A picture containing graphical user interface

Description automatically generated

CS 25-349:

AI-Powered Email Response System Using Fine-Tuned LLMs for Customer Service in React

**Final Design Report**

Prepared for

Keroles Hakem, CoStar Group

By

Cameron Clyde, Emma Smith,

Angela Harris, Sohil Marreddi

Under the supervision of

Preetam Ghosh

5/2/2025

**Executive Summary**

As CoStar Group continues to grow, the demands on its customer service teams increase significantly. CoStar Group’s internal customer relationship management system is known as Case Management. Case Management employees are responsible for handling complex cases and responding to customer emails. Balancing these responsibilities often results in delayed responses and inconsistent communication quality, as employees struggle to maintain both professionalism and personalization in their emails. Furthermore, replying to emails takes away time employees have to deal with their cases.

These challenges highlight the need for a solution that streamlines the email process, allowing employees to respond quickly and efficiently without compromising the quality of service. This project will leverage Large Language Models (LLMs) to assist in generating quality email responses for Case Management Employees. The overall aim of this project is to create a web application with a fine-tuned LLM on specific customer service data to generate relevant and context-aware email responses. The project goals include implementing a feedback loop to continuously improve the model’s performance, allowing employees to review and edit generated drafts before sending them to customers and reducing the time required to respond to customer emails.

The design specifications and constraints outline a system that facilitates the generation of professional email responses for open customer service cases, integrating the React library for frontend development and utilizing FastAPI and PostgreSQL for backend functionality. The key requirements that will be implemented in this design include a feedback loop for AI suggestions, JWT-based authentication, AI response templates that will be customizable for the user, storage management for data related to case assignment logic, AI-generated responses, and representative feedback, and continuous integration and deployment pipelines for automated testing and updates. A successful design will effectively reduce the workload for customer service representatives, handle an increasing number of customer service cases and interactions, minimize customer service response times, and enhance customer satisfaction. In the creation of this design, we will adhere to a budget of $1000, comply with data privacy regulations, and take necessary security measures against vulnerabilities that can present themselves.

In order to meet the necessary design requirements, it’s important that we are always applying a set of codes and standards that will ensure quality, reliability, and safety. In this document, a number of codes and standards relating to security, privacy, software development, quality, ethics, design, etc. are given that can be directly applied to our design. Two codes relating to security and privacy that we will be adhering to in our design are the GDPR (General Data Protection Regulation) and CCPA (California Consumer Privacy Act). We will also be applying multiple standards, including ISO/IEC 27701, relating to information security management, ISO/IEC 25010, which outlines the software quality requirements and evaluation, OpenAPI, the standard for API documentation, ISO/IEC TR 24027, the guidelines for evaluating AI-based systems, Material Design Guidelines, which describes the best practices for UI design using Material UI components, etc. With the usage of these codes and standards, we can be sure to create the most efficient and secure design.

The scope of the project is clearly defined to ensure all objectives are met on time and within budget. The web application, CASEflow, will be developed using an Agile methodology with sprints spanning 2-3 weeks each. Deliverables include a fully functional web application that allows employees to log in, access customer emails, and generate AI-assisted responses, fine-tuned on CoStar's customer service data. Key features include an interface for employees to review and edit AI-generated responses, a feedback loop to improve the AI's performance over time, and detailed metrics for response times and model accuracy improvements. Academic deliverables include the project proposal, reports, and presentations required for the Capstone EXPO.

The project will utilize resources such as GitHub for version control, React for frontend development, FastAPI for backend operations, and PostgreSQL for database management. The large language model (LLM) will be hosted on a cloud provider. The team will focus on continuous integration to ensure a smooth development process, with testing and feedback incorporated at each sprint to avoid scope creep and ensure all deliverables are met. The application aims to streamline email response times, reduce employee workload, and improve the quality and consistency of customer interactions.

**Table of Contents**

[Section A. Problem Statement 5](#_heading=h.1fob9te)

[Section B. Engineering Design Requirements](#_heading=h.2et92p0) 6

[B.1 Project Goals (i.e. Client Needs)](#_heading=h.3dy6vkm) 6

[B.2 Design Objectives](#_heading=h.1t3h5sf) 6

[B.3 Design Specifications and Constraints](#_heading=h.4d34og8) 7

[B.4 Codes and Standards](#_heading=h.2s8eyo1) 8

[Section C. Scope of Work](#_heading=h.17dp8vu) 10

[C.1 Deliverables](#_heading=h.3rdcrjn) 10

[C.2 Milestones](#_heading=h.26in1rg) 11

[C.3 Resources](#_heading=h.lnxbz9) 12

[Section D. Concept Generation 1](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.35nkun2)3

[Section E. Concept Evaluation and Selection 1](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.1ksv4uv)6

[Section F. Design Methodology](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.44sinio) 18

[F.1 Computational Methods (e.g. FEA or CFD Modeling, example sub-section)](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.2jxsxqh) 18

[F.2 Experimental Methods (example subsection)](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.z337ya) 19

[F.3 Validation Procedure](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.3j2qqm3) 19

[Section G. Results and Design Details](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.1y810tw) 21

[G.1 Modeling Results](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.4i7ojhp) 21

[G.2 Experimental and Testing Results](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.2xcytpi) 23

[G.3. Final Design Details/Specifications](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.3whwml4) 24

[Section H. Societal Impacts of Design](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.2bn6wsx) 29

[H.1 Public Health, Safety, and Welfare](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.qsh70q) 29

[H.2 Societal Impacts](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.3as4poj) 29

