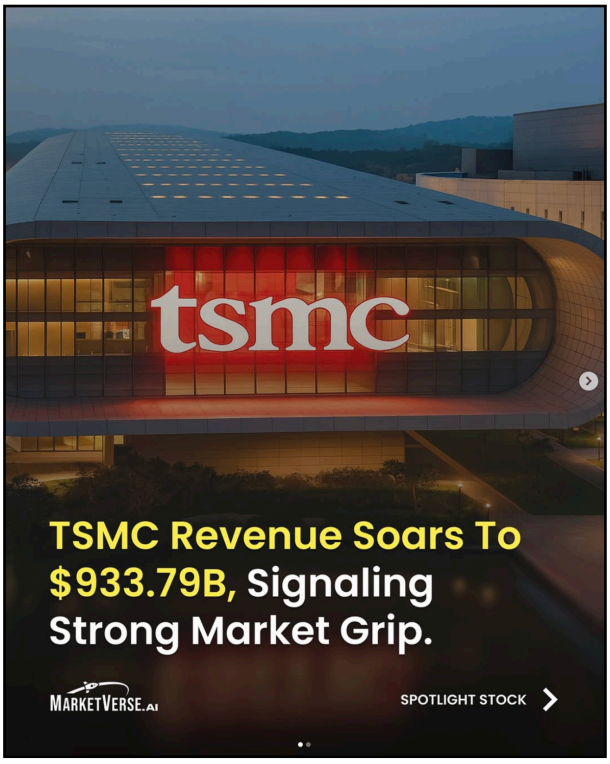


*Figure 2.5: Actual Output by Marketeer*

## 2.5. The Problem of AI's Hallucinations in FinTech Applications

While there is no doubt that generative AI is highly efficient at content generation, there are some limitations when it comes to the use of this technology in the finance sector since AI generates fluent and logical texts with factual inaccuracies that lead to serious legal implications for the company and make fatal misreporting in the markets more likely.

According to the latest research, the best AI systems are still prone to the issue of hallucination when working with lengthy texts in the language of finance. This may be explained by two reasons, incorrect prompting due to vague instructions and the preference of probability generation within the model. To avoid the risk when using AI, **Marketeer prevents any assumptions by including real market data and brand details in the prompt instantly.**

### 2.6. Human-in-the-Loop Automation for Regulator Compliance

It goes without saying that the legal framework of the financial industry is highly rigid. According to multiple studies that explore how risk management and compliance work, despite rapid development in fintech and fast adoption of AI, it is vital for us to preserve human supervision to keep our businesses running effectively and in a proper way.

This phenomenon gives birth to **"human-in-the-loop" (HITL)** agentic AI as an innovative technology that tries to find a compromise between the quick work of machines and more structured human review of the process. HITL implementation in the sphere of financial services not only boosts the efficiency of the AI but also ensures regulatory compliance. **Marketeer is an example of a compliant HITL AI solution** as it keeps the ultimate decision on publication to a human operator.

# Chapter 3

# System Requirements

The focus of this chapter lies on researching the features necessary for the proper functioning of **Marketeer**, from the perspective of its practical implementation, interaction with its environment, and the limitations that may appear.

## 3.1. Use Case Diagram

The diagram shows how various entities associated with the system interact with each other. The number of major actors in the picture is equal to five: **Marketing Manager**, **Content Strategist**, **Brand Manager**, **System Administrator**, and automatic **Financial Company API**. Actions performed by these entities include authentication, defining brand policies, creation of captions by means of language models, creation of images, and finally, conformity check before scheduling.

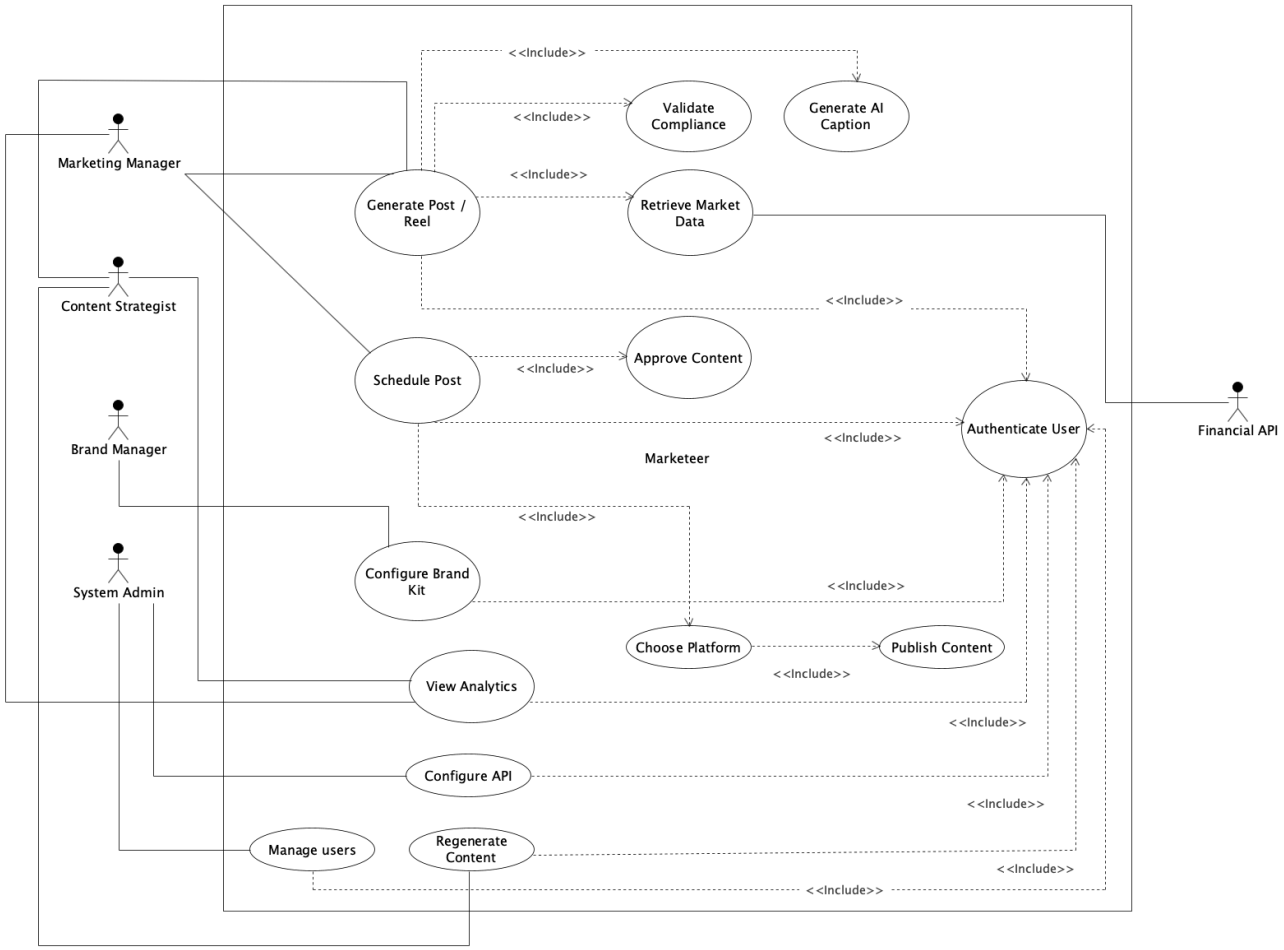


*Figure 3.1: Use Case Diagram*

## 3.2. Functional Requirements

Below are the details of our use cases, providing a detailed definition of our system's functional requirements and elaborations of the processes that were mentioned earlier in Section 3.1.

### 3.2.1. Authenticate User

*Table 3.1: Authenticate User*

| | | | |
|---|---|---|---|
| **Use Case ID:** | UC-UM-01 | | |
| **Use Case Name:** | Authenticate User | | |
| **Actors** | Marketing Manager, Content Strategist, Brand Manager, System Admin | | |
| **Pre-Conditions:** | A fully registered account must already exist on the platform before proceeding. | | |
| **Priority:** | High | | |
| **Basic Flow:** | An approved individual signs in with their details to reach their private workspace. | | |
| **Actor Actions** | | **System Response** | |
| **1** | The person types in their email or username alongside their password, then hits the login button. | **2** | The platform verifies these details. If everything matches up, it grants entry and pulls up the main interface. |
| **3** | The person waits while their custom workspace gets ready. | **4** | Depending on their assigned permissions, the platform loads the appropriate modules, such as the admin panel, the brand management suite, or content builders. |
| **Alternative Course of Action (if any)** | | | |
| **Actor Actions** | | **System Response** | |
| **2.a** | The person enters the wrong login info. | The platform shows an error message and refreshes the sign-in screen. | |

### 3.2.2. Manage User

*Table 3.2: Manage User*

| | | | |
|---|---|---|---|
| **Use Case ID:** | UC-UM-02 | | |
| **Use Case Name:** | Manage User | | |
| **Actors** | System Admin | | |
| **Pre-Conditions:** | Administrator should log into the system before performing any operation. | | |
| **Priority:** | High | | |
| **Basic Flow:** | The administrator is supposed to perform the creation, modification, or deletion of a particular user profile. | | |
| **Actor Actions** | | **System Response** | |
| **1** | The administrator opts for creating, changing, or deleting the user profile. | **2** | The system initiates the user account management tools and stores the updated information into the database. |
| **Alternative Course of Action (if any)** | | | |
| **Actor Actions** | | **System Response** | |
| | Administrator provides incorrect data. | The system returns an error message and halts the procedure. | |

### 3.2.3. Configure API

*Table 3.3: Configure API*

| | | | |
|---|---|---|---|
| **Use Case ID:** | UC-SYS-01 | | |
| **Use Case Name:** | Configure API | | |
| **Actors** | System Admin | | |
| **Pre-Conditions:** | Administrator must be logged in the system. | | |
| **Priority:** | High | | |
| **Basic Flow:** | Admin configures external API credentials. | | |
| **Actor Actions** | | **System Response** | |
| **1** | Administrator logs in to the API configuration page. | **2** | The Platform shows the configured API keys along with connectivity status. |
| **3** | Administrator configures API keys. | **4** | Platform validates the API keys, encodes and saves them securely. |
| **Actor Actions** | | **System Response** | |
| | Administrator configures with incorrect or incomplete data. | Platform shows error messages. | |

### 3.2.4. Configure Brand Kit

*Table 3.4: Configure Brand Kit*

| | | | |
|---|---|---|---|
| **Use Case ID:** | UC-BK-01 | | |
| **Use Case Name:** | Configure Brand Kit | | |
| **Actors** | Brand Manager, System Admin | | |
| **Pre-Conditions:** | The person must be actively logged into the platform. | | |
| **Priority:** | High | | |
| **Basic Flow:** | The individual configures the main visual assets that represent the brand. | | |
| **Actor Actions** | | **System Response** | |
| **1** | The person logs in and launches the Brand Kit section. | **2** | The platform brings up the brand's existing design kit. |
| **3** | The person adds a new logo, then selects their specific typography and color palette. | **4** | The platform scans the uploaded files to confirm they meet the required format standards. |
| **5** | The person clicks the "Save" button. | **6** | The platform updates the visual settings and automatically enforces them across all modules. |
| **Alternative Course of Action (if any)** | | | |
| **Actor Actions** | | **System Response** | |
| | The person attempts to upload media that has an incompatible format or exceeds size limits. | The platform blocks the file upload and flashes a warning message. | |

### 3.2.5. Retrieve Market Data

*Table 3.5: Retrieve Market Data*

| | | | |
|---|---|---|---|
| **Use Case ID:** | UC-MD-01 | | |
| **Use Case Name:** | Retrieve Market Data | | |
| **Actors** | System | | |
| **Pre-Conditions:** | You must set up the API keys beforehand. | | |
| **Priority:** | High | | |
| **Basic Flow:** | The platform automatically grabs the latest financial figures. | | |
| **Actor Actions** | | **System Response** | |
| | — | **1** | The platform asks the Financial Company API for the information. |
| | — | **2** | The API replies by sending over the freshest market stats. |
| | — | **3** | The platform saves these figures directly into the database or cache. |
| | — | **4** | These numbers are now ready to be used to build content. |
| **Alternative Course of Action (if any)** | | | |
| **Actor Actions** | | **System Response** | |
| | — | If the API connection drops, the platform records the error and immediately tries to connect again. | |

### 3.2.6. Generate AI Caption

*Table 3.6: Generate AI Caption*

| | | | |
|---|---|---|---|
| **Use Case ID:** | UC-AI-01 | | |
| **Use Case Name:** | Generate AI Caption | | |
| **Actors** | Marketing Manager, Content Strategist | | |
| **Pre-Conditions:** | You must have current market stats available and a fully configured brand kit ready to go. | | |
| **Priority:** | High | | |
| **Basic Flow:** | A person utilizes the platform to create a smart, automatically generated caption. | | |
| **Actor Actions** | | **System Response** | |
| **1** | The person selects the specific asset or market event, along with the tone they want to use. | **2** | The platform builds a prompt that combines live market numbers with the brand's specific writing rules. |
| **3** | The person clicks the "Generate Caption" button. | **4** | The LLM writes the text by running it through the Agent Orchestration workflow. |
| **5** | The person clicks "Save". | **6** | The platform saves these settings and strictly applies the rules everywhere. |
| **Alternative Course of Action (if any)** | | | |
| **Actor Actions** | | **System Response** | |
| | The person decides they want a new version or wants to tweak the text by hand. | The platform makes it simple to rewrite or manually adjust the wording. | |

### 3.2.7. Generate Post / Reel

*Table 3.7: Generate Post / Reel*

| | | | |
|---|---|---|---|
| **Use Case ID:** | UC-CG-01 | | |
| **Use Case Name:** | Generate Post / Reel | | |
| **Actors** | Marketing Manager, Content Strategist | | |
| **Pre-Conditions:** | Caption created, Brand kit created. | | |
| **Priority:** | High | | |
| **Basic Flow:** | Visual content is generated using the brand kit templates. | | |
| **Actor Actions** | | **System Response** | |
| **1** | The user selects the media type (image/video). | **2** | The system retrieves the corresponding template. |
| **3** | The user clicks on "Generate Content". | **4** | The system generates the visuals using Design API and brand guidelines. |
| **Alternative Course of Action (if any)** | | | |
| **Actor Actions** | | **System Response** | |
| | Failure to generate the visual. | Display error message. | |

### 3.2.8. Regenerate Content

*Table 3.8: Regenerate Content*

| | | | |
|---|---|---|---|
| **Use Case ID:** | UC-CG-02 | | |
| **Use Case Name:** | Regenerate Content | | |
| **Actors** | Marketing Manager, Content Strategist | | |
| **Pre-Conditions:** | The attempt to generate the content must already have been done. | | |
| **Priority:** | Medium | | |
| **Basic Flow:** | The user attempts to generate new caption/artwork. | | |
| **Actor Actions** | | **System Response** | |
| **1** | User touches the "Generate Content Again" button. | **2** | System re-generates the caption/artwork. |
| **3** | User waits for the results. | **4** | System shows the generated caption/ artwork. |
| **Alternative Course of Action (if any)** | | | |
| **Actor Actions** | | **System Response** | |
| | The user generates repeatedly. | System keeps records of each version. | |

### 3.2.9. Approve Content

*Table 3.9: Approve Content*

| | | | |
|---|---|---|---|
| **Use Case ID:** | UC-AP-01 | | |
| **Use Case Name:** | Approve Content | | |
| **Actors** | Marketing Manager, Brand Manager | | |
| **Pre-Conditions:** | The content must be developed and then approved for conformity. | | |
| **Priority:** | High | | |
| **Basic Flow:** | The user will approve the developed content for publishing/ scheduling. | | |
| **Actor Actions** | | **System Response** | |
| **1** | The user accesses the content approval page. | **2** | The system displays the preview of the caption and images. |
| **3** | The user clicks on "Approve". | **4** | The system will approve the content and make it available for publishing/ scheduling. |
| **Alternative Course of Action (if any)** | | | |
| **Actor Actions** | | **System Response** | |
| | The user disapproves the content. | The system will disapprove the content and create it again. | |

### 3.2.10. Choose Platform

*Table 3.10: Choose Platform*

| | | | |
|---|---|---|---|
| **Use Case ID:** | UC-SP-01 | | |
| **Use Case Name:** | Choose Platform | | |
| **Actors** | Marketing Manager, Content Strategist | | |
| **Pre-Conditions:** | The content must be developed and then approved for conformity. | | |
| **Priority:** | Medium | | |
| **Basic Flow:** | The user will approve the developed content for publishing/ scheduling. | | |
| **Actor Actions** | | **System Response** | |
| **1** | The user accesses the content approval page. | **2** | The system displays the preview of the caption and images. |
| **3** | The user clicks on "Approve". | **4** | The system will approve the content and make it available for publishing/ scheduling. |
| **Alternative Course of Action (if any)** | | | |
| **Actor Actions** | | **System Response** | |
| | The user disapproves the content. | The system will disapprove the content and create it again. | |

### 3.2.11. Schedule Post

*Table 3.11: Schedule Post*

| | | | |
|---|---|---|---|
| **Use Case ID:** | UC-SP-02 | | |
| **Use Case Name:** | Schedule Post | | |
| **Actors** | Marketing Manager, Content Strategist | | |
| **Pre-Conditions:** | The post should be created and approved. | | |
| **Priority:** | High | | |
| **Basic Flow:** | User schedules the posts for publishing in the future. | | |
| **Actor Actions** | | **System Response** | |
| **1** | User chooses date, time and platforms. | **2** | System checks scheduling information. |
| **3** | User clicks “Schedule”. | **4** | System schedules post. |
| **Alternative Course of Action (if any)** | | | |
| **Actor Actions** | | **System Response** | |
| **1.a** | User chooses incorrect date or time. | System displays error message. | |

### 3.2.12. Publish Content

*Table 3.12: Publish Content*

| | | | |
|---|---|---|---|
| **Use Case ID:** | UC-SP-03 | | |
| **Use Case Name:** | Publish Content | | |
| **Actors** | System | | |
| **Pre-Conditions:** | The scheduled time must be reached. | | |
| **Priority:** | High | | |
| **Basic Flow:** | The content is published by the system on the selected platforms. | | |
| **Actor Actions** | | **System Response** | |
| | — | **1** | Triggered publish event by scheduler |
| | — | **2** | Published content on selected platforms |
| | — | **3** | Platform returns success or failure result |
| | — | **4** | Modifies post status |
| **Alternative Course of Action (if any)** | | | |
| **Actor Actions** | | **System Response** | |
| | — | — | |

### 3.2.13. View Analytics

*Table 3.13: View Analytics*

| | | | |
|---|---|---|---|
| **Use Case ID:** | UC-AC-01 | | |
| **Use Case Name:** | View Analytics | | |
| **Actors** | Marketing Manager, Content Strategist | | |
| **Pre-Conditions:** | Posts should be published. | | |
| **Priority:** | Medium | | |
| **Basic Flow:** | System checks for content compliance. | | |
| **Actor Actions** | | **System Response** | |
| **1** | User logs into Analytics Dashboard. | **2** | System fetches engagement numbers. |
| **3** | User sets dates/website options. | **4** | System alters graphs/diagrams. |
| **Alternative Course of Action (if any)** | | | |
| **Actor Actions** | | **System Response** | |
| | — | — | |

## 3.3. Non-functional Requirements

This section defines the quality attributes and constraints that Marketeer must meet.

### 3.3.1. Performance Requirements

*Table 3.14: Performance Requirements*

| | |
|---|---|
| **3.3.1.1** | The system should be capable of generating a branded static post (image + caption) within a short duration of few seconds to a minute, subject to data availability and AI response. |
| **3.3.1.2** | The status of the posts once posted will be updated on the dashboard within one minute after the response from the platform. |

### 3.3.2. Safety Requirements

*Table 3.15: Safety Requirements*

| | |
|---|---|
| **3.3.2.1** | System shall prevent accidental deletion of brand kits and content by requiring an explicit confirmation step. |
| **3.3.2.2** | System shall log critical operations (publishing, deletion, brand changes) for audit and debugging. |

### 3.3.3. Security Requirements

*Table 3.16: Security Requirements*

| | |
|---|---|
| **3.3.3.1** | All user accounts will be authenticated through HTTPS protocol and password hashing. |
| **3.3.3.2** | The role of users will either be Admin, Marketing Manager, Content Strategist, or Viewer. |
| **3.3.3.3** | Only the admin role will have access to API keys/tokens. |
| **3.3.3.4** | Any failed login attempts exceeding three will lock out account. |

### 3.3.4. Software Quality Attributes

*Table 3.17: Software Quality Requirements*

| | | |
|---|---|---|
| 3.3.4.1 | **Availability** | The system must be available at least 95% in demo/Testing mode. |
| 3.3.4.2 | **Usability** | The core processes (Generate > Schedule > Publish) must be accomplished in 5–7 steps from an experienced user point of view. |
| 3.3.4.3 | **Maintainability** | The code base must be modular (API, AI, Content Generation, Scheduler Services must be separate). |
| 3.3.4.4 | **Scalability** | Design of architecture must support scalability with independent AI and content rendering components. |

### 3.3.5. Business Rules

*Table 3.18: Business Rules*

| | |
|---|---|
| 3.3.5.1 | Only users who have the roles of Manager/Administrator will be allowed to approve content to publish. |
| 3.3.5.2 | Compliance testing should be done prior to the scheduling of content. |
| 3.3.5.3 | A predefined disclaimer should be included automatically for all required posts. |

### 3.3.6. Other Requirements

*Table 3.19: Other Requirements*

| | | |
|---|---|---|
| 3.3.6.1 | **Database** | The database should store all post information, configurations, and logs at least until the duration of FYP evaluation. |
| 3.3.6.2 | **Language** | The system shall be programmed in English only; multilingual support is beyond the current scope of this project. |
| 3.3.6.3 | **Legal** | All legal aspects other than the minimum disclaimer are the client's concern (Financial Company). |

## 3.4. Interface Requirements

### 3.4.1. User Interfaces

The application is accessed through a web interface accessible through desktop and mobile web browsers. The interface structure includes a context-aware menu bar situated at the bottom of the screen and a workspace split into different panels. Several interfaces include the Dashboard, Image Mode, Video Mode, Brand Kit Manager (Settings Page), Ticker Connection and the Scheduler.

### 3.4.2. Hardware Interfaces

Specific hardware devices are not required to use this application. This application can be operated in regular computer devices such as desktops and smartphones. An active internet connection is required for the system.

### 3.4.3. Software Interfaces

**Marketeer** connects to a range of different Application Programming Interfaces (APIs) in a unified manner to carry out its primary functions:

- **MarketVerse & Financial APIs:** Acquires live stock prices and market information.

- **Design APIs & Remotion**[1]**:** Placid, BannerBear[5] and Figma[14] REST API is used for the creation of static visuals, whereas Remotion[1] manages video production through React[2] code, ElevenLabs[6] outputs the audio and voice-overs and Brandfetch[16] provides the logos and colours for the companies featured in stocks.
- **AI Tools:** Gemini[17], LangGraph[18] and OpenClaw[10] regulate the AI process and creating textual content.
- **Social Media APIs:** They help in automation of posting content to their platforms.

### 3.4.4. Communication Interfaces

The client, server, and various external APIs connect through HTTP/HTTPS on port 443. The communication between the server and client uses RESTful architecture, enabling the transmission of JSON data encrypted with TLS/SSL.

## 3.5. Database Requirements

**Supabase**[3] acts as the RDBMS that stores all data in a permanent form.

The database holds data related to:

- **User Data:** Credentials for user authentication, role type, and token.
- **Brand Kits:** Brand logo, primary colour code, typography, and disclaimer.
- **Content Data:** Metadata of the post, captions generated, URL of the media file, and compliance score.
- **Activity Data:** It comprises the posting schedule and performance.

## 3.6. Analysis Models

The challenge of content generation at runtime is tackled by the system using a four-layer analysis strategy.

1. **AI-Powered Content Generation:** Market and company data are gathered, and the generation guidelines for Gemini[17] and LangGraph[18] generation algorithms are formulated based on that.

2. **Compliance Validation of Generated Content:** The generated content is validated and evaluated from an accuracy standpoint. In the absence of text similar to "Risk Warning," the procedure terminates until the matter is sorted out.

3. **Posting (Background Process):** A background process periodically checks the database for any content requiring posting. Once a piece of content is detected, the required information is sent to chosen social media platforms.

4. **Data Gathering for Performance Evaluation:** The required data is gathered for performance evaluation purposes.

## 3.7. Project Feasibility

- **Technical feasibility:** It can be stated that architectural approach is very feasible due to reliability of the technology stack used in this case. The use of React[2] for frontend development and Python together with Node.js[4] for backend allows us to consider the chosen architectural pattern rather good. Technology used for creation of videos programmatically is called Remotion[1]. As far as modularity of the future app is concerned, there should be API for integration purposes.

- **Operational feasibility:** Undoubtedly, one can say that the application Marketeer can solve a certain business problem. Specifically, such an application helps reduce time for publishing financial articles in social media channels. In some cases, time required for publication will decrease dramatically. It will help to stay up-to-date with all the market trends.

- **Legal and ethical feasibility:** One can say that application Marketeer considers all possible legal issues. In particular, due to compliance validator all financial disclaimers will be integrated in all posts.

## 3.8. Conclusion

This illustrates the responsiveness and effectiveness as well as the connectivity of Marketeer. It is from the application flow and API integration that we are able to create a good foundation for the architectural design discussed in the next chapter.

# Chapter 4

# System Design

This chapter talks about the functioning of Marketeer, as well as interaction between components in Marketeer and different modules. This is an overview of the system's design pattern.

## 4.1. Design Approach

Marketeer has been designed based on the **Modular Monolithic Architecture**. Even though the entire system operates as a unified backend application, it is designed in a way that each module performs a different task. In order to effectively segregate the code, we followed a **layered architecture** approach where each layer corresponds to a particular responsibility; those are presentation, business, connector, and persistence layers as shown by diagrams below.

## 4.2. Design Constraints

There were some technological and operational constraints, which shaped the design of Marketeer. Our team had to accomplish the project within 12 weeks with a budget of PKR 200,000. Therefore, we had to use open source development environments and technologies that would provide us with licenses for academic purposes.

Moreover, the system requires continuous Internet connectivity to interface with real-world financial APIs and cloud services like OpenClaw[10] and Remotion[1]. Due to the small team of two, we opted to develop an extremely modular backend architecture.

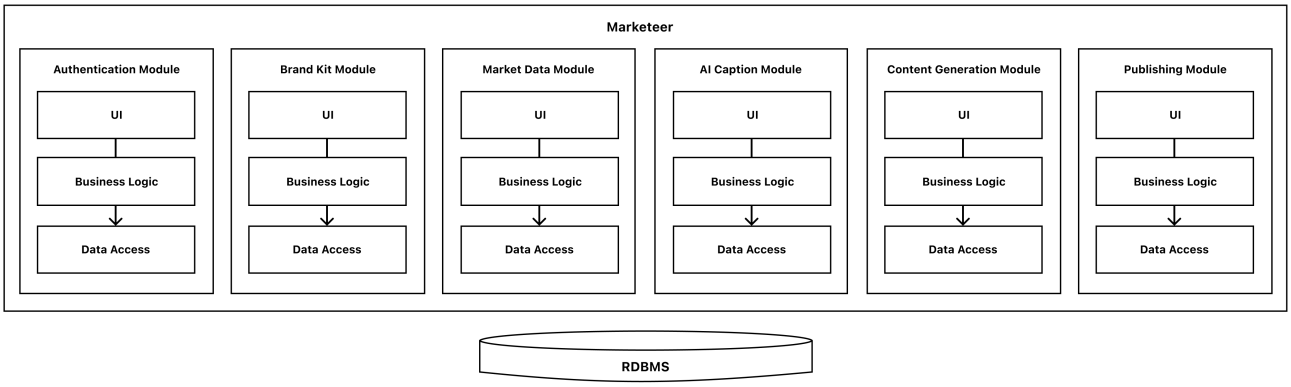


*Figure 4.1: Modular Monolithic Architecture*

## 4.3. System Architecture

The modular monolith is separated into three major modules. These include client's device, application server, and background worker server.

- **Client Device:** This module contains the user interface built using React[2] and Next.js technology which runs within the web browser of the client and communicates with the backend using HTTPS protocol.
- **Application Server:** Works as the backbone of the whole system where login services are performed by this server along with content creation, compliance test execution, connection between the system and financial services, and artificial intelligence applications.
- **Background Worker Server:** Performs heavy tasks in the background which include scheduling of postings, posting on social media platforms, and data tracking in order to make the application server speedy and efficient while communicating with external resources.

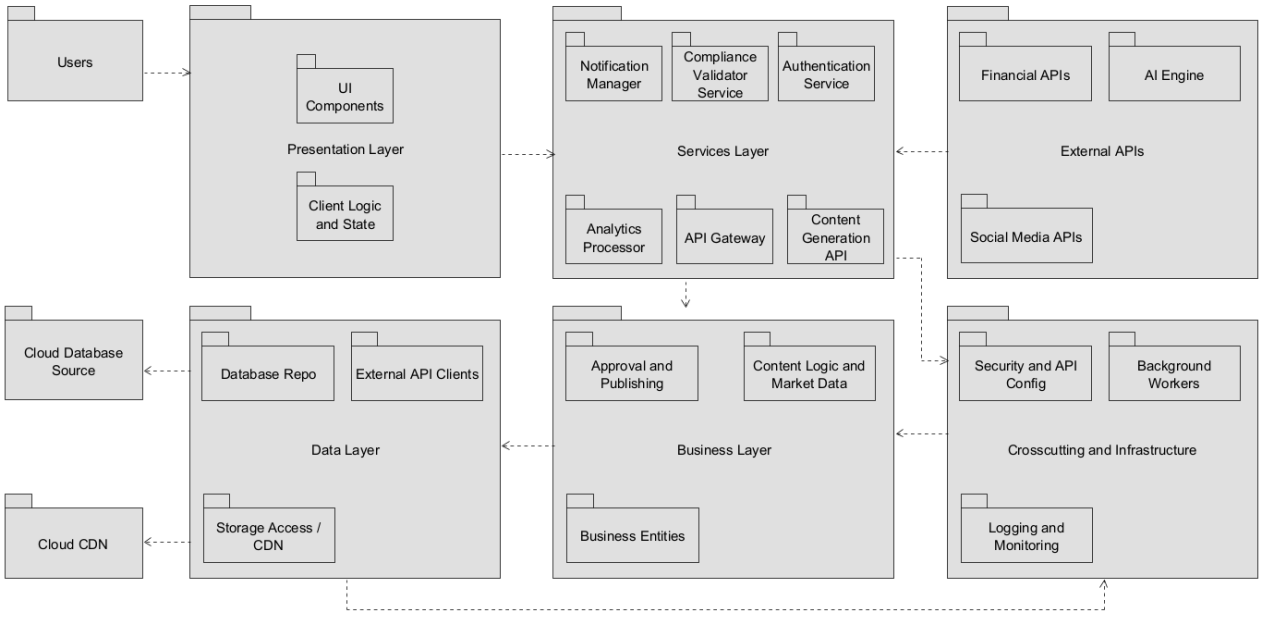


*Figure 4.2: Package Diagram*

## 4.4. Logical Design

The below-listed items are the descriptions of some functionalities forming the Marketeer architecture.

- **Authentication Module:** Login processing functionality, ongoing session handling, and access control depending on the role of the user.
- **Brand Kit Module:** Brand information including logo, fonts, colours, and other settings.
- **Market Data Module:** Acquires market data through APIs or MarketVerse in real-time.
- **AI Caption Module:** Creates captions and headlines through agent orchestration.
- **Content Generation Module:** Handles content creation by applying the use of design APIs and libraries.
- **Compliance Module:** Makes sure that all created content complies with the brand guidelines and includes legal disclaimers.
- **Scheduler Module:** Schedules posts and handle a calendar of posts.
- **Publisher Module:** Posts distribution on chosen social media platforms.

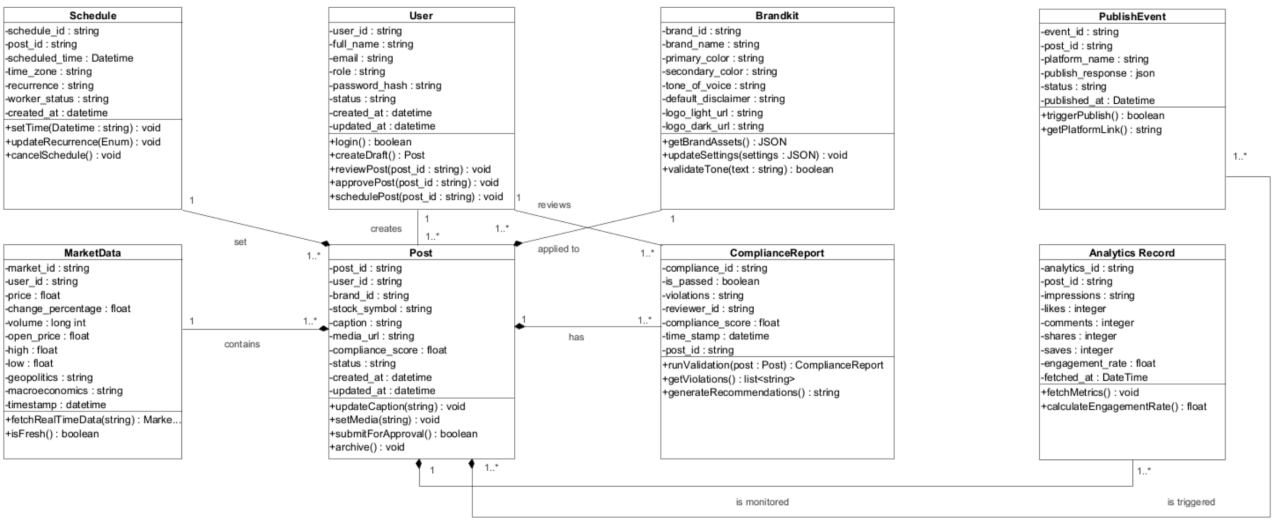


*Figure 4.3: Class Diagram*

## 4.5. Dynamic View

This section gives insight into the working of **Marketeer** at the time of its execution. Performing various operations in Marketeer involves four main stages of interaction among components in order to accomplish specific tasks.

- **Phase 1: Content Generation by AI:** Current market data, like current TSLA quotes, as well as information provided by the user about his brand kit, is combined and used to create a message that will go to the AI engine.
- **Phase 2: Compliance and Approvals:** After submission of the initial copy, the compliance tool conducts automatic checks for quality and assigns a certain rating and comments related to it, pointing out certain mistakes like lack of risk warnings, for example.
- **Phase 3: Publishing (Background Job):** In addition, while all the above processes are running, there is a constant check of database for posts to be published according to schedule. Posts are sent through social media APIs to social networks, such as X, LinkedIn, and Instagram at the scheduled moment.
- **Phase 4: Performance Metrics:** At certain intervals, the background job takes recent statistics of live social media posts for users. Information about views, likes, and other statistics is retrieved through social media APIs.