[H.4. Economic Impacts](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.49x2ik5) 30

[H.7. Ethical Considerations](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.3o7alnk) 30

[Section I. Cost Analysis](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.23ckvvd) 31

[Section J. Conclusions and Recommendations sohil](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.ihv636) 32

[Appendix 1: Project Timeline](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.32hioqz) 34

[Appendix 2: Team Contract (i.e. Team Organization)](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.1hmsyys) 35

[References](https://docs.google.com/document/d/1JXnZbI9s5-Lo3YwapsDwXBdkRTNz8cva/edit#heading=h.2grqrue)39

### Section A. Problem Statement

CoStar Group is the global leader in the digitalization of commercial real estate. Since their start in 1986, they have expanded into the world of residential real estate and become a leader in this industry. Their mission is to digitize the world’s real estate and make it easier for people and companies to discover properties. They provide real estate information, analytics, and online marketplaces (CoStar Group, 2024). In their mission to digitize real estate around the world, CoStar Group continues to grow every year. CoStar Group acquires new brands and companies to help achieve this mission. CoStar Group has over 25 brands, including Homes.com and Apartments.com.   
 Merging these businesses into CoStar Group can be difficult. Every company that CoStar Group acquires does their business differently. In particular, most companies have their own subscriptions with different Customer Relationship Management (CRM) systems. CRM systems are a strategic tool that integrate technology, customer knowledge and relationships to enhance business efficiency, foster customer loyalty, and drive economic growth (Gil-Gomez et al., 2020). It is evident that CRMs are essential, but CoStar Group does not want to deal with multiple different CRMs and pay fees for each CRM. To solve this, they decided to run it all internally and develop their own CRM, known as Case Management.  
 In CoStar’s existing customer service response process, representatives begin by reading the incoming email along with the associated email history to fully understand the customer's issue. After reviewing the context, they select an appropriate response template from a set of predefined options that best aligns with the customer's inquiry or concern. The representative then customizes the chosen template by filling in relevant details specific to the case, such as the customer’s name, case number, and any additional information that addresses the issue.  
 As the amount of cases increases, the current process begins to show limitations. This process becomes time consuming, labor intensive, and prone to human error as representatives are required to manually fill each template (Mesquita et al., 2022). The reliance on templates can result in repetitive and impersonal responses, which may not fully address each customer's issue. This adds additional time and effort as representatives must write their own reply. While technologies have been proposed to automatically query and select customer data to fill response templates, they are not completely accurate and still fail to address customer issues that are not covered by templates (Malik et al,. 2007). As case volumes rise, it becomes increasingly difficult for employees to manage the workload effectively, leading to delayed responses, inconsistencies in communication, and reduced overall quality of customer service (Sheth et al., 2024). This issue creates the need for a more scalable solution that can maintain personalization without sacrificing speed or accuracy.  
 Our project aims to improve and streamline this process by developing an email response system that utilizes a fine-tuned large language model (LLM) to generate personalized responses. This system will analyze the incoming email and its history to understand the customer's issue, then automatically create a response without relying on premade templates. The LLM will use the context of the email history to create responses more tailored to the specific case, while automatically filling in relevant customer data, which increases both the speed and quality of customer service interactions.

### Section B. Engineering Design Requirements

#### B.1 Project Goals (i.e. Client Needs)

This project aims to assist customer service employees at CoStar Group with the high-volume of emails received. The demands for customer service employees grow as CoStar Group acquires new businesses. It can be difficult for employees to manage their existing work on top of the many emails they receive. Responding to emails in a personalized and professional way is time consuming. When trying to balance work on existing complex cases, writing these emails can lead to inconsistencies. With this in mind, the project goals are as follows:

* To design an app that reduces the time required to respond to customer emails
* To enable employees to focus on high priority work by automating the generation of strong email replies
* To improve the personalization and professionalism of email responses from customer service employees
* To allow employees to review and edit generated drafts, ensuring accuracy and appropriateness
* To enhance the AI model through employee feedback to continuously improve the model’s performance

#### B.2 Design Objectives

The following are key objectives of the design:

* The design will reduce email response times for employees by generating draft replies.
* The design will create a web app, with an authentication system to ensure only CoStar Group Case representatives can access this information. This structure will be completed halfway through the fall semester, with testing to ensure functionality as expected.
* The design will incorporate retrieval-augmented generation (RAG) using a company knowledge base to generate relevant, context-aware email responses.
* The design will implement a feedback loop where employees can rate the email responses produced by the model to improve the model. This will be measured by the model’s ability to learn from feedback.
* The design will include a user interface allowing employees to review and edit responses for accuracy and appropriateness. This will be measurable by employee usage and edit rates.

#### B.3 Design Specifications and Constraints

**Design Specifications**

*Functional Requirements*

* The system must be powered by fine-tuned LLMs
* Design must facilitate the generation of professional email responses for open customer service cases
* The system must integrate the React library for frontend development
* Design must integrate with existing customer service platforms
* Design must integrate FastAPI and PostgreSQL for backend functionality
* Design must implement JWT-based authentication for user login and signup on the backend
* Design must implement login and signup pages on the frontend
* Design must be able to store data related to case assignment logic, AI-generated responses, and representative feedback on those responses
* Design must allow for customization of the AI response templates
* Design must follow *IEEE P7003 Algorithmic Bias Considerations* to ensure that AI-generated responses are free from bias and are transparent to both employees and customers
* The design must include unit tests, integration tests, and automated testing pipelines to follow ISO/IEC/IEEE 29119 software testing standards
* Design must implement continuous integration and deployment pipelines for automated testing and seamless updates (CI/CD best practices)