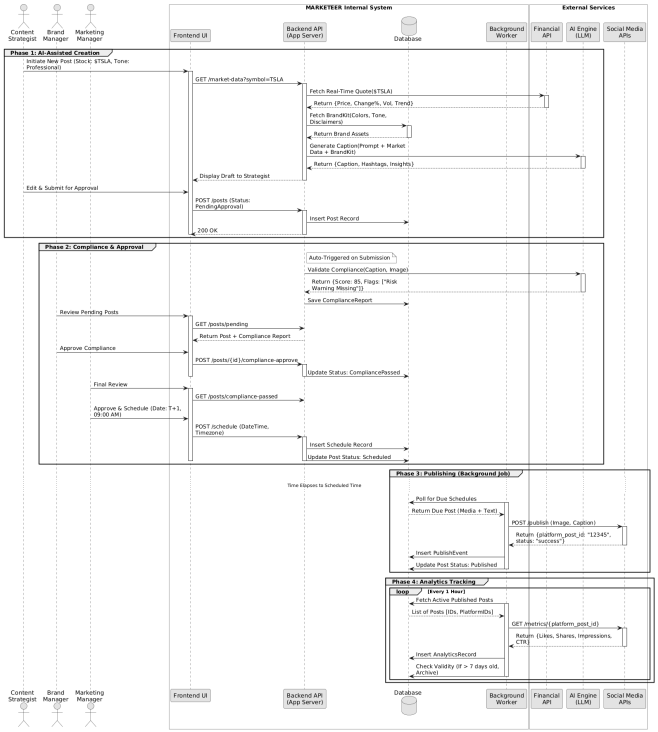


*Figure 4.4: Sequence Diagram*

## 4.6. Component Design

Marketeer has been developed in an organised manner for efficient control over the process involved in its operations.

- **Presentation Layer:** Consists of all UI elements and their processing logic.
- **Services Layer:** Primary access point for the system, and it does operations for logging, verification, and notification activities.
- **Business Layer:** The content processing logic and market data operations are performed with proper approval processes.
- **Data Layer:** Performs the database retrieval, file storing and exchange of information with external API services.
- **Crosscutting Infrastructure:** All common processes such as security measures, logging, monitoring, and background activities.

All the various layers interact with each other using predefined JSON structures for security purposes. For instance, the main server sends the JSON data object for the purpose of passing the details such as post ID and URL of the media to the integration layer, and from there it is forwarded to relevant social media sites.

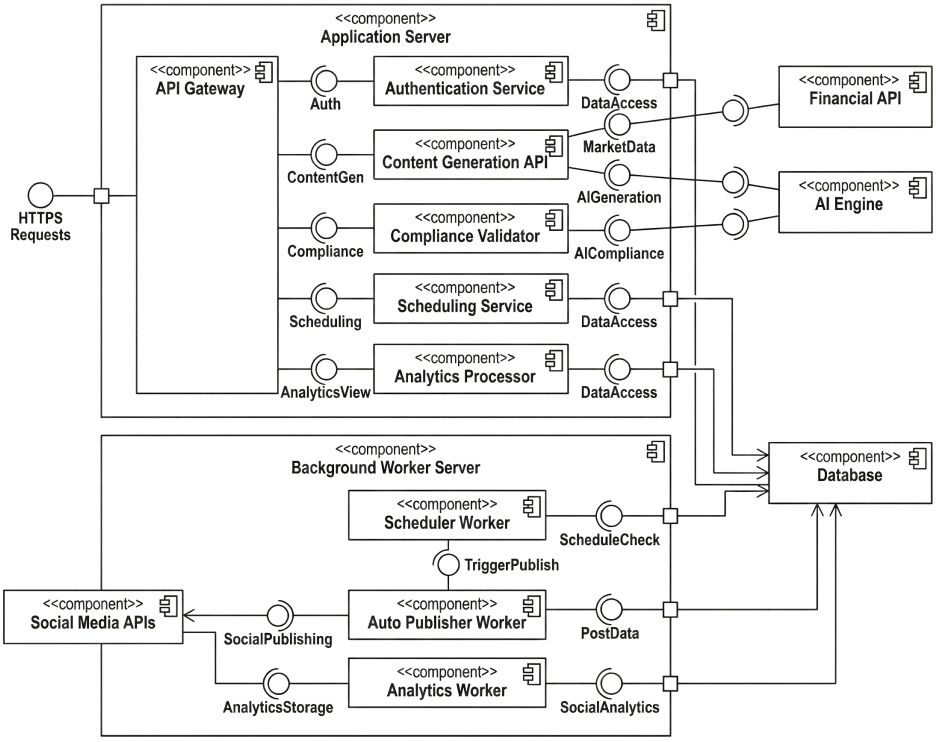


*Figure 4.5: Component Diagram*

## 4.7. Data Models

The data model guarantees that all system-related data is well structured and follows some criteria. It is necessary to make relational databases in order to regulate the relationship between all the mentioned data structures. Let us discuss the structure of our database:

- **User & Auth:** This subsystem is designed to organise user identity and store authentication data (Admin, Marketing, Content).
- **BrandKit:** Entity designed to link a user's personal style (color palettes, fonts, logos etc.) with the produced final product.
- **MarketData:** Storage of live market data used to improve system performance and reduce the number of API calls.
- **Post (Central Hub):** Central element of the database containing combined information from generated captions, media, and platform.
- **Schedule & PublishEvent:** Element regulating publication schedule of posts and time-stamping the exact moment of publication in social networks.
- **Compliance & Analytics:** Subsystem with compliance metrics for generated posts.

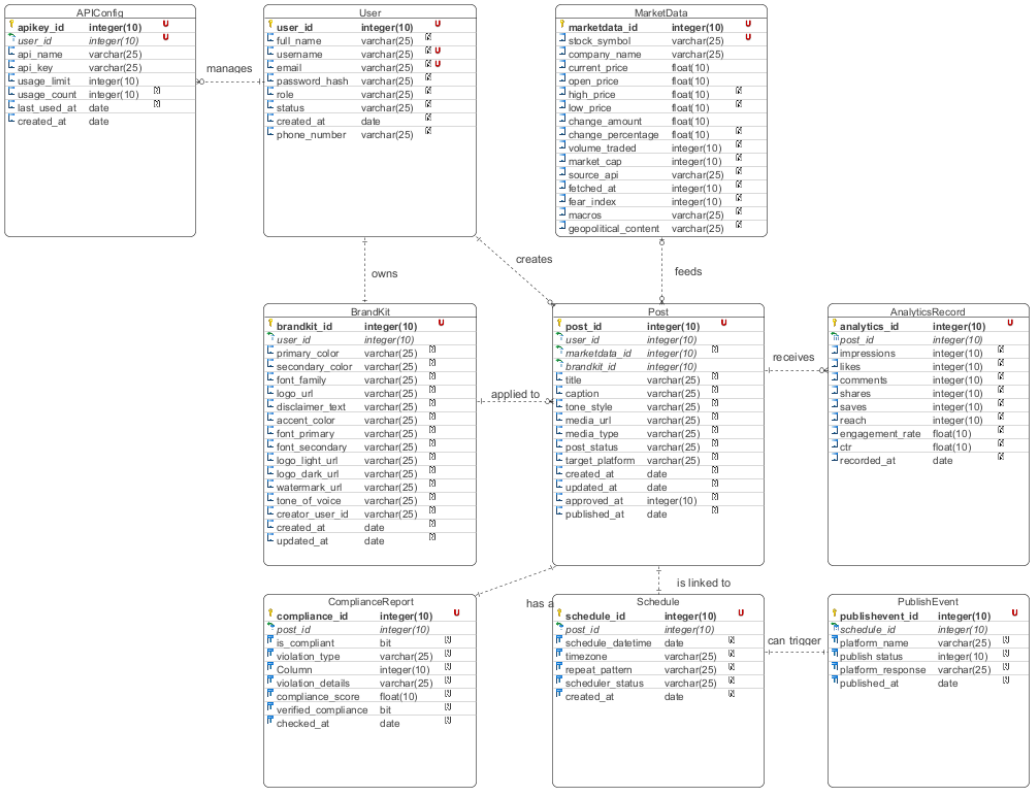


*Figure 4.6: Entity Relationship Diagram (ERD)*

## 4.8. User Interface Design

**Marketeer's** interface is built to be user-friendly both on the desktop and mobile. Its simple and consistent flow make it easy to operate for marketers who might not possess any technical knowledge.

### 4.8.1. Dashboard

It provides a common point for all activities. The most distinctive feature is that it changes according to the demands of its user. Thus, if you change from one mode to another, the icons, titles, and text fields will automatically change on their spots. Therefore, due to the lack of unnecessary layout shifting, the user is never left overwhelmed.

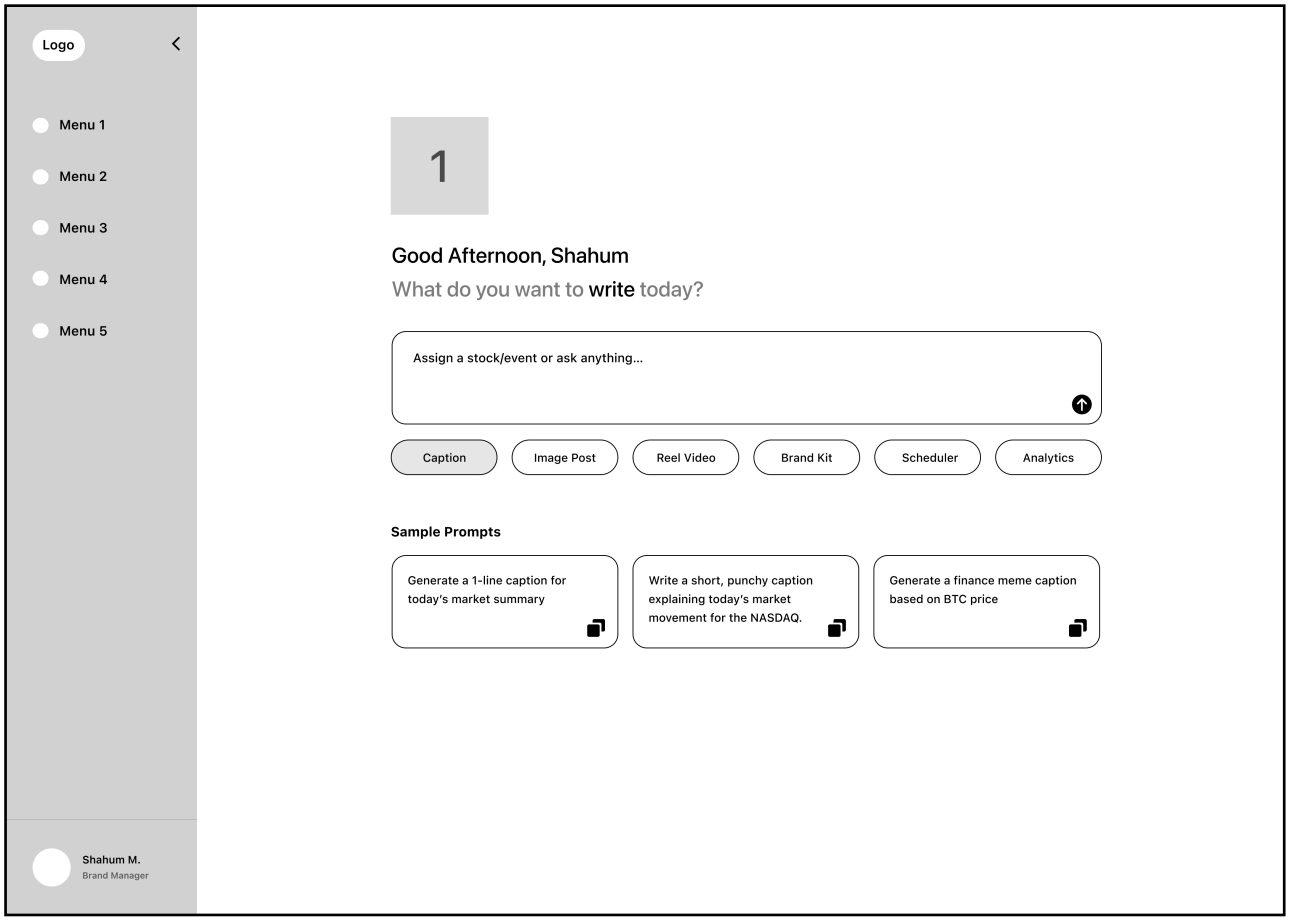


*Figure 4.7: Dashboard Wireframe*

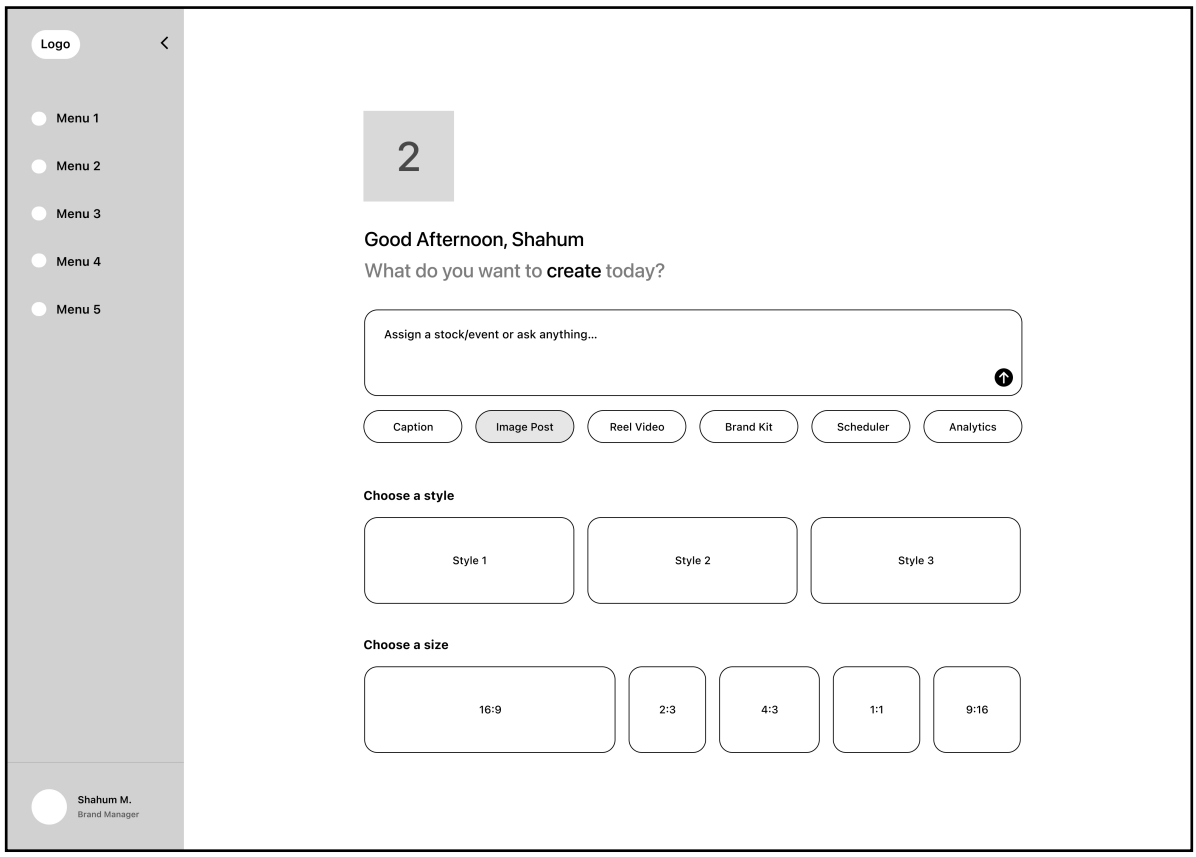


*Figure 4.8: Image Mode Wireframe*

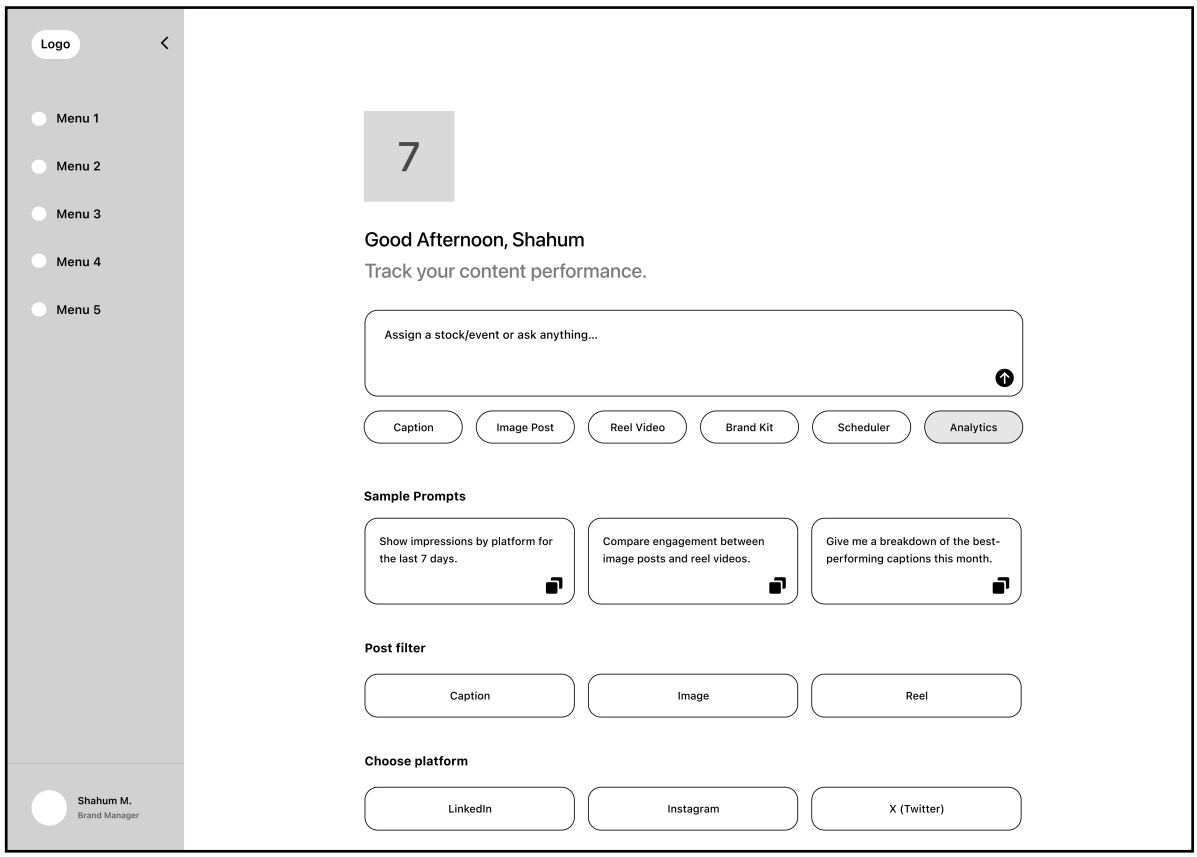


*Figure 4.9: Analytics Mode Wireframe*

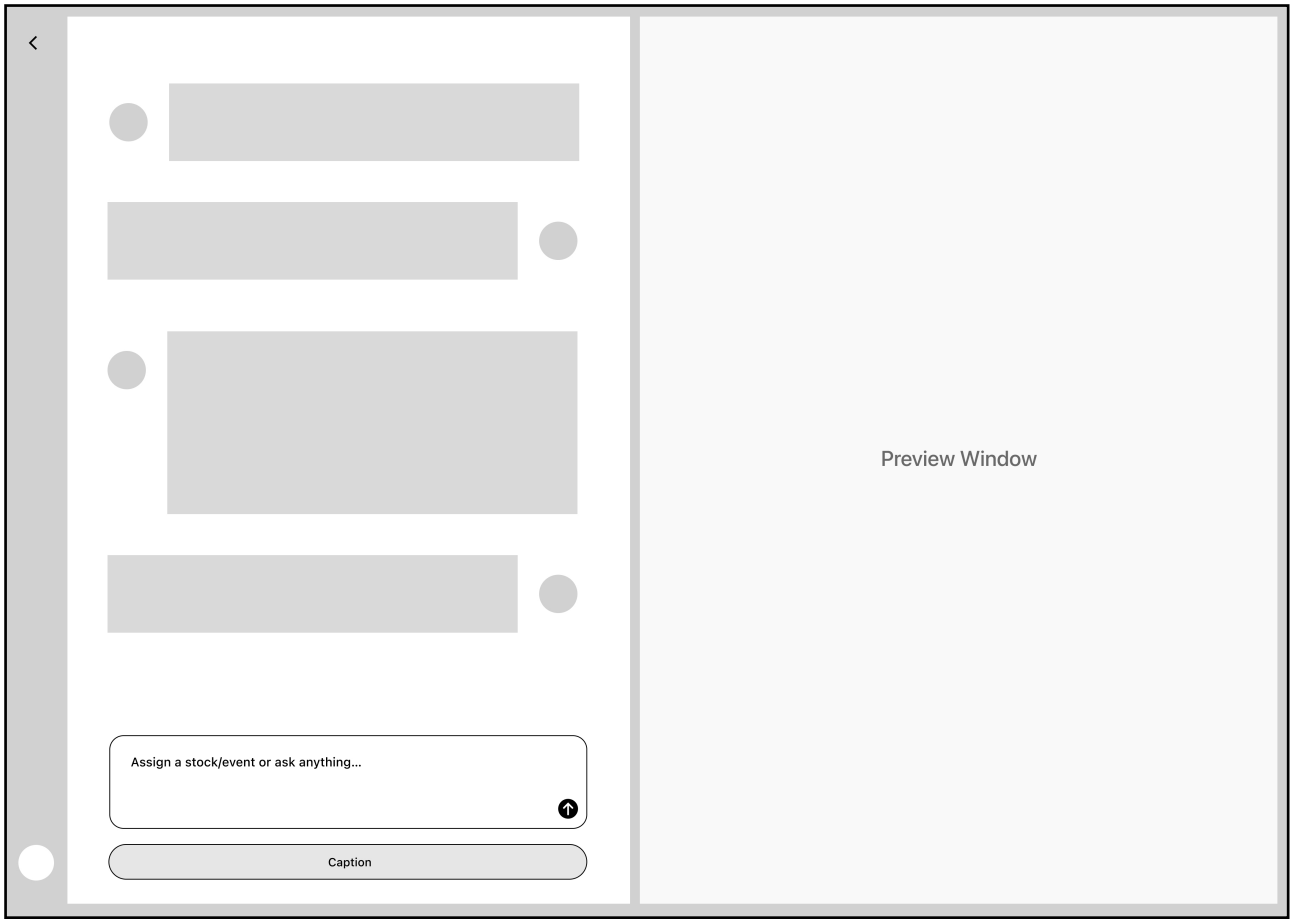


*Figure 4.10: Split-Screen Preview*

Through the split-screen function, you will have an opportunity to observe what is happening while editing the document. As a result, you will not have to keep shifting from one menu to another; rather, you can get everything done through one unified interface.

## 4.9. System Prototype

Taking all these ideas into consideration in the early stages of design, we have put together an entire user interface which includes several components. Unlike the old static designs, the final frontend is an interactive application made using React.js[2]. All the components adhere to the finalised design ideas, and can be expanded, collapsed, or customised as per the requirements. It means that you have the freedom to work as per your convenience with the well-organised and unified interface experience.

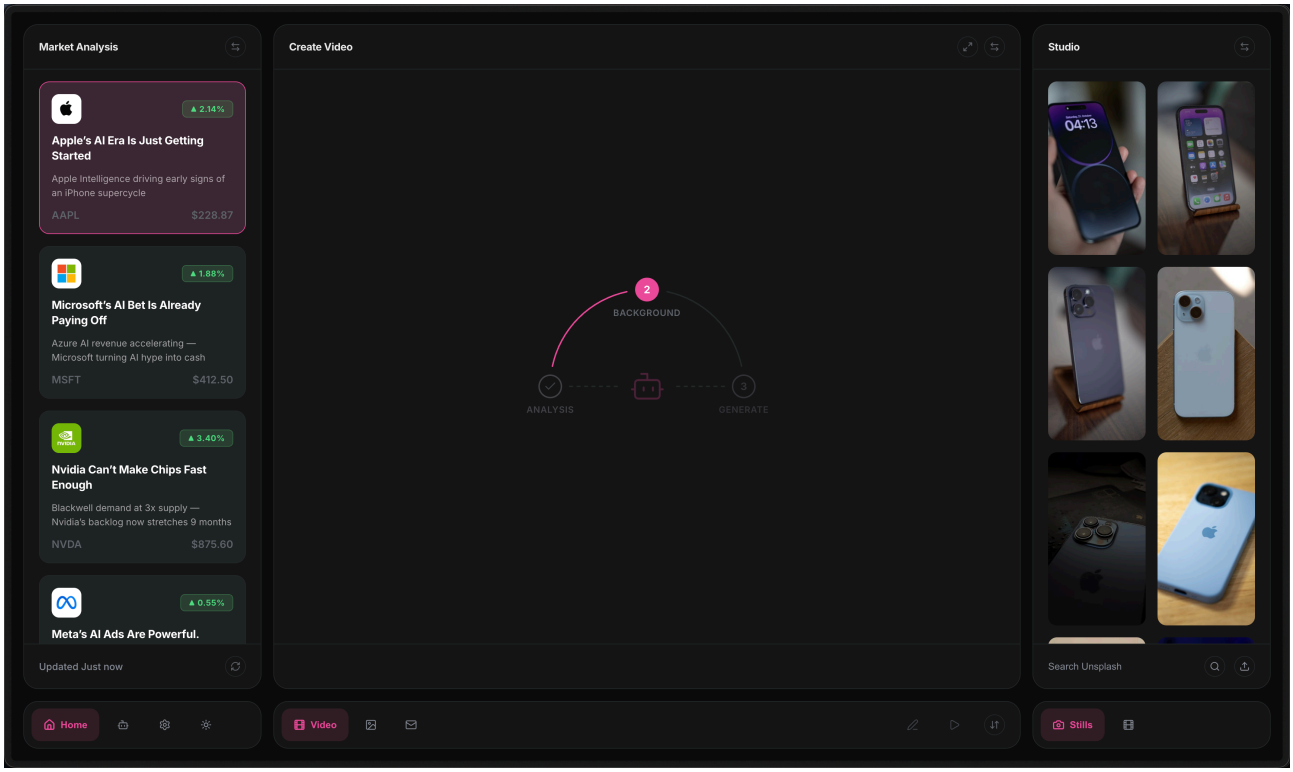


*Figure 4.11: Guided Flow*

The **Guided Flow** is located at the center of the dashboard, and it exists in two modes: Automatic and Manual. In case of **Manual** Mode, the UI guides through three tasks to be fulfilled: Analysis, Selection, and Generation. The progress bar indicates status accordingly.

For those users who want to shorten the time spent working, it will be enough to click on the bot located in the center of the screen, thus initiating **automated** work and skipping the need to configure anything manually. In this case, content relevance is checked by analysing trending data points.

### 4.9.1. Reel Generation

**Video Mode** gives you access to an all-rounded editor for the videos and allow you to add background photos or videos straight from Unsplash[12] and Pexels[13] API. It will create an automatic stock graph that will be updated real-time and the colours will change depending on whether there is an up-trend or down-trend on the stock.

The application uses the right brand logo based on the stocks you choose and also applies animations to it. You will have full control over the editor as you can move, resize, and edit any element within the frame.

For audio, ElevenLabs[6] assists you in getting a natural sounding **voiceover** as well as giving you the option to add **background music or sound effects.**

All elements appearing on the screen are editable according to your wish, starting from charts, logo images, text, and background colours. The editor can be used equally effectively by the users whether on their computer or mobile.

At the end, videos can be exported in various sizes and resolutions.

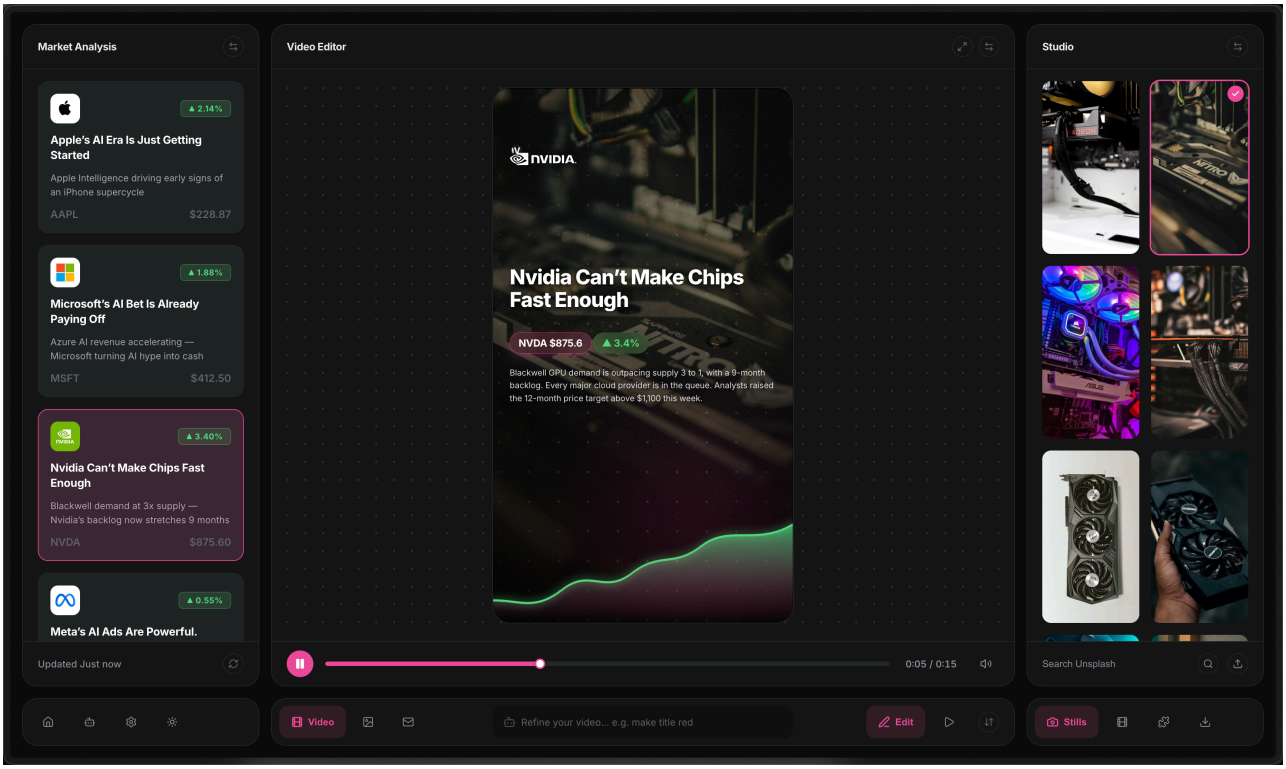


*Figure 4.12: Video Mode*

### 4.9.2. Image Generation

**Image Mode** offers the same level of editing capabilities as Video Mode. If you wish to bring in branded assets from **Figma**[14], then the application will automatically pick up on your colour palette and use that throughout content automation. Information about stocks, such as names, descriptions, icons, automatically fills in the placeholders in your template, while giving you full freedom to edit anything else that you wish.

Everything that you import into the application, templates, designs, is broken down into separate **layers**, each with their own toggle switch, allowing you to make changes anywhere in the design. Infographics, texts, logos, backgrounds, all are available for selection and customisation individually.

The background may be pulled in through the Unsplash[12] API or uploaded yourself, whichever way works for you. Choosing a stock immediately affects all of the content associated with it: titles, logos, descriptions, the entire design will update in less than a second.

At the end, images can be exported in various sizes and resolutions.

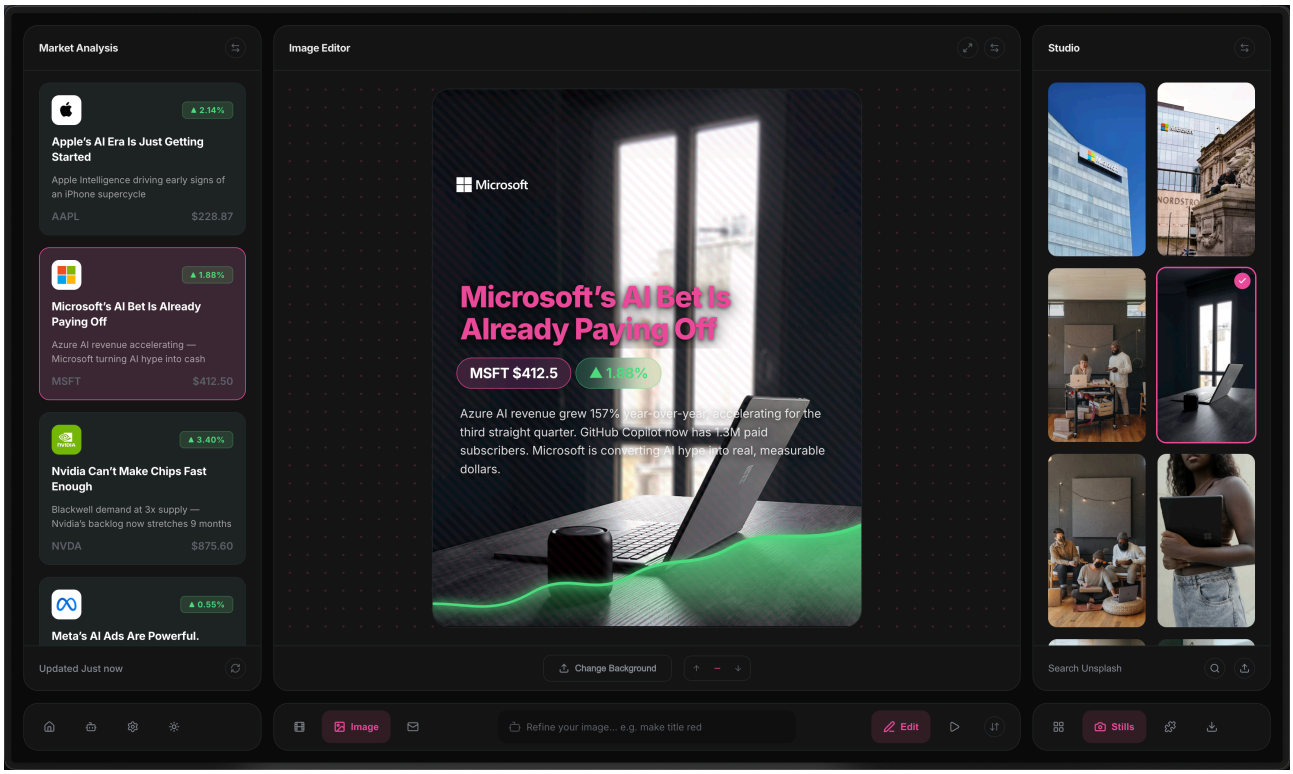


*Figure 4.13: Image Mode*

### 4.9.3. Newsletter Generation

In Newsletter Mode, your company branding, logo, and other configurations are fetched straight from your settings, eliminating the need for double configuration. With one button, it pulls the most current stock data using live APIs and creates a fully formatted newsletter.