*Non-Functional Requirements*

* Design must effectively reduce the workload for customer service representatives
* The system should be designed to handle an increasing number of customer service cases and employee interactions
* The AI model must generate responses quickly to ensure minimal delays in customer service workflow
* The design must follow *ISO/IEC 25010* to ensure software quality, including functionality, usability, and maintainability
* The design must adhere to *WCAG 2.1 guidelines* to ensure that the interface is accessible to users with disabilities
* AI model must undergo regular evaluation to ensure that its suggestions are accurate, relevant, and helpful
* Design should aim to minimize response customer service response times
* Design must enhance customer satisfaction

**Design Constraints**

*Cost*

* Design must adhere to a budget of $1000

*Security and Data Privacy*

* All user data must comply with data privacy regulations
* Design must implement data retention policies that comply with *GDPR/CCPA* standards to ensure user data is only stored for the required period
* Design must comply with *GDPR* guidelines, ensuring that user data is protected and the necessary opt-ins and transparency measures are in place
* Design must ensure protections against the top 10 web application vulnerabilities, including secure authentication, access control, and input validation (OWASP Top Ten)
* The system must follow standards for managing personally identifiable information (ISO/IEC 27701 Compliance)
* Design must securely store JWTs within the system

*Quality Assurance and Testing*

* Design must ensure that tests cover security vulnerabilities and performance bottlenecks, which limit the system’s performance (ISTQB guidelines)
* Design must include robust error handling

#### B.4 Codes and Standards

**Codes:**

*Security and Privacy*

* [GDPR](https://gdpr-info.eu/) (General Data Protection Regulation): Data protection and privacy law for EU residents
* [CCPA](https://oag.ca.gov/privacy/ccpa) (California Consumer Privacy Act): Privacy law for California residents, focusing on data transparency and user control over personal data

**Standards:**

*Security*

* [OWASP Top Ten](https://owasp.org/www-project-top-ten/): Guidelines for mitigating common web vulnerabilities
* [NIST Cybersecurity Framework](https://www.nist.gov/itl/smallbusinesscyber/nist-cybersecurity-framework-0): Best practices for managing cybersecurity risks
* [ISO/IEC 27701](https://www.iso.org/home.html): Information security management systems standard
* [JWT Best Practices](https://curity.io/resources/learn/jwt-best-practices/): Guidelines for secure handling of JSON Web Tokens
* [CIS Benchmarks](https://www.cisecurity.org/cis-benchmarks/cis-benchmarks-faq) (Center for Internet Security): Best practices for cloud infrastructure security

*Privacy*

* [ISO/IEC 27701](https://www.iso.org/home.html): Privacy extension to ISO 27001, focusing on managing personal data (PII)

*Software Development Standards*

* [ISO/IEC 25010](https://www.iso.org/home.html) (Software Quality Requirements and Evaluation): Software quality metrics, such as usability and reliability
* [IEEE 12207](https://www.ieee.org/): Framework for software development lifecycle (SDLC)
* [SOLID Principles](https://www.geeksforgeeks.org/solid-principle-in-programming-understand-with-real-life-examples/): Object-oriented design principles for maintainability
* [Agile Development Framework](https://d1wqtxts1xzle7.cloudfront.net/24840355/ijse-65-libre.pdf?1390868387=&response-content-disposition=inline%3B+filename%3DAn_Agile_Software_Development_Framework.pdf&Expires=1728598822&Signature=fqRxL23F-YhUeNF9poRw2FdfBRREas6wW6Xlbrx9WTkuQdVW4wOsWMuJRTESdPKk~2QBK7mZlFNKTFVrP-Ooknx0spC51QKuGKKpO2Uk8GkCxBO8FlcExbUMFHgaga36F9J7XI0lI4DSCSiS0IvM~m~FMQd~43p-cKQl2ityKW19v-mxERGVvHXbgcPraeJ8ze5kQyS0IhXHKwJsnpt60uSyrqwQIpiSFD8P3eXMQQJSpMCXipDgjqZq2X6~K1Kd7b-feh9viAeamea920JMnT882TcLcDPw29ft82j9BfD7xuF4dlOYkJmhBTuae7jssIM90jXas3iR7jh2tsDghQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA): Practices for iterative, adaptive development

*API and Web Standards*

* [RESTful API Best Practices](https://restfulapi.net/): Guidelines for designing RESTful APIs (proper use of HTTP methods, statelessness)
* [OpenAPI](https://www.openapis.org/): Standard for API documentation
* [W3C Web Standards](https://www.w3.org/): Accessibility, responsive design, and cross-browser compatibility
* [WCAG 2.1](https://www.w3.org/TR/2023/REC-WCAG21-20230921/) (Web Content Accessibility Guidelines): Guidelines for designing accessible web content

*AI Ethics and Guidelines*

* [ISO/IEC TR 24027](https://www.iso.org/home.html): Guidelines for evaluating AI-based systems
* [IEEE P7003 Algorithmic Bias Considerations](https://www.ieee.org/): Recommendations for avoiding bias in AI systems

*Quality Assurance Standards*

* [ISTQB](https://www.istqb.org/) (International Software Testing Qualifications Board): Best practices for software testing
* [ISO/IEC/IEEE 29119](https://www.iso.org/home.html): Software testing standards for comprehensive testing
* [CI/CD Best Practices](https://www.jetbrains.com/teamcity/ci-cd-guide/ci-cd-best-practices/): Guidelines for automate testing, deployment, and version control