You get an editor that offers real-time previews so that you can see how your newsletter will look on devices of various sizes, from mobile phones to tablets to computers. You may toggle each piece of content on and off, change its content, or alter colours and fonts to meet your branding needs from inside the editor.

This includes dynamic data such as name, description, icons, and images which will be automatically included in your template, but do not forget that you still have complete control on anything that you wish to change. Your content will dictate the way the template looks.

Once you are satisfied, you can send the newsletter right away to all subscribers without needing to open a new tool or copy content anywhere. This process takes place via the **Resend**[15] **API.**

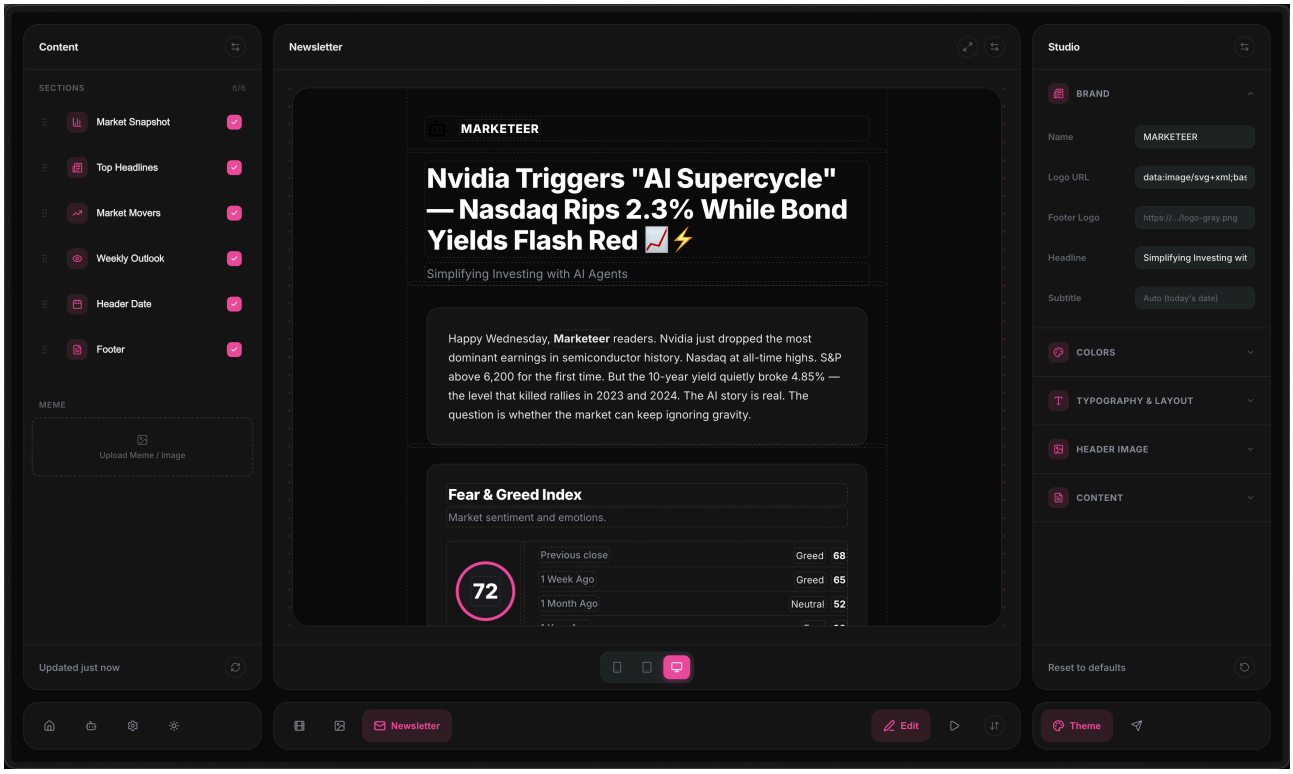


*Figure 4.14: Newsletter Mode*

### 4.9.4. Settings

**Settings** is designed to act as the control center for the whole system. You get to fill out everything about your brand and logos, colours, as well as any API keys you have access to. All of that data will go into every little piece of the system whenever it needs it.

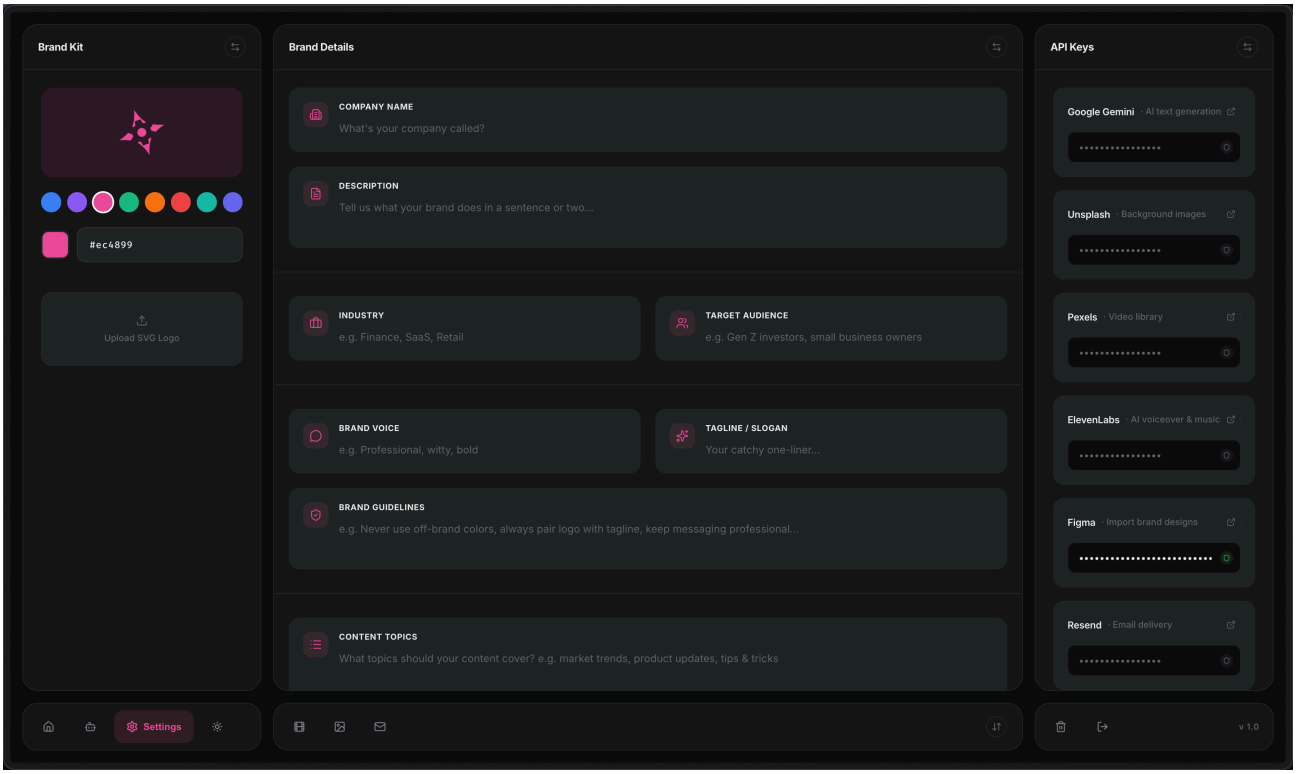


*Figure 4.15: Settings Page*

### 4.9.5. Automation Bot

**Ticker** is our bot designed with artificial intelligence and connected to the WhatsApp through OpenClaw[10] and works independently inside it. You are able to chat with Ticker or create content right from your WhatsApp interface, and get stock news feed live without ever needing to go into the dashboard. And, what's more amazing, Ticker even gives you the **links** for editing right on your mobile device that open Marketeer in **Edit Mode** so you can make changes on the fly.

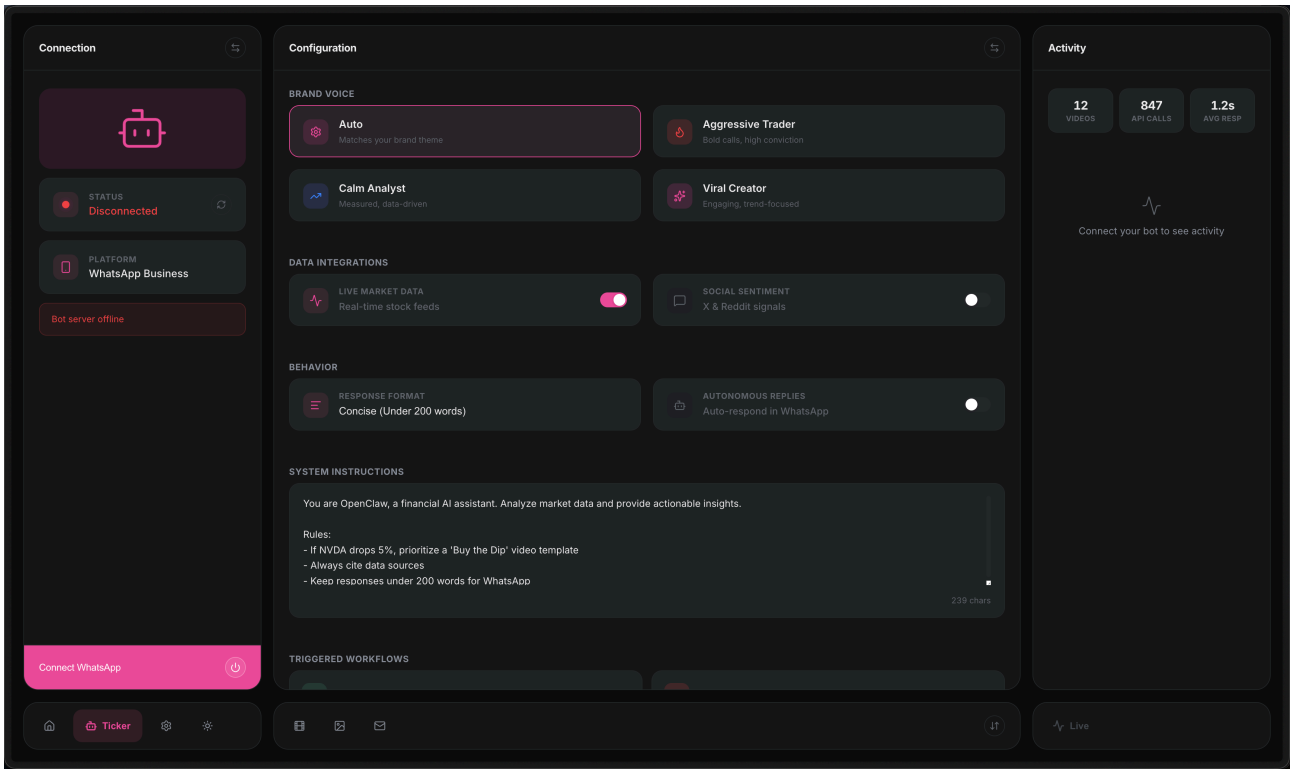


*Figure 4.16: Ticker Page*

### 4.9.6. Mobile Interface

The Mobile UI is the adaptive version of the modular panel structure, in which every panel of the desktop version corresponds to a separate full-screen page and provides the user with an automatic navigation process through these pages. In this way, every module of the desktop version is presented on a mobile device fully functional and easy to use without compromising the usability of the modules.

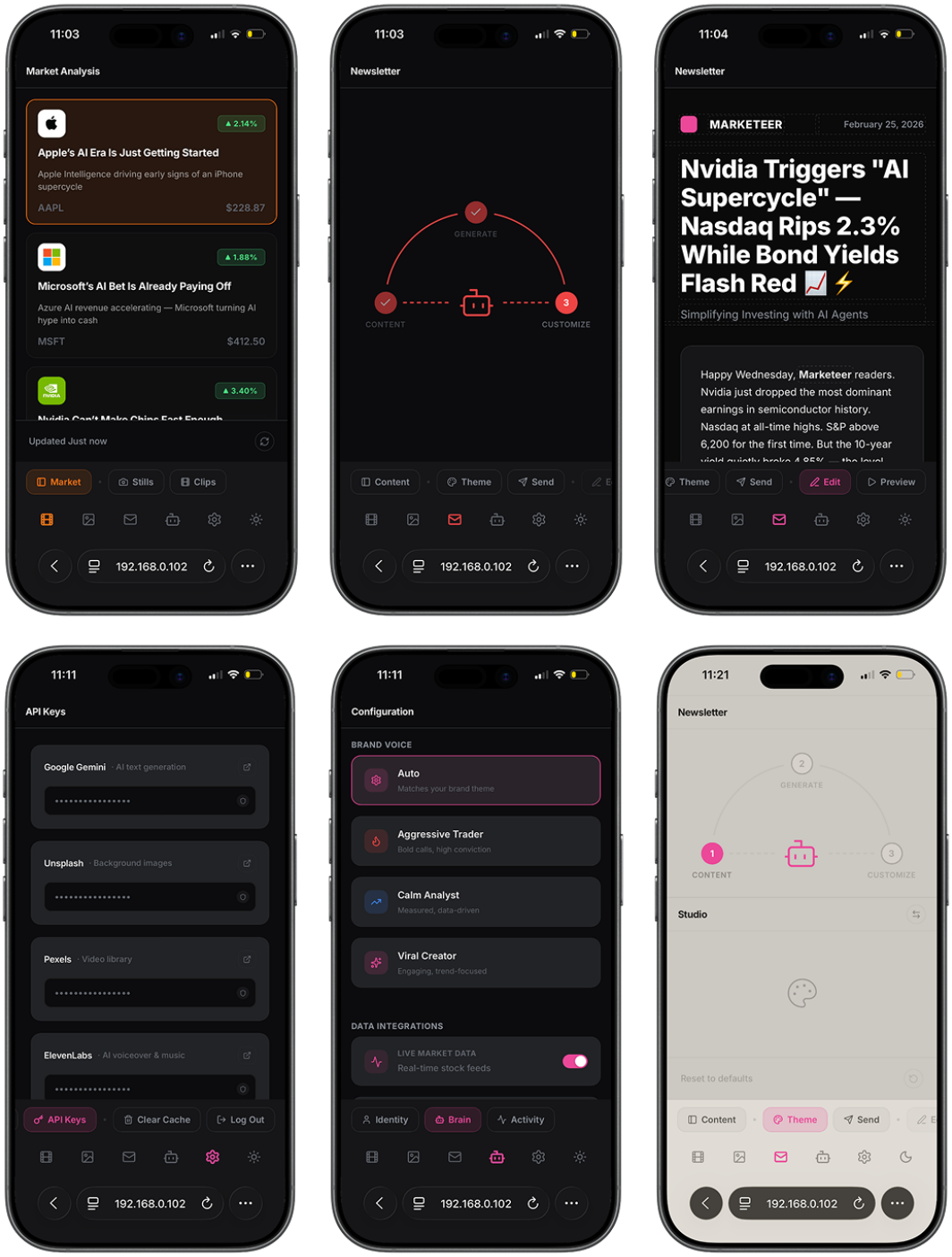


*Figure 4.17: Mobile UI*

## 4.10. Conclusion

This chapter covered all the aspects of the **Marketeer** application design. We have chosen the Modular Monolithic architecture to make sure that the user interface, business logic, APIs, and the back-end applications are independent of one another. The models, sequence diagrams, and the interface designs developed by us will serve as a good starting point for developing our application. The app will be capable of handling live data efficiently and will be highly user-friendly.

# Chapter 5

# System Implementation

This chapter will discuss the development process involved with the creation of the Marketeer software. We will be discussing the exact software, programming languages, and frameworks that were used to create this application. We will also talk about the agile methodology involved with the development process as well as the logic of the code used throughout our automation process.

## 5.1. Implementation Details

The deployment stage transformed our theoretical designs into an operational MVP running on cloud services. Our product was developed leveraging a state-of-the-art, open-source stack of technologies designed to integrate AI and programmatic media production.

### 5.1.1. Software and Technology Stack

The following core technologies were used by us to satisfy our requirements in the project:

- **Frontend Interface:** React.js[2] enabled us to create a responsive SPA interface that allows the users to navigate and preview their content and make it look like a studio experience.

- **Backend APIs/ Business Logic:** Node.js[4] was used for handling our web servers and API gateways. **LangGraph**[18] was used to enable us to build agentic workflows in which we could instruct **Gemini's**[17] lightweight AI language models to generate captions from data APIs and as per brand requirements and validate it through an agent made with LangGraph[18]. Additionally, we used **ElevenLabs**[6] to manage voice narration features. **OpenClaw**[10] for creating the whatsapp bot for control of the entire application from whatsapp directly.

- **Programmatic Media Rendering:** Bannerbear[5], Placid and Brandfetch[16] APIs, Remotion[1], Bannerbear[5] and Placid APIs are used to generate static and dynamic posts from templates. Remotion[1] is a library to programmatically render short-form video reels using React[2] code.

- **Database Management:** All our relational databases such as user session details, brand kits, post queue, platform analytics, etc., were managed with **Supabase**[3].

- Finally, for **version control and deployment**, we used GitHub to manage our source code and deployed the live application using cloud platforms like Vercel[11].