*Frontend UI/UX Design Standards*

* [Material Design Guidelines](https://m3.material.io/): Best practices for UI design using Material UI components
* [User-Centered Design](https://usabilitygeek.com/user-centered-design-introduction/) (UCD): Focus on user needs during the design process ([ISO 9241-210](https://cdn.standards.iteh.ai/samples/77520/8cac787a9e1549e1a7ffa0171dfa33e0/ISO-9241-210-2019.pdf))

*Database Standards*

* [GDPR](https://gdpr-info.eu/art-5-gdpr/)/CCPA Data Retention Policies: Clear policies for data storage and retention, especially regarding personal and sensitive data

### 

### Section C. Scope of Work

#### C.1 Deliverables

1. A Web Application named CASEflow, a secure web application which allows users to login to access protected data (emails), and get an AI generated email response which improves on the Large Language Model we finetune.
   1. Given real company data (data cleaned for privacy), we are to provide Fine Tuning to the existing Large Language Model to improve speed and performance of the AI response.
   2. Build a secure web app to facilitate interaction between service representatives and customers with JWT authentication​.
   3. Functionality for employees to review, edit, and send the AI-generated email responses using the application.
   4. Create a case dashboard to enable representatives to view, add, and manage cases efficiently.​
2. Provide technical documentation, a user manual, and build steps for running the application CASEflow.
3. Academic Deliverables
   1. Team Contract
   2. Project Proposal
   3. Preliminary Design Report
   4. Fall Poster and presentation
   5. Final design report
   6. Capstone EXPO poster and presentation

Some important issues to discuss with the design team, sponsor, and faculty advisor include the following:

The only third-party vendor we were ultimately unable to gain access to was Amazon Web Services (AWS). Our original plan of action relied on using an AWS account to deploy and fine-tune a large language model, which is a core component of our CASEflow application. Since access to AWS was not granted, we adapted by using Together AI’s hosted inference APIs and Qwen, an open-source language model, to support the AI-driven email generation feature.

Method of deployment, issues with our deployment, whether that our application is failing to build, or not enough testing being done because everything is being run on our local environment until deployment.

All deliverables can be worked on remotely; however, due to a large volume of remote work the group has multiple means of communication including Discord, Slack, email, and Zoom. Meeting Minutes and other information is all organized in shared google drives for effective remote work.

#### C.2 Milestones

Our team will be utilizing Agile Development with a focus on Continuous Integration. The project will be divided into distinct phases, or 'sprints,' each spanning 2-3 weeks, depending on the complexity of the tasks. The length and content of each sprint will be collaboratively decided by the student team, project mentor, and faculty advisor. The deliverables from each sprint will serve as milestones to track progress and ensure that the final project objectives are completed on time.

**Sprints:**

Sprint 1: Set up basic project structure Date: 9/9/24 - 9/23/24

* Set up folder structure for Frontend(react), Backend(FastAPi), and connect database(PostgreSQL).

Sprint 2: Authentication Date: 9/23/24 - 10/7/24

* Complete Sign in and Log in page UI(react)
* Create JWT tokens to enable secure logins, Connect frontend and backend for testing.

Sprint 3: Case Management & Representative Schema Date: 10/7/24 - 10/21/24

* Create routes for Different pages
* Create Casetable(frontend)
* Create database models for Casetable(backend)
* Create API Endpoints for retrieval and entry of the CaseTable information.
* Create Form for users to submit new cases, and a dashboard to view assigned cases(frontend)

**Milestone: Completion of basic project design Date: 10/21/24**

Sprint 4 (Weeks 8-10): Connecting LLM Date: 10/21/24 - 11/4/24

* Connect LLM to backend VIA AWS Sagemaker (pending aws account approval)
* Test all components of application before fine tuning next sprint

Sprint 5 - 8: LLM Fine-Tuning Date: 11/4/24 - 12/16/24

**Milestone: Completion of LLM Fine Tuning Date: 12/16/24**

Sprint 9 - 10: LLM Feedback Date: 1/13/25 - 2/10/25

* LLM Feedback and metrics report testing.

Sprint 11 (Weeks 19-20): Testing Date: 2/10/25 - 2/24/25

* Testing and Documentation write up

**Milestone: Completion of testing Date: 2/24/25**

Sprint 12: Bug Fixes Date: 2/24/25 - 3/10/25

Sprint 13: Deployment Date: 3/10/25 - 3/24/25

**Milestone: Completion of project Date: 3/24/25**

**Academic Milestones:**

**CMSC 451 Deliverables**

Team Contract Date: 9/4/24

Project Proposal Date: 10/11/24

Preliminary Design Report Date: 11/15/24

Fall Design Poster and Presentation Date: 12/12/24

**CMSC 452 Deliverables**

Final Design Report Date: 2/23/25

Capstone EXPO Abstract Date: 3/5/25

Capstone EXPO Poster Date: 4/18/25

Capstone EXPO Presentation Date: 4/25/25

#### C.3 Resources

**Paid Resources**

Access to Amazon web service account, needed to utilize AWS SageMaker for LLM fine-tuning and machine learning aspect. These accounts will be provided by Project Sponsor (CoStar Group).

**Free Resources**

We will be using GitHub for our Version Control System, which is accessible to all VCU students. Programming language, frameworks and technologies are all available to open source download. We will be using React, FastAPI, PostgreSQL, SwaggerUI and Visual Studio Code. For Communication we are using Slack, Discord, Zoom, and Email.

### Section D. Concept Generation

When finalizing our design concept, we focused on the needs of the customer, which are the Case Management Employees at CoStar Group. When drafting design concepts, we thought it was beneficial to start simple then build complexity. We started with a basic AI-powered email generator and progressively introduced additional functionalities, such as case management and advanced AI capabilities.

**Design Concept 1: Simple AI-Powered Email Generator Application**

The first design concept starts simple, but still aims to meet all of the necessary customer requirements. This design has a simple email generation system. It incorporates retrieval-augmented generation (RAG) to generate email responses based on a knowledge base containing company procedures. Once a draft is produced, representatives have the option to edit the draft with a simple text editor.

The primary technologies for this design include React for the frontend, providing a simple interface, and FastAPI for the backend to facilitate API calls to the RAG agent and our knowledge base. Using cloud-based AI generation, we will use Qwen 2.5-7b to handle tool calls and text generation, because it is easy to set up and cost-effective. To protect sensitive data, there would be a simple user authentication system. User authentication would rely on a basic username-password structure.

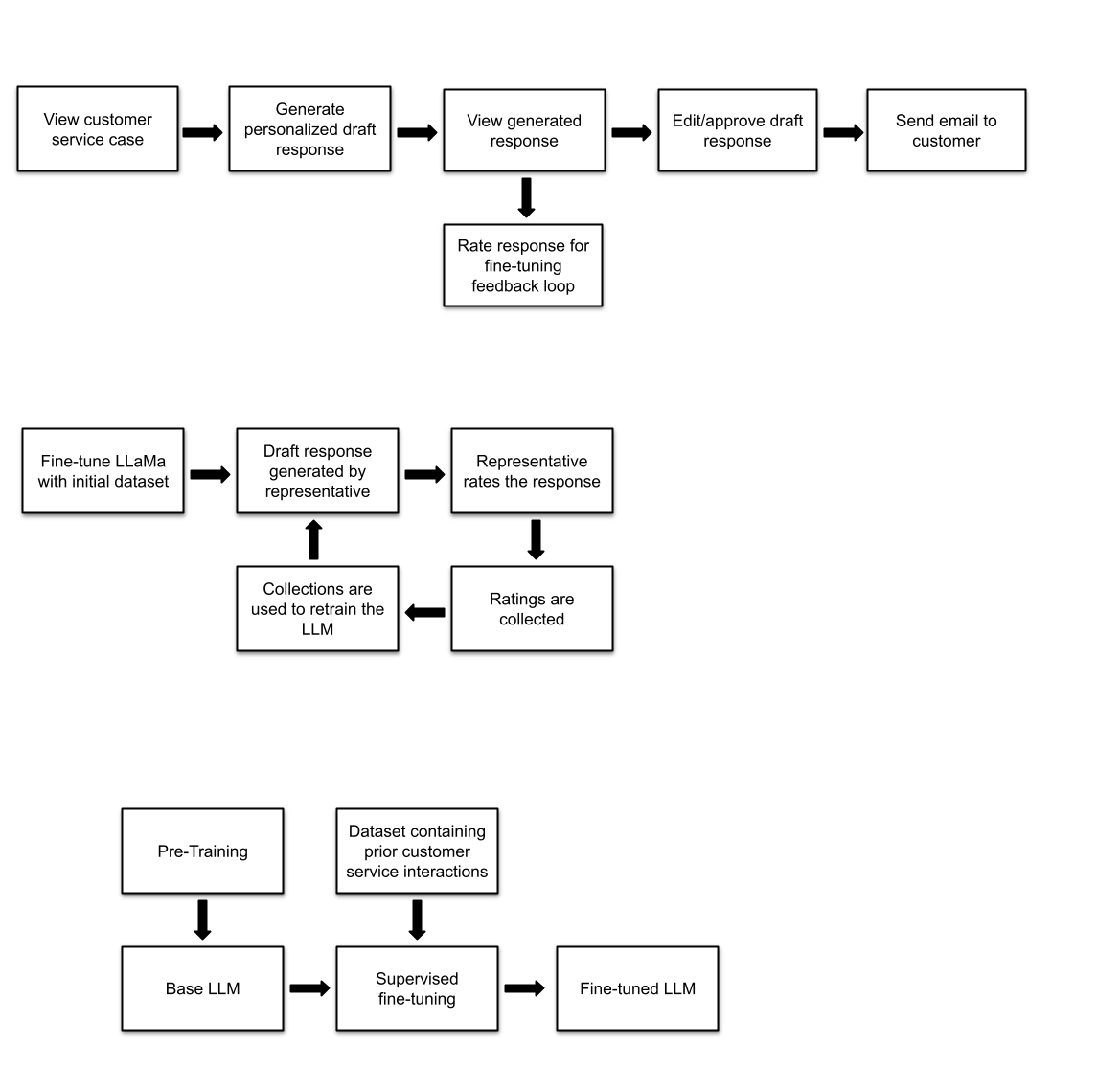
The advantages of this design are its simplicity, ease of development, and cost-effectiveness. The simple features mean faster deployment and lower development costs. It is also user-friendly, as the straightforward interface allows representatives to quickly learn and use the tool. However, it has disadvantages due to its simplicity. The AI model is not robust enough to address multiple types of emails. It uses a predefined knowledge base, which restricts its flexibility for generating responses for more complex emails. In addition, the basic feedback loop and use of a simple LLM is not robust enough for the AI to continuously learn and improve.