### 5.1.2. Development Process

The development process was done using an Agile Incremental model, where the entire development process was segmented into six two-week sprint cycles.

The **first sprint cycle** was dedicated to creating the architecture of the solution and developing the brand-kit ingestion module.

The **second sprint** was focused on the development of captions generation for the AI and static posts rendering capability.

The **third sprint** was focused on developing the video creation capability and voice-over integration.

In the **fourth sprint** cycle, we developed the backend activity scheduling functionality and integration of APIs for publishing posts to different social media platforms.

The **fifth sprint** was dedicated to the development of compliance guardrails and analytics tracking.

Finally, in the **last sprint** cycle, we did end-to-end testing, user acceptance testing, and solution deployment to the cloud environment.

### 5.1.3. Implementation of Key Workflows

We built the core functionality using distinct modules that operate independently. Here is a detailed look at how we coded, integrated, and validated these workflows.

#### 5.1.3.1. Database Configuration & Management

Data Layer carries all the truths concerning all the factors in the app. In executing all this, we relied on Supabase[3] all through; be it session management, brand kit, post queue, or even analytics.

In detail, we designed tables at the level of the relational database for the users' states and pipelines of content. In ensuring that all our processes concerning app data were done securely, we ensured that we used RLS or Row Level Security. Consequently, we restricted some data, for instance, posts and brand kits, to their respective owners only. More

importantly, we ensured that we had stringent foreign keys. This means that the deletion of an account automatically deletes all of its schedules.

```
CREATE TABLE posts (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    user_id UUID NOT NULL REFERENCES auth.users(id) ON DELETE CASCADE,
    market_data_id UUID,
    caption TEXT NOT NULL,
    media_url TEXT,
    status post_status DEFAULT 'draft'::post_status NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT timezone('utc'::text, now()) NOT NULL,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT timezone('utc'::text, now()) NOT NULL
);
ALTER TABLE posts ENABLE ROW LEVEL SECURITY;
```

*Figure 5.1: Database Configuration Code*

### 5.1.3.2. Data Ingestion & Caching Service

Our Market Data service integrates through an HTTPS connection to external endpoints for obtaining financial data, such as those from Alpha Vantage[7] and MarketVerse APIs. These are used to provide the real-time financial data that will be used to create the content.

To make these GET requests through HTTP, we have made use of Axios. We are also caching the data from the financial endpoint into Supabase[3] to prevent hitting the API limit and to ensure that the frontend page renders in less than 500 milliseconds. In the event where a user requests information on a stock ticker, which has been cached in the last 5 minutes, the cached JSON will be returned by our application.

```
{
    id: "aapl-001",
    ticker: "AAPL",
    heading: "Apple's AI Era Is Just Getting Started",
    description:
      "Apple Intelligence features are now live on 200M+ devices, driving early upgrade
momentum. Analysts expect the biggest iPhone supercycle since 5G. App Store revenue beat
estimates by 9% this quarter.",
    highlight: "Apple Intelligence driving early signs of an iPhone supercycle",
    signal: "BUY",
    searchQuery: "iPhone smartphone",
    change: 2.14,
    price: 228.87,
  },
```

*Figure 5.2: Database Ingestion Code*

### 5.1.3.3. AI Orchestration & Prompt Construction

Instead of using simple prompts to generate a response from an LLM, we have leveraged LangGraph[18]. It allowed us to leverage structured prompt templates and agentic workflows that our AI model adhered strictly to brand compliance policies. The dynamic prompt template injects three key variables into the main prompt: live market data, additional instructions from the user, and guidelines for brand voice. That ensures the AI generates captions that are consistent with the identity of our brand. We have broken down the workflow process into multiple nodes.

- **Step 1 functions** as Drafter and drafts the caption.
- **The second node** works as a Compliance Reviewer. Its task is to review the draft and ensure that it contains all the mandatory disclaimers.
- If it discovers that the draft lacks any required disclaimers, then it sends the caption back to the drafter via a conditional edge.
- Finally, the output parsers ensure that the text is generated in the form of a strict JSON format, segregating caption text from headline image.

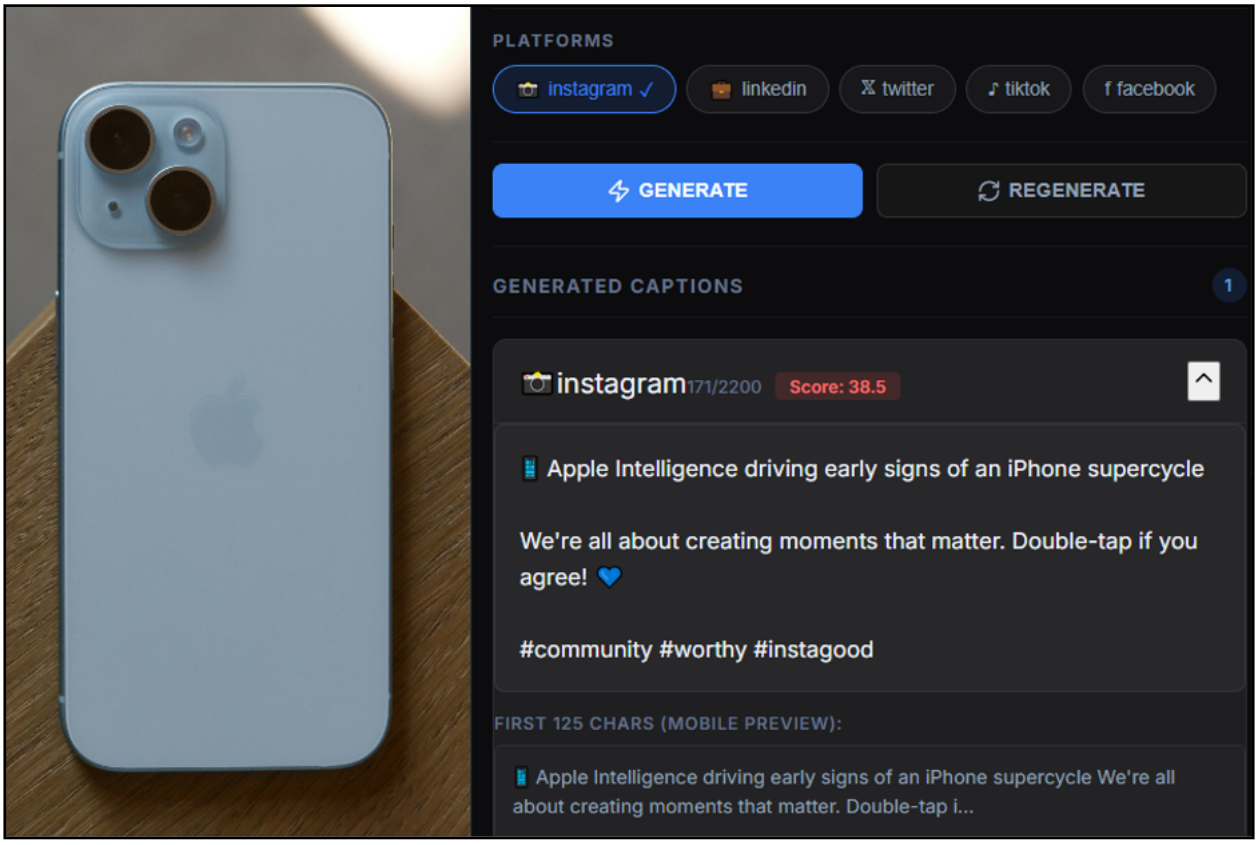


*Figure 5.3: Prompt Output*

```python
def compliance_check(state: CaptionState) -> Dict[str, Any]:
        """Check captions for compliance and quality"""
        metrics = {}
        for platform, caption in state['final_captions'].items():
            score = CaptionMetrics.calculate_score(caption, platform,
state['brand_profile'])
            metrics[platform] = score
            logger.info(f'Compliance {platform.upper()}: {score["overall"]:.1f}/100')
        return {'quality_metrics': metrics, 'status': 'done'}

# Build graph
    try:
        graph = StateGraph(CaptionState)
        graph.add_node('generate', generate_captions)
        graph.add_node('comply', compliance_check)
        graph.add_edge('START', 'generate')
        graph.add_edge('generate', 'comply')
        graph.add_edge('comply', 'END')
        compiled = graph.compile()
        initial = CaptionState(
            post_type=post_type,
            platform_targets=[p.lower() for p in platform_targets],
            post_topic=post_topic,
            post_media_url=post_media_url,
            brand_profile=brand_profile,
            final_captions={},
            quality_metrics={},
            status='running'
        )
        result = compiled.invoke(initial)
        passed = all(m['overall'] >= 60 for m in result['quality_metrics'].values())
        return {
            'success': True,
            'final_captions': result['final_captions'],
            'quality_metrics': result['quality_metrics'],
            'compliance_passed': passed,
            'mode': 'langgraph'
        }
```

*Figure 5.4: AI Orchestration Code*

### 5.1.3.4. Media Assembly & Programmatic Rendering

For automation of the graphics design aspect, our **Content Generation module integrates the output text from the AI engine with the rendering API.** As soon as the AI generates the content, our backend automatically fetches the user's brand kit information. This data is used to map the variables such as colors, fonts, logos, and AI-generated headlines into a JSON object that we POST to **Remotion**[1] for video reels, or **Bannerbear**[5]**, Placid, and Brandfetch**[16] for static images. Once the rendering process completes on their end, we just wait for them to return a **CDN URL for the output image or video via whatsapp.**

### 5.1.3.5. Frontend User Interface Development

We built the client-facing application using React.js[2] and Next.js. This provided a fast and highly responsive single-page application experience for the users.

A **modular panel system** is used on the frontend, **wherein each panel is a standalone element.** Updates to these elements occur separately from the backend. Consequently, the website interface allows us to adapt to modern trends or track bugs without affecting other modules in the program. That explains why our website looks absolutely uniform throughout the entire process of interaction. Stability in layout has been our utmost priority. Marketeer avoids random layout shifting.

The panels move only when you wish to **rearrange** them according to your personal preferences. Every single panel adapts to your needs as you progress with Video, Images, Newsletter, or Scheduling. These features have been incorporated in the mobile version of Marketeer as well. To adapt the interface to smaller screens, we decided to turn the panels into separate pages with clear step-by-step guided flow.

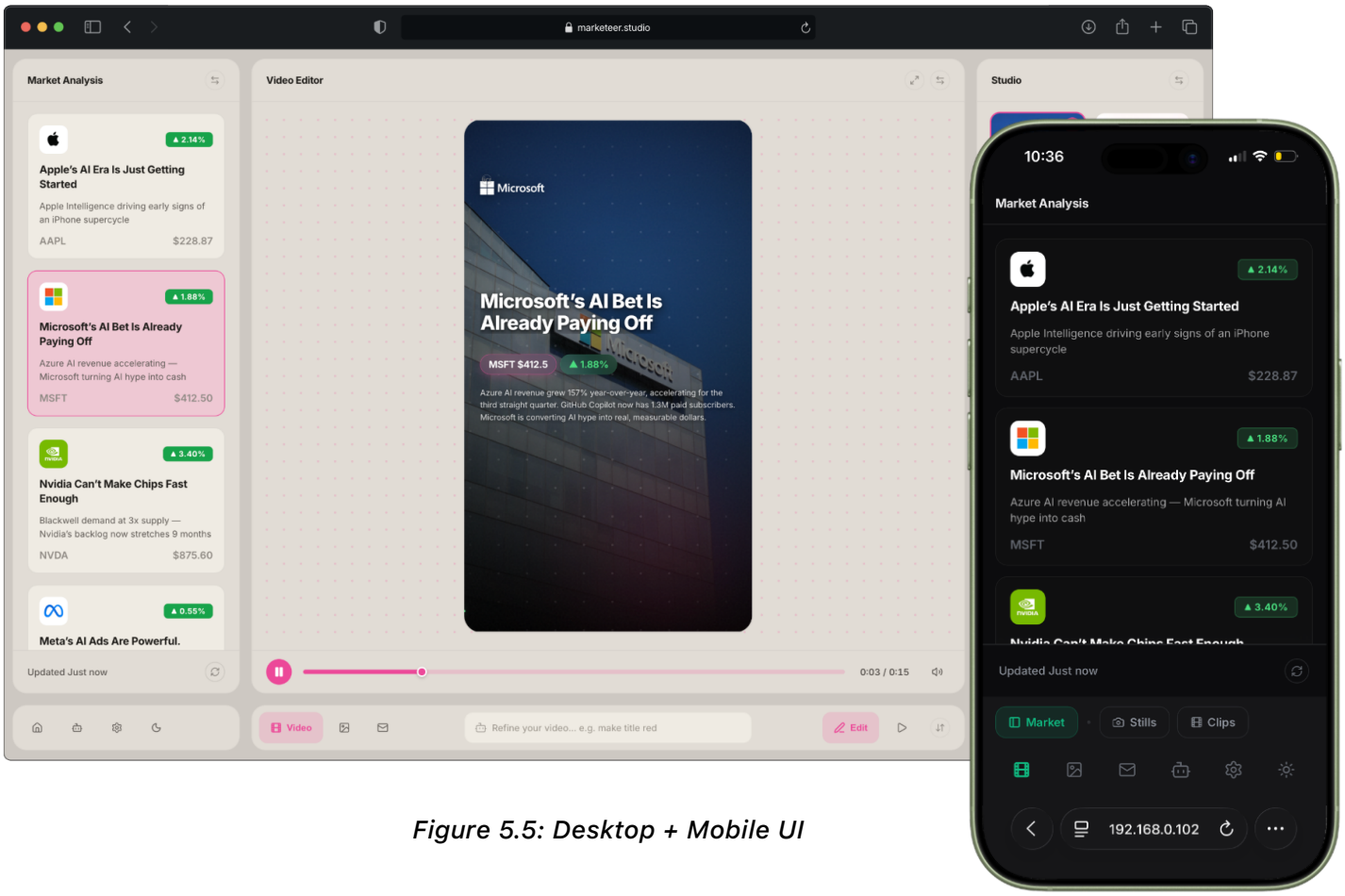


*Figure 5.5: Desktop + Mobile UI*

### 5.1.3.6. Asynchronous Publishing & Scheduler Backend

In order to schedule the auto-publishing of approved posts, we have employed an asynchronous worker\scheduling API in our scheduling system.

We have implemented a cron job in Node that implements a polling algorithm. It queries the database once every minute for all those posts that have their scheduled date and time equal to or less than the current date and time, along with their status being approved. If found, then the correct authentication tokens are fetched and POST requests are sent to the relevant social media API endpoints, such as LinkedIn and X. The response to the request is logged directly in the database, and the status of the post is updated to published.

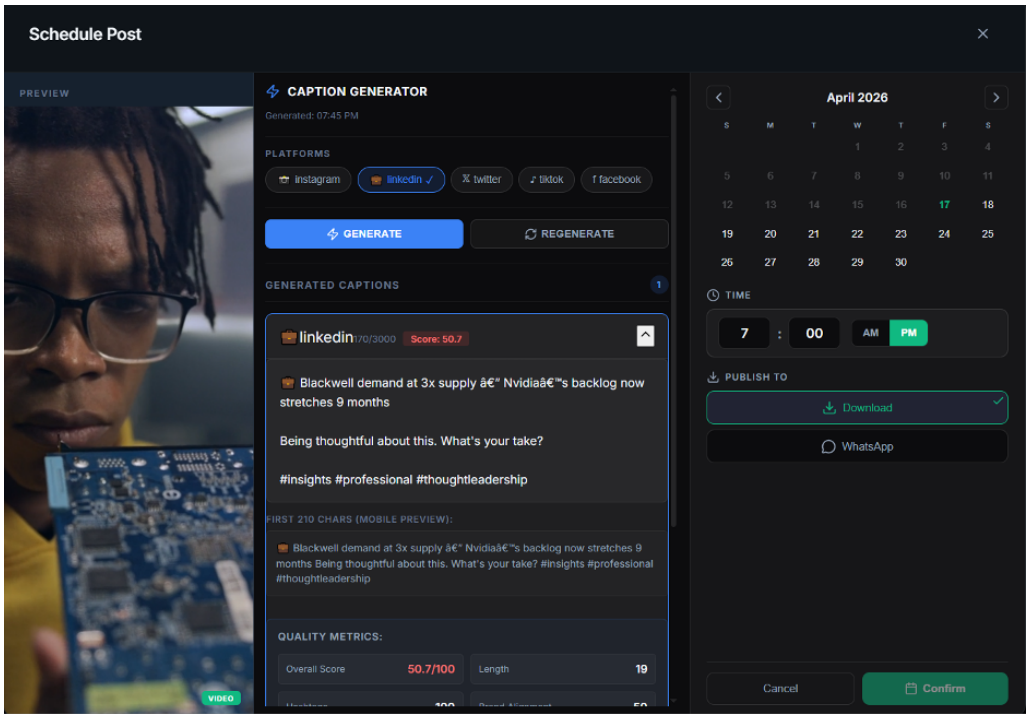


*Figure 5.6: Post Scheduling*


*Figure 5.7: Scheduled Posts*

### 5.1.3.7. WhatsApp Bot

We built Ticker as an an AI-bot on WhatsApp powered by OpenClaw[10] for the purpose of using the app directly from the comfort of your phone. Get alerts, detailed info on a finance related topic, create and schedule posts and much more.

It runs with the website server, reads WhatsApp messages and gives responses and executes commands on the server, the intent handling for the bot is being done through Gemini's[17] lightweight API so responses feel less AI and more direct as a person and relevant to the topics.