This concept addresses the design problem by providing a simple yet effective tool AI-assisted email response generation. Representatives benefit from a system that generates appropriate responses while remaining easy to use. However, more advanced email generation and case management is necessary to make the application useful.

**Design Concept 2: Simple AI-Powered Email Generator Application with Case Management System**

This design builds on Design Concept 1 by introducing a case dashboard to the application. This allows users to view and manage existing cases more effectively. The dashboard provides an overview of case details such as their status and assignment. Like the first concept, this system includes basic login functionality for authentication. This concept could use React for the frontend along with a component-based design to implement the case dashboard and email generator pages. For the backend, FastAPI would also be utilized to manage case data and communicate with the AI email generator.

The image below shows the workflow of this design concept. Users can view their assigned cases through the dashboard, generate personalized email drafts based on the model’s suggestions, and edit the drafts as necessary before sending them to customers.



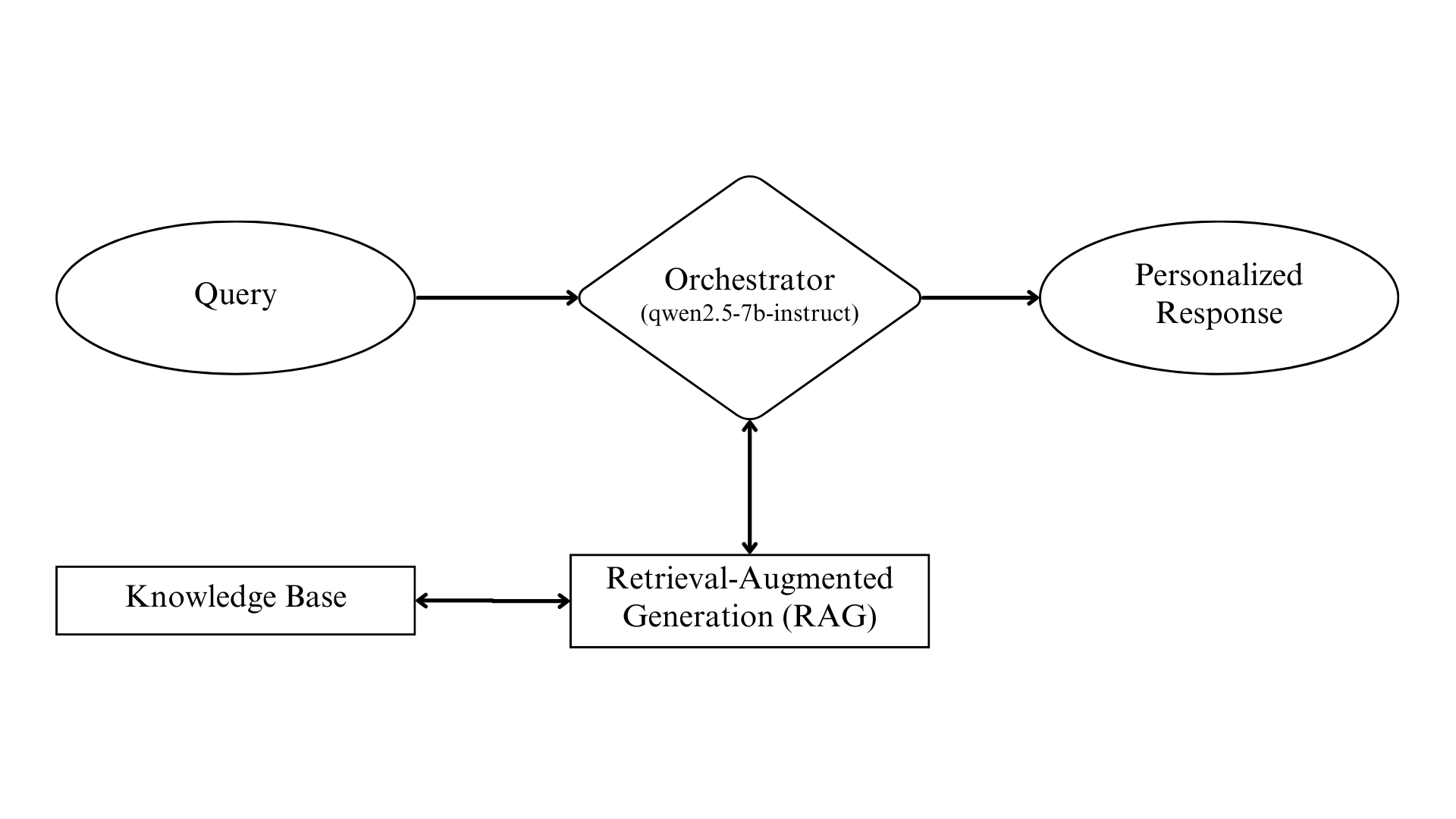
This design offers several key advantages. First, the introduction of a case dashboard enhances case management, providing users with an overview of cases. This allows representatives to manage and prioritize cases more effectively. Additionally, the integration of a RAG agent for email generation ensures high-quality, contextually relevant drafts that can be customized with ease. The interface remains simple and user-friendly. However, this design also comes with some drawbacks. One key limitation is the lack of a rich text editor, which still restricts the ability to fully personalize email responses. Representatives may struggle with editing emails for specific customer needs, limiting the flexibility of the system. Additionally, the lack of a robust feedback loop means that the AI-generated email responses cannot be continuously improved based on user feedback. The basic authentication system may become a security concern as the system grows and handles more sensitive data.

By introducing a case dashboard and improving the email generation technology, this design addresses some of the limitations of the first concept. However, it still lacks a rich text editor and a robust feedback loop, which could limit the personalization and adaptability of email responses. This design is more sophisticated but remains relatively simple, making it a practical next step for enhancing user experience without adding significant development complexity.

**Design Concept 3 (Final): Advanced Case Management and Email Generation System**

The final design concept builds on the previous designs, adding advanced features to meet more complex user needs. This design will use Qwen 2.5-7b, running on a cloud provider, to generate high-quality email responses, improving the AI’s capabilities. This implementation will have a more robust case management system, allowing users to add, delete, and edit existing cases. On the case dashboard, representatives can sort the case by status, who it is assigned to, or creation date. Furthermore, each case will have details and a timeline page that representatives can use to have a holistic view of the case. The backend, built using FastAPI, will store this critical case information. The backend will also store draft email responses, email response rates, and real customer emails. This design also incorporates a rich text editor, powered by Lexical, to allow representatives to format and customize email responses. Secure login functionality is implemented using JWT tokens to ensure user security.

The workflow for Design Concept 3 combines Retrieval-Augmented Generation (RAG) with the Queen 2.5-7B-Instruct model. The process begins when a user submits a query, which is handled by an orchestrator that manages the RAG pipeline. The orchestrator retrieves relevant information from a structured knowledge base and a vector database using LangGraph, a graph-based retrieval system designed to optimize search efficiency. This knowledge base contains customer service documents, FAQs, and historical case data. The retrieved context is then passed, along with the original query, to the Qwen 2.5 model. The model generates a personalized and contextually accurate response based on both the query and the retrieved documents:



There are several advantages to this design concept, which is ultimately why we decided on it. The integration of Retrieval-Augmented Generation (RAG) with Qwen 2.5-7b enables advanced email generation, ensuring that responses are contextually accurate and professional. This ability enhances customer interactions. The rich text editor offers representatives flexibility when editing email drafts. The advanced case management dashboard allows users to track, edit, and manage cases efficiently, with additional features like a timeline view providing a detailed history of case progression. This helps representatives with their work by storing everything they need in one application. Secure user authentication, implemented using JWT tokens, ensures that user data and system access are protected. Furthermore, the modular architecture of the system, supported by FastAPI and SageMaker, allows for scalability as user demands grow. The system’s capability to store user feedback allows for continuous improvement of AI-generated responses over time.

However, this robust system architecture introduces certain challenges. Advanced features such as AI-driven email generation using RAG and Qwen 2.5-7b, a rich text editor, and secure authentication add development complexity. This can extend deployment timelines and increase the risk of integration issues. Additionally, running AI components via external APIs and maintaining a responsive backend infrastructure require careful optimization to ensure system performance. Since the effectiveness of the application relies heavily on the quality of AI responses, any limitations in training data or lack of user feedback may negatively impact accuracy and user satisfaction.

This concept offers a robust solution by addressing the need for detailed case management, rich email editing, and secure user authentication. However, the complexity of integrating these advanced features increases the development effort and potential risks, such as performance concerns or a steep learning curve for users. Despite these challenges, this concept is the most scalable and adaptable, making it well-suited for long-term use as the system evolves.

### Section E. Concept Evaluation and Selection

### In our initial planning phases, we researched several methods to approach the project. Through our research and conversations with our sponsor, we identified key objectives and constraints that our design concepts must satisfy to ensure project success. In this section, we will evaluate the design concepts we researched and develop a systematic method to compare them against the established criteria. Based on this evaluation, we will select the most suitable options for the project.

**Criteria**  
 The primary goal of our project is to enhance the efficiency of customer service representatives when responding to clients. This goal can be achieved by designing essential criteria. To satisfy these requirements, we need a robust and efficient application, while also addressing practical constraints such as budget, development time, and tool complexity.

The criteria we used to evaluate our design concepts are:

* **Ease of Use:** How much of an adjustment would the representative go through to use our application.
* **Performance**: How well the application meets the functional needs.
* **Reliability**: The robustness of the application and its capacity to handle expected workloads.
* **Cost-effectiveness**: Whether the application offers a good balance between cost and functionality.
* **Security**: How secure is the application’s handling of sensitive data.
* **Response Quality**: Does the model’s generated responses meet the client’s requirements.

**Design Concepts** In section D of the report we discussed our three iterations of design concepts. In concept one, we came up with a simple application that is only able to generate responses based on a prompt that the customer service representative supplies. The representative is then able to copy and paste the generated response into their own program.

For concept two, we introduced the addition of a case dashboard into our program. This dashboard gives basic information on each case such as status, assigned representative, and communication history. This extra information can be fed into the AI model when generating responses for a more context-aware and personalized response.

Last, concept 3 adds our fine-tuned model with a more fully fledged-out case management system. This design iteration will allow users to add, delete, and edit existing cases, and will feature a more informative dashboard. This dashboard will contain a full timeline of the case, and a Lexical text editor for responses with the AI response generator integrated into the system. In addition to that we also created a user authentication system using industry standards such as JWT authentication and OAuth 2.0 scheme.

**Scoring Methodology**

We will score how well each concept fits each criterion based on discussions with our advisors. For each criteria we will rank design concepts from best fit to worst fit. For example, if we had five options in a concept, the best fit would be ranked (5) and the worst fit would be ranked (1). These numbers will then be summed for a total score and for each criteria and the concept with the highest score will be chosen.