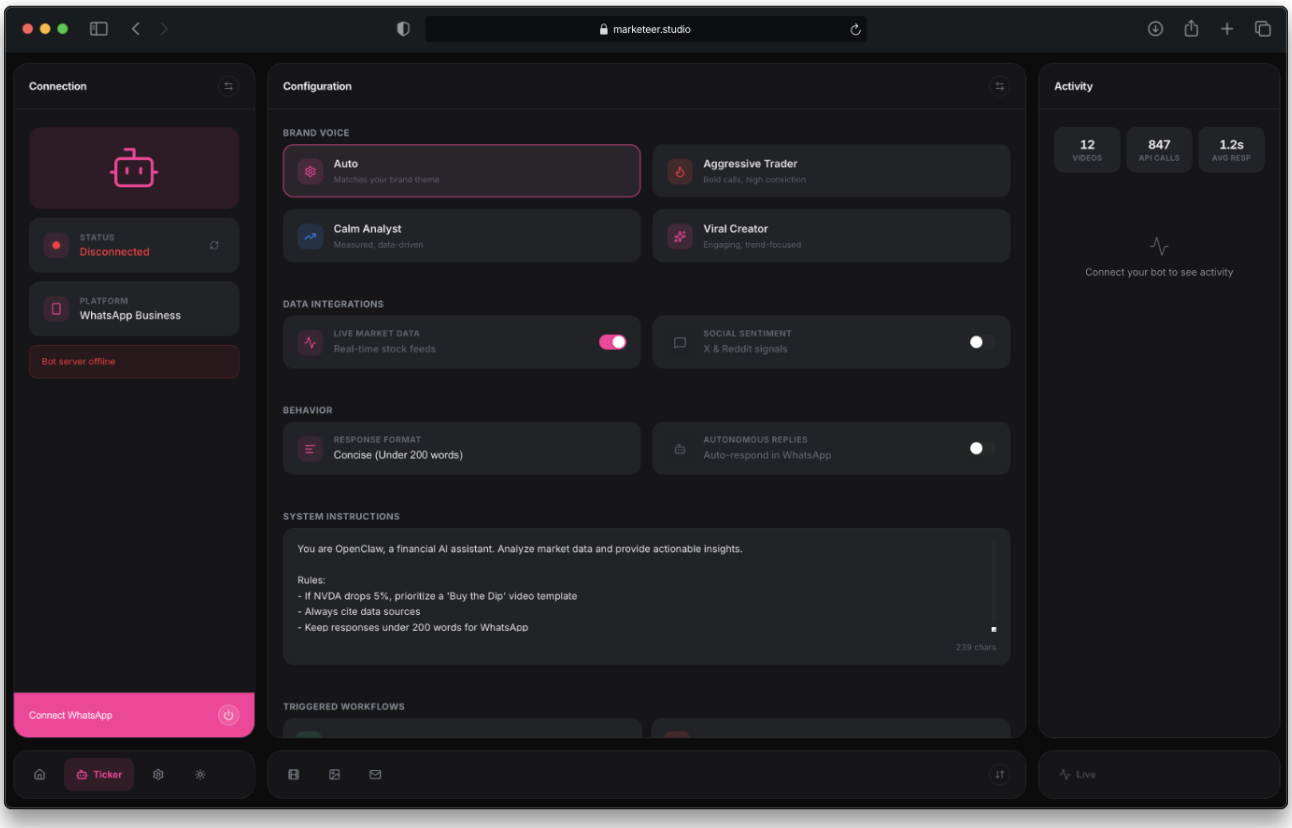


*Figure 5.8: OpenClaw Integration*

It utilizes and takes knowledge from the attached APIs rather than the LLM itself which takes more time and is more error prone. We've also allowed Ticker tone adjustment for better control of its behaviour in real-time.

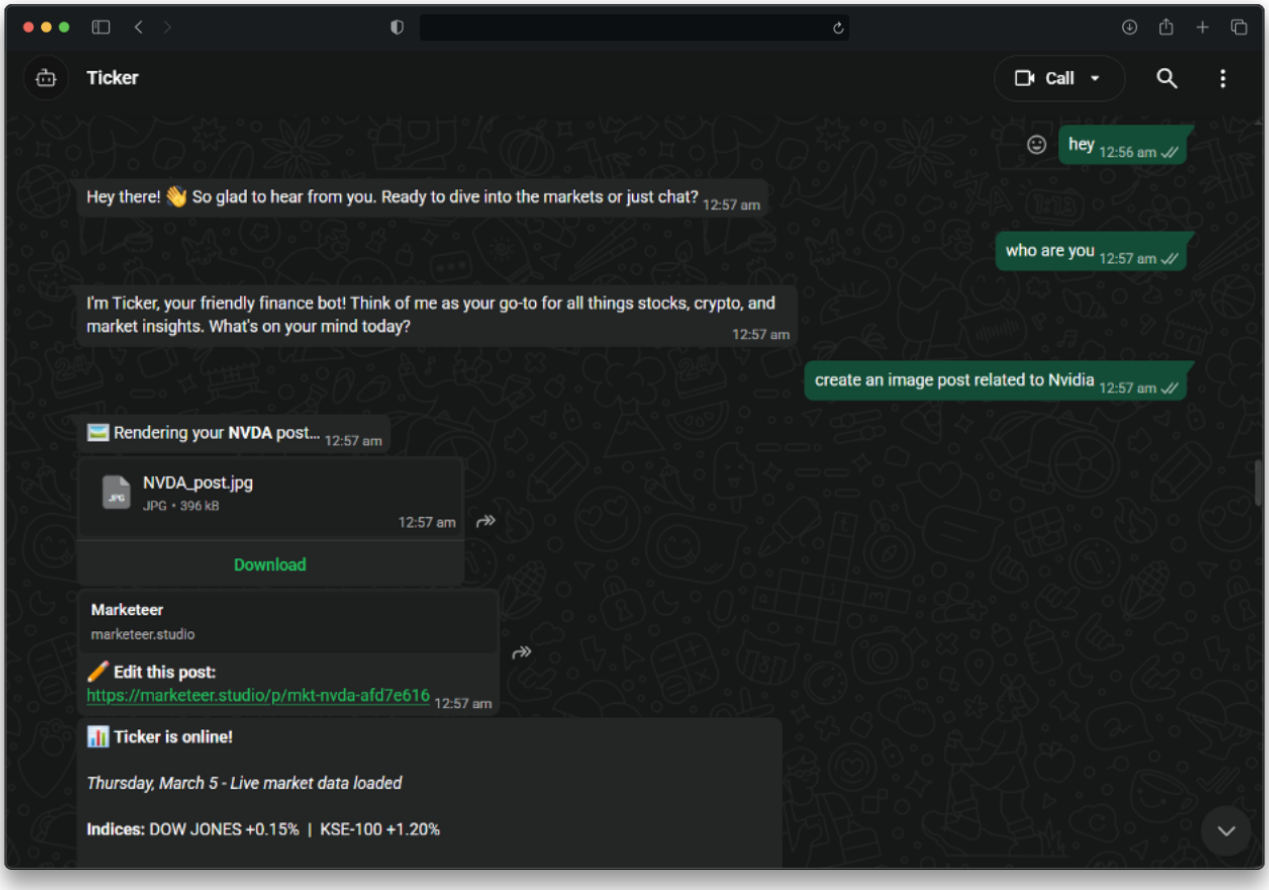


*Figure 5.9: Conversing with Ticker*

## 5.2. Conclusion

The construction phase led to the effective implementation of Marketeer. We managed to automate the design process instead of doing everything manually thanks to the employment of Gemini[17] in generating AI texts, Remotion[1] in video production, OpenClaw[10] for on-the-go automation and much more. Our Agile methodology allowed us to keep all the integration phases, financial, AI, and social media posting included, stable through the whole process.

# Chapter 6

# System Testing & Evaluation

This chapter provides a description of the methodology for verifying the Marketeer operations. It presents our testing approach, specifies our testing stages that include component testing, integration testing, system testing, and usability testing, and finally points out the key test cases needed to verify the possibility of automatic content creation.

## 6.1. Test Strategy

The predictable and non-predictable elements of system behaviour were included in the marketeer testing. Predictable tasks such as scheduling, database synchronisation, and compliance testing were conducted with the help of predetermined test cases. For the non-predictable tasks, the testing process involved the examination of content for its quality and consistency with the brand language, accuracy based on the data provided by the market, and the presence of necessary disclaimers.

The testing process involved four main stages, including **components, unit, integration, and system testing.** This approach entailed the progressive enhancement of software quality, from simple user interface elements and backend processes to complete end-to-end workflows. A user acceptance testing session was also scheduled in this project, during which feedback would be collected with the help of questionnaires targeting marketing professionals in fintech companies.

All tests were carried out in a safe development environment, using the live sandbox API credentials when appropriate. In other cases, mock data was used. Testing outcomes were manually documented and compared with the exact specifications provided in **Chapter 3.**

## 6.2. Component Testing

We used a modular testing strategy for the React.js[2] frontend to ensure that each component was operational individually before it was added to the entire application.

We paid special attention to critical components such as the **content creation**, **brand kit** settings, and the **analytics** dashboard. Key tests include:

1. Verified that the "Generate" trigger stays inactive until the selection of either stock or market event; otherwise, an API request would be issued with no input parameters.
2. Validated that the Colour Picker will trigger immediate rendering based on the state changes in the React[2] component without any need for refreshing the page.
3. Validated that the layout sort is done based on vertical/square layouts proportionally with respect to different post types.

In order to ensure that the correct formatting of the input data is done before being transferred to the server side, edge cases testing is done for the Settings tab and Brand Kit tabs. These include:

1. Acceptance of invalid HEX codes and logos.
2. Data validation before submitting the input data to the server side, including API key validation.
3. Error messages display for failed data validation.

All the test cases have been passed successfully.

## 6.3. Unit Testing

For testing smaller functions of the Python and Node.js[4] backend, we used unit testing. The main idea was to test the inner logic before connecting with other parts of the system. There were three main aspects of testing that were taken into account.

1. The transformation logic for the raw JSON data fetched from the MarketVerse and Alpha Vantage[7] APIs was validated to make sure that the data mapping process of tickers, prices, percent changes, and market status strictly adheres to data types.
2. To validate the prompt engineering heuristic, we examined whether it is able to assemble the prompts by combining the live market data, stylistic tone, and brand guidelines. String truncation prevention and omission of necessary data fields was checked.

3. The validation process of the compliance heuristic was conducted using a wide range of data captions. In particular, we made sure that the caption contains all the necessary information (disclaimer and no fragmented text).

## 6.4. Integration Testing

Integration testing was conducted to verify the functionality of Marketeer's internal modules with their external dependencies. This stage tested the flow of data in a distributed environment.

#### AI Caption Generation Pipeline

A comprehensive test run involved simulating an entire request workflow whereby the primary application server kicked off a request in the Orchestration Layer (Gemini[17]/ OpenClaw[10]), which proved successful in:

1. Addition of stock symbol in the input.
2. Generation of output in the form of caption generated by the AI, Hashtag created, and the Regulatory Compliance Score.
3. Low latency ensured a timely processing of all requests despite time constraints.

#### Visual Rendering Pipeline

Testing ensured the Content Generation module would be able to effectively integrate with visual content rendering engines.

1. Verified the proper serialisation of accepted captions and brand kits into JSON format for passing along to BannerBear[5]/Remotion[1].
2. Successful validation demonstrated a fully functioning URL endpoint producing media assets consistent with the designated brand kit, including exact match colors and logo overlays.

## 6.5. System Testing

Marketeer went through rigorous testing in evaluating its efficiency in conducting all operations without errors. The tests covered all areas of operations which include but are

not limited to user verification, stock picking, auto-generation of captions, validation, media rendering, then finally scheduling in the social media.

### Performance Metrics

Marketeer proved to be very efficient in doing all its operations after several test runs utilising different ticker codes and brand configuration:

1. It takes 10-30 seconds for the complete generation of a validated media object starting from stock selection to completion of media rendering.
2. This result conforms to the standards required in terms of performance metrics as specified in **Section 3.3.1** of 60 seconds or less.
3. Posts are scheduled within 60 seconds following the selected time stamp and queued automatically.

### Fault Tolerance Testing

To evaluate the fault tolerance capabilities of Marketeer under unfavourable circumstances, we performed several tests:

1. An environment where the system had no connectivity with market data providers was simulated. The system handled the failure appropriately by throwing an exception without crashing and alerting the user about it.
2. We evaluated how the system handles erroneous text inputs and image uploads made by the user. The request was appropriately rejected by the Compliance Validator and other filtering services.
3. The re-try reaction of the Orchestration Layer in case of packet loss while transmitting data packets through network connections was tested.

## 6.6. Test Cases

To make sure that the system satisfies both its functional and non-functional requirements, a series of tests have been developed. This testing covers the important cases, edge cases, API connections, and security considerations of the Marketeer system.

### 6.6.1. Test Case #1: Real-Time Market Data Ingestion & Prompting

*Table 6.1: Test Case 1*

| | |
|---|---|
| **Test Case ID** | **TC-01** |
| **Objective** | Confirm the system pulls live market info and adds it to the AI prompt. |
| **Pre-Conditions** | Make sure the MarketVerse API key is set up and you are logged in. |
| **Test Steps** | 1. Log into the dashboard and enter a stock symbol like TSLA.<br>2. Click the "Generate Caption" button.<br>3. Verify that the data is sent over to the AI Engine. |
| **Expected Result** | The data should include the latest price and the percentage change from the last 5 seconds, following the brand rules. |
| **Actual Result** | It took the software about two seconds to retrieve the live data of the stock market from the ticker symbol. The AI prompt was provided with the latest price, change in percentage, and market events. Therefore, the output is accurate with live figures. |
| **Status** | Pass |

### 6.6.2. Test Case #2: Brand Kit Compliance Validation

*Table 6.2: Test Case 2*

| | |
|---|---|
| **Test Case ID** | **TC-02** |
| **Objective** | Make sure the compliance tool spots content that doesn't follow the rules and warns the user. |
| **Pre-Conditions** | Set up a Brand Kit that makes financial disclaimers mandatory for every post. |
| **Test Steps** | 1. Use the AI to create a caption and then manually delete the financial disclaimer from the text.<br>2. Try to publish the post. |
| **Expected Result** | The system should block the post and display a message stating that the financial disclaimer is missing. |
| **Actual Result** | Publishing the caption without mentioning the disclaimer failed because the system prevented this. An error message indicated a problem with violating the policy and missing the disclaimer. The caption was still in the Draft stage and could not proceed to the scheduler stage. |
| **Status** | Pass |

### 6.6.3. Test Case #3: Programmatic Visual Rendering

*Table 6.3: Test Case 3*

| | |
|---|---|
| **Test Case ID** | **TC-03** |
| **Objective** | To validate the creation of static images and video reels using external APIs. |
| **Pre-Conditions** | Caption should be validated and colors and logo of Brand Kit to be stored. |
| **Test Steps** | 1. Select 16:9 Image Template.<br>2. Tap the 'Generate Content' button. |
| **Expected Result** | The call placed to either the BannerBear[5] or Placid API will generate an image that accurately matches the text, colours, and the brand logo. |
| **Actual Result** | The API was capable of generating an image of 16:9 ratio within 8 seconds. The generated image featured the right color of the brand, the logo, and also the caption. |
| **Status** | Pass |

### 6.6.4. Test Case #4: Automated Scheduled Publishing

*Table 6.4: Test Case 4*

| | |
|---|---|
| **Test Case ID** | **TC-04** |
| **Objective** | To confirm that the background system follows through and successfully triggers the ‘send’ button for approved posts at the exact scheduled time. |
| **Pre-Conditions** | A post must be fully approved and scheduled to go live exactly two minutes from the current time. |
| **Test Steps** | 1. Wait for the clock to hit the precise scheduled time.<br>2. Monitor the database logs and the connected social media accounts for activity. |
| **Expected Result** | The system should publish the post automatically. The database will record a ‘success’ status alongside a platform ID, and the post will appear live on the designated social feed. |
| **Actual Result** | It was able to find the scheduled post in less than one minute after which it was able to deliver it to the intended social media platform through the API. This made the database change to the status of published and the post identifier was registered in the database. |
| **Status** | Pass |

### 6.6.5. Test Case #5: Security and Account Lockout

*Table 6.5: Test Case 5*

| | |
|---|---|
| **Test Case ID** | **TC-05** |
| **Objective** | To verify whether the system triggers the brute-force attack protection procedure to prevent any kind of illegal access. |
| **Pre-Conditions** | A user profile must have already been added to the system's database. |
| **Test Steps** | 1. Go to the login screen.<br>2. Enter a correct username but an incorrect password three times consecutively.<br>3. Attempt to log in again by entering the correct password a fourth time. |
| **Expected Result** | The system should display an error message after three failed attempts and block the account for 1 hour. The fourth attempt should not succeed due to the temporary lockout, even when the correct password is used. |
| **Actual Result** | Because there had been three failed attempts at logging in successively, a notification was generated by the system, and no further login attempts were entertained for one hour. When I tried logging in again using my correct credentials, even then, my login attempt was not accepted. |
| **Status** | Pass |

### 6.6.6. Test Case #6: External API Failure Handling

*Table 6.6: Test Case 6*

| | |
|---|---|
| **Test Case ID** | **TC-06** |
| **Objective** | To verify that the software can handle external service interruptions gracefully without crashing. |
| **Pre-Conditions** | To verify that the software can handle external service interruptions gracefully without crashing. |
| **Test Steps** | 1. Query the market performance of a specific stock through the Dashboard.<br>2. Monitor the system's attempt to retrieve the required data. |
| **Expected Result** | This is because the system must recognize the error, alert the user regarding it, and yet still be operational. For example: “Market data is currently not available. We’re trying to get it back for you….” Moreover, the system must allow users to enter the data themselves. |
| **Actual Result** | Since there was an error in the Market Data API, the application recognised it and stored this error information in its database. Then, the application warned the user that market data could not be provided at the moment but suggested attempting once again. All other functionalities within the application remained operational. |
| **Status** | Pass |

### 6.6.7. **Test Case #7:** Content Regeneration Workflow

*Table 6.7: Test Case 7*

| | |
|---|---|
| **Test Case ID** | **TC-07** |
| **Objective** | To ensure that users can reject an undesirable output and receive a revised version. |
| **Pre-Conditions** | An AI-generated caption draft using the most updated market data is currently available. |
| **Test Steps** | 1. Analyze the current caption output.<br>2. Press the "Regenerate Content" button.<br>3. Compare the new output to the previous version. |
| **Expected Result** | The AI processes the request using a different set of parameters, resulting in a completely new output that maintains the original brand tone and updated market data. |
| **Actual Result** | By pressing on the “Regenerate Content” button, the software created a completely different caption. It was different from the former one; however, it had still kept the brand voice of the company as well as the data from the real market. Versions of this caption were saved in the database for comparisons. |
| **Status** | Pass |

### 6.6.8. Test Case #8: Analytics Metric Synchronisation

*Table 6.8: Test Case 8*

| | |
|---|---|
| **Test Case ID** | **TC-08** |
| **Objective** | To check whether the Analytics functionality can collect and present data regarding engagements on various platforms. |
| **Pre-Conditions** | The test subject needs to have posted at least one post each on platform X[8] (Twitter) and LinkedIn using the software. |
| **Test Steps** | 1. Login to the Analytics Dashboard.<br>2. Filter "Last 7 Days" data.<br>3. Start the background task to fetch new data. |
| **Expected Result** | The system should successfully fetch data from API sources, update the database entries, and plot chart data for Impressions and Engagement Rate. |
| **Actual Result** | The analytics background process operator managed to extract the information relating to engagement and input the data into the database. The two dashboards named 'Impressions' and 'Engagement Rate' were updated within 45 seconds. |
| **Status** | Pass |

## 6.7. Results & Evaluation

The following Table 6.9 highlights the results of all test cases carried out during the testing phase, along with the corresponding performance metrics observed.

In all cases involving testing, success has been attained. It took between 45 and 55 seconds to produce each content; therefore, the condition set out by Section 3.3.1 in which such a process should not exceed 60 seconds was fulfilled. The criteria for designing have also been complied with in view of the fact that the success rate for brand kits has always been at least 95%.

The compliance checking component is among the most stable in the entire system, and it is able to detect any kind of non-compliance to the content produced. The postings were completed successfully, because all of them managed to get into the scheduling process.

The only significant vulnerability detected during the testing phase involved the system's reliance on third-party API availability. This was evident from TC-06, where disruptions to external services impact the data ingestion process. Although the system handled the disruption efficiently, this is a built-in flaw that is further elaborated upon in Section 7.2.

*Table 6.9: Test Summary*

| Test Case | Description | Status | Metric |
|---|---|---|---|
| **TC-01** | Market Data Collection | Pass | Data collected in <2 seconds |
| **TC-02** | Compliance Verification | Pass | Disclaimer mentioned appropriately |
| **TC-03** | Visualization | Pass | Images generated in <8 seconds |
| **TC-04** | Publication | Pass | Published within <60 seconds from scheduled time |
| **TC-05** | Account Lockout | Pass | Lockout performed on third try |
| **TC-06** | API Failover | Pass | Failover done successfully |
| **TC-07** | Content Update | Pass | Coherent output for brand |
| **TC-08** | Analytics Update | Pass | Dashboards refreshed in <45 seconds |

All the eight test cases ran successfully. The entire process in generating the content took about 10-30 seconds each time. This meets the criterion set forth in **Section 3.3.1** in taking less than 60 seconds to complete. Also, compliance in terms of brand kits was consistently at least 95 percent in all the generated designs.

One of the modules in the application that was highly reliable during the tests is the compliance checking module. In all the cases, any non-compliance with respect to the generated content was detected. Moreover, the scheduling pipeline was proven to work well as it published any posts not more than one polling cycle after the selected time.

From TC-06, one potential weakness in the application could be the reliance on third-party API providers for data collection. This is one of the structural weaknesses highlighted in **Section 7.2.**

## 6.8. Conclusion

During the testing process, it became evident that **Marketeer** is a fully reliable tool which operates flawlessly in all areas. Each element was tested separately first, followed by a complete test run from beginning to end. This approach allowed us to catch all possible problems and eliminate them in advance. In conclusion, it is possible to note that the platform is able to produce brand-compliant content powered by artificial intelligence in the designated time frame.

# Chapter 7

# Conclusion

**Marketeer** has managed to move from theory to practice, establishing itself as an operational platform based on artificial intelligence, which is tailored to the needs of fintech. Thanks to the integration of a smart and event-driven engine, this platform enables quick and accurate distribution of insights in the financial markets provided by financial experts.

## 7.1. Contributions

The development of **Marketeer** helped us achieve all the goals of the project set out at its initial stage. The contributions made by our research include the following:

- **Real-Time Post Generation:** The pipeline we developed can create full-fledged branded posts in seconds after retrieving the data from the source. In essence, this means that the time required for content generation has dropped from hours to seconds.
- **Integration of API & AI:** The integration of real-time APIs of financial platforms such as MarketVerse in the workflow of Agent Orchestration enabled us to ensure that the captions created by the AI would not contain any hallucinations.
- **Programmatic Design Ensuring Brand Compliance:** The team designed an efficient visualisation engine using Remotion[1] and various design APIs. This leads to creation of programmatic design which guarantees 95%+ brand-kit compliance via color themes, fonts, and logos by removing discrepancies from human design.
- **Centralised Distribution:** The team created a platform-independent scheduler module that helps publish approved designs directly on X[8] (Twitter), LinkedIn, and Instagram.

## 7.2. Reflections

The development of Marketeer was highly educational for understanding the delicate equilibrium between generative AI technology innovation and business governance.

- **Strengths & Effectiveness:** One of the key advantages of Marketeer is its innovative concept that integrates the strengths of the highly effective but

unpredictable nature of the language models (LLM) with those of the strictly regulated design and compliance requirements. In turn, the platform ensures consistent brand awareness together with unprecedented adaptability to current market conditions faster than any person can react to the changes. Ultimately, the platform converts the so-called "missed opportunities" into a proactive interaction model.

- **Weaknesses & Limitations:** Similar to any other platform, there are certain limitations associated with Marketeer. For instance, one of the limitations relates to the need for reliable Internet connection and functioning third-party services (e.g., finance and social media). In other words, should one of the services experience an issue, the system will become "blind." In addition, due to limited timeframes of 12 weeks for the development sprint, we were unable to develop a full-fledged version of the software.

### 7.3. Future work

There are several paths that could be pursued in order to improve Marketeer and make it ready for enterprise use:

- **Mobility:** Creating a dedicated mobile app will allow marketing managers to control the content moderation and posting process directly on their smartphones.
- **International Availability:** To ensure that Marketeer works for as many customers as possible, upcoming versions of the software will support multiple languages and translations, allowing fintech companies to address their global clientele.
- **Custom Template Library:** At the core of Marketeer’s speed are pre-defined templates. We plan on continuing to add more templates for Images, Videos and Newsletters to cater for large number of audiences.
- **Exclusive Data Sources:** Using paid or proprietary financial databases to analyze the market trends will yield better insights, resulting in exclusive content.

# References


[1] Remotion, "Remotion Documentation," 2024. [Online]. Available: https://www.remotion.dev/docs

[2] Meta Open Source, "React Documentation," 2024. [Online]. Available: https://react.dev/

[3] Supabase, "Supabase Documentation," 2024. [Online]. Available: https://supabase.com/docs

[4] Node.js Foundation, "Node.js v20.x API Documentation," 2024. [Online]. Available: https://nodejs.org/docs/latest-v20.x/api/

[5] Bannerbear, "Bannerbear API Documentation," 2024. [Online]. Available: https://www.bannerbear.com/api/

[6] ElevenLabs, "ElevenLabs API Reference," 2024. [Online]. Available: https://elevenlabs.io/docs/api-reference

[7] Alpha Vantage Inc., "Alpha Vantage Documentation," 2024. [Online]. Available: https://www.alphavantage.co/documentation/

[8] X Corp., "Twitter (X) API Documentation," 2024. [Online]. Available: https://developer.twitter.com/en/docs

[9] OpenAI, "OpenAI API Documentation," 2024. [Online]. Available: https://platform.openai.com/docs/

[10] OpenClaw, "OpenClaw GitHub Repository," 2024. [Online]. Available: https://github.com/OpenClaw/openclaw

[11] Vercel Inc., "Vercel Documentation," 2024. [Online]. Available: https://vercel.com/docs

[12] Unsplash, "Unsplash Developer Documentation," 2024. [Online]. Available: https://unsplash.com/developers

[13] Pexels, "Pexels API Documentation," 2024. [Online]. Available: https://www.pexels.com/api/

[14] Figma, "Figma REST API Documentation," 2024. [Online]. Available: https://developers.figma.com/docs/rest-api/

[15] Resend, "Resend Documentation Introduction," 2024. [Online]. Available: https://resend.com/docs/introduction

[16] Brandfetch, "Brandfetch Developer Documentation," 2024. [Online]. Available: https://brandfetch.com/developers

[17] Google, "Gemini API Documentation," 2024. [Online]. Available: https://ai.google.dev/gemini-api/docs

[18] LangChain Inc., "LangGraph Documentation," 2024. [Online]. Available: https://langchain-ai.github.io/langgraph/

# Appendices

# Appendix A: API Integration Payloads

The following appendix presents the exact JSON data structures used for the exchange of information within the components of the system as well as the APIs used outside the system.

## A.1. Financial Company API (Market Data Collection) - Response

This example showcases the live market data used by the system as inputs in the process of generating the agentic AI.

JSON


```
{
  "status": "success",
  "data": {
    "symbol": "TSLA",
    "asset_type": "Equity",
    "current_price": 205.45,
    "change_percentage": "+2.34%",
    "market_status": "open",
    "last_updated": "2026-04-12T14:30:00Z",
    "key_events": [
      "Q1 Earnings Report released",
      "Volume surge detected"
    ]
  }
}
```


## A.2. Programmatic Design API (Bannerbear / Remotion) - Request

This example shows the way in which the Content Generation Module transforms the generated text and brand kit constraints into design templates.

JSON


```
{
  "template_uid": "tmpl_a8x9v2BqK",
  "modifications": [
    {
      "name": "background_color",
      "color": "#0B0C10"
    },
    {
      "name": "brand_logo",
      "image_url": "https://storage.marketeer.com/user_102/logo_primary.png"
    },
    {
      "name": "headline_text",
```

```
    "text": "TSLA UP +2.34% FOLLOWING Q1 EARNINGS",
    "font_family": "Inter-Bold"
   },
   {
    "name": "financial_disclaimer",
    "text": "Capital at risk. Past performance is not indicative of future results."
   }
  ],
  "webhook_url": "https://api.marketeer.com/v1/webhooks/render-complete"
}
```


## A.3. Social Media Publishing (LinkedIn API) - Request

JSON


```
{
  "author": "urn:li:organization:123456789",
  "lifecycleState": "PUBLISHED",
  "specificContent": {
   "com.linkedin.ugc.ShareContent": {
    "shareCommentary": {
                "text": "Market Update: TSLA sees a +2.34% surge following Q1 Earnings. \n\n#Finance #StockMarket\n\n*Capital at risk.*"
    },
    "shareMediaCategory": "IMAGE",
    "media": [
     {
      "status": "READY",
      "media": "urn:li:digitalmediaAsset:C5422AQH_x",
      "title": {
       "text": "Automated Market Graphic"
      }
     }
    ]
   }
  },
  "visibility": {
   "com.linkedin.ugc.MemberNetworkVisibility": "PUBLIC"
  }
}
```

# Appendix B: Environment Configuration & Deployment

The following describes the necessary environment variables needed to be able to execute the monolith module-based backend and background services locally or in the cloud.

## .env Configuration File

### Code snippet

```
# Project: Marketeer
# Author: Faaiz Muzzammil

# Server Configuration
PORT=8000
NODE_ENV=development
CLIENT_URL=http://localhost:3000

# Database Configuration (Supabase/PostgreSQL)
DATABASE_URL=postgresql://postgres:[PASSWORD]@db.[PROJECT-
REF].supabase.co:5432/postgres
SUPABASE_SERVICE_ROLE_KEY=ey...

# AI & Orchestration
OPENAI_API_KEY=sk-...
LANGCHAIN_TRACING_V2=true
LANGCHAIN_API_KEY=ls__...

# External Integrations
FINANCIAL_COMPANY_API_KEY=fin_...
REMOTION_GITHUB_TOKEN=ghp_...
BANNERBEAR_API_KEY=bb_...

# Social Media OAuth Credentials
X_API_KEY=...
X_API_SECRET=...
LINKEDIN_CLIENT_ID=...
LINKEDIN_CLIENT_SECRET=...
```

# Appendix C: Database Data Dictionary

This appendix offers the detailed schema description of some of the relational database tables used in the storage of application state within Supabase[3] (PostgreSQL).

### Table D.1: users

*Table C.1: Users*

| Column Name | Data Type | Constraints | Description |
|---|---|---|---|
| id | UUID | PRIMARY KEY | Unique identifier for the user. |
| email | VARCHAR(255) | UNIQUE, NOT NULL | User's login email. |
| password_hash | VARCHAR(255) | NOT NULL | Encrypted password string. |
| role | VARCHAR(50) | NOT NULL | Access level (e.g., Admin, Marketing Manager). |

### Table D.2: brand_kits

*Table C.2: Brand Kit*

| Column Name | Data Type | Constraints | Description |
|---|---|---|---|
| id | UUID | PRIMARY KEY | Unique identifier for the brand kit. |
| user_id | UUID | FOREIGN KEY | Links to users.id. |
| logo_url | TEXT | NOT NULL | CDN link to the uploaded brand logo. |
| primary_color | VARCHAR(7) | NOT NULL | Hex code for the primary brand color. |
| font_family | VARCHAR(100) | NOT NULL | Selected typography style. |
| disclaimer | TEXT | | Mandatory compliance text. |

### Table D.3: posts

*Table C.3: Posts*

| Column Name | Data Type | Constraints | Description |
|---|---|---|---|
| id | UUID | PRIMARY KEY | Unique identifier for the post content. |
| user_id | UUID | FOREIGN KEY | Creator of the post. |
| market_data_id | UUID | FOREIGN KEY | Links to the cached market metrics used. |
| caption | TEXT | NOT NULL | The AI-generated or edited text. |
| media_url | TEXT | | CDN link to the programmatic image/video. |
| status | VARCHAR(50) | NOT NULL | State (e.g., Draft, Approved, Rejected, Scheduled). |

### Table D.4: schedules

*Table C.4: Schedules*

| Column Name | Data Type | Constraints | Description |
|---|---|---|---|
| id | UUID | PRIMARY KEY | Unique identifier for the schedule instance. |
| post_id | UUID | FOREIGN KEY | Links to posts.id. |
| platform | VARCHAR(50) | NOT NULL | Target platform (X, LinkedIn, Instagram). |
| publish_time | TIMESTAMP | NOT NULL | Exact date and time to execute publishing. |
| is_published | BOOLEAN | DEFAULT FALSE | Status flag for the background worker. |

# Appendix D: User Acceptance Testing (UAT) Questions

Questionnaire used during system testing to collect formal feedback from Marketing Managers testing the application.

## Section 1: User Interface and Usability

1. On a scale of 1 to 5, how did you perceive the usability of the split-view design for creating and previewing posts? **4 (Four)**
2. Was it possible for you to move from Generate tab to Scheduler tab without requiring any help? (**Yes**/No)
3. Was the Dashboard capable of guiding you towards the content automation pipeline after logging in? (**Yes**/No)

## Section 2: Generation of Content and Compliance

4. Was the information generated by the AI captions accurate in terms of reflecting the financial information you provided? (**Yes**/No)
5. When you tried removing a compulsory disclaimer on the Compliance Module, was it unable to approve the post? (**Yes**/No)
6. Was the Programmatic Design Template correct in showing your brand colours, fonts, and logo? (**Yes**/No)

## Section 3: Performance & Efficiency

7. Was the generation process (including data fetching, drafting, image rendering) done within an acceptable period (i.e. less than 60 seconds)? (**Yes**/No)
8. Relative to traditional manual design processes, how much time is saved per social media post using Marketeer? **1 hour or more.**

# Kashif Sultan

## Marketeer - FYP Report

Student's Task

### Document Details

Submission ID
trn:oid:::3618:135806092

Submission Date
Apr 20, 2026, 9:08 AM GMT+5

Download Date
Apr 20, 2026, 9:16 AM GMT+5

File Name
Marketeer - FYP Report.pdf

File Size
3.8 MB

91 Pages

14,705 Words

78,959 Characters

# 13% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

## Match Groups

- **132** Not Cited or Quoted 13%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **3** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

## Top Sources

7% Internet sources
3% Publications
12% Submitted works (Student Papers)

## Integrity Flags

**0 Integrity Flags for Review**

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

# *% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

**Caution: Review required.**

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.



## Frequently Asked Questions

**How should I interpret Turnitin's AI writing percentage and false positives?**
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

**What does 'qualifying text' mean?**
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.