**Decision Matrix**

| Criteria | Design Concept 1 | Design Concept 2 | Design Concept 3 |
| --- | --- | --- | --- |
| Ease of Use | 3 | 2 | 1 |
| Performance | 1 | 2 | 3 |
| Reliability | 1 | 2 | 3 |
| Cost-effectiveness | 3 | 2 | 1 |
| Security | 1 | 2 | 3 |
| Response Quality | 1 | 2 | 3 |
| **Total Weighted Score** | **10** | **12** | **14** |

**Results**

After our evaluation, we decided to proceed with design concept three for our application. This concept is far more complex than the other concepts with many moving parts. While it has a lower ease of use due to its higher complexity, it makes up for it in other areas. This concept will have the highest response quality due to its access to the full communication history between the representatives and the customers. Since we are using an open source model, LLaMa, we are able to keep customer service communication data within the company, increasing security. LLaMa 3 8B is also far cheaper to use for this application than other models, as we only pay for the time it is being run. Overall, design concept three is the most difficult to implement, however, it offers the best balance between cost-effectiveness, security, and response quality, making it the best choice for our application.

### Section F. Design Methodology

The success of this project relies on a robust and iterative design methodology that guarantees the final product meets all specified objectives and client requirements. This section outlines the methods and processes that we created to help with evaluation, improvement, and validation of the design. We will discuss our computational, architectural, and validation methods.

**F.1 Computational Methods**

The primary computational method used in this project involves creating a retrieval-augmented generation (RAG) agent with access to a company knowledge base. This process will be conducted using LangGraph, Chroma DB with an embedding model, and the LLM - Qwen 2.5-7b:

1. **Data Embedding**First we must create our knowledge base for RAG functionality. To do this we will use Chroma DB - a vector database, and an embedding model - mxbai-embed-large. We will chunk our gathered company information, embed each chunk, then store in the vector database.
2. **RAG Agent**Next, we will create our workflow for the RAG agent using LangGraph. This library will allow us to give our LLM tool-calling capabilities, and in this case, the tools include RAG. For the agent we are using Qwen 2.5-7b hosted on a cloud provider. For the RAG tool we set up a function that embeds the users query and retrieves relevant information from the knowledge base for which our model can use for email draft generation.
3. **Model Evaluation**To evaluate our model, we will test its output against the information stored in the knowledge base to ensure it is generating correct information. Also, we can look at measurements such as word similarity between the model and real emails to ensure the model sounds like a real representative.
4. **Continuous Improvement**We will implement a feedback loop in production that allows representatives to rate model responses. The feedback will further fine-tune the model to create continuously enhanced responses.

#### F.2 Experimental Methods

We developed experimental methods to validate the various parts of our application and their usability for customer service representatives. Key aspects include:

1. **Testing Equipment**:   
   We can conduct testing with tools such as Postman to simulate API interactions between the frontend (React) and backend (FastAPI).
2. **Test Setup**: Representative email drafts will be generated for various customer service cases. Test cases will evaluate accuracy, response relevance, grammatical correctness, and overall professional tone.
3. **Data Acquisition and Instrumentation**: Automated logs will record response generation times, error rates, and user interactions during testing. Observational data will be collected during usability sessions with representatives.
4. **Testing Procedures**:
   * **Baseline Testing**: Evaluate the default performance of the fine-tuned LLM against standard customer service cases.
   * **Usability Testing**: Collect qualitative feedback from employees regarding the quality and efficiency of AI-generated responses.

**F.3 Architecture/High-level Design**

**The high-level architecture consists of the following components:**

1. **Frontend:**A React-based web application will provide an interface for representatives to review and edit AI-generated drafts. Core features include a dashboard, case management interface, and response editor.
2. **Backend:**FastAPI will handle authentication, response generation requests, and database management. JWT-based authentication ensures secure access to the application.
3. **Database:**PostgreSQL will store case data, response templates, representative feedback, and system logs. Data will be organized to support efficient retrieval and management.
4. **Continuous Integration/Deployment (CI/CD):**CI/CD pipelines will automate testing and deployment processes to ensure reliable updates. GitHub Actions and AWS CodePipeline will facilitate this workflow.

#### Validation Procedure

Validation is important to ensure that the system meets the client’s requirements and performs as intended. The following steps outline the validation plan:

**Performance Metrics**: The final design will be evaluated based on:

* **Response Accuracy**:  
  After we conduct fine-tuning we will give the model real-world incoming customer emails and compare the model’s responses to real representative responses. We can measure accuracy based on metrics such as word similarity between the model’s and real representatives’ responses.
* **Efficiency Gains**:  
  We can measure how long on average it would take for someone to generate a response and respond to customers using our model, and compare that to the current average customer service response time.
* **User Satisfaction**:   
  With our integrated AI generated response rating system, we can gauge how well our model is performing based on the feedback given in the ratings.

**Client Demonstration**

Our team will schedule a demonstration with CoStar in the spring semester to present a working prototype. During the demonstration, the system’s features, including AI-generated email drafts, case management functionality, and feedback integration, will be showcased. After this demonstration we can take feedback to make improvements to our application.

This design methodology ensures a robust and systematic approach to achieving a validated solution that ensures the client needs are met in our final iteration.

**Section G. Results and Design Details**

We have successfully implemented key components of the web application designed to streamline customer service email responses through AI-powered assistance. By focusing on both backend and frontend development, we’ve developed a fully functional prototype. This includes efficient case management, intuitive user interfaces, and AI integration for generating email drafts. Together, these advancements have allowed us to achieve our objectives of enhanced response times, while maintaining professional communication standards, and reducing the workload on customer service teams.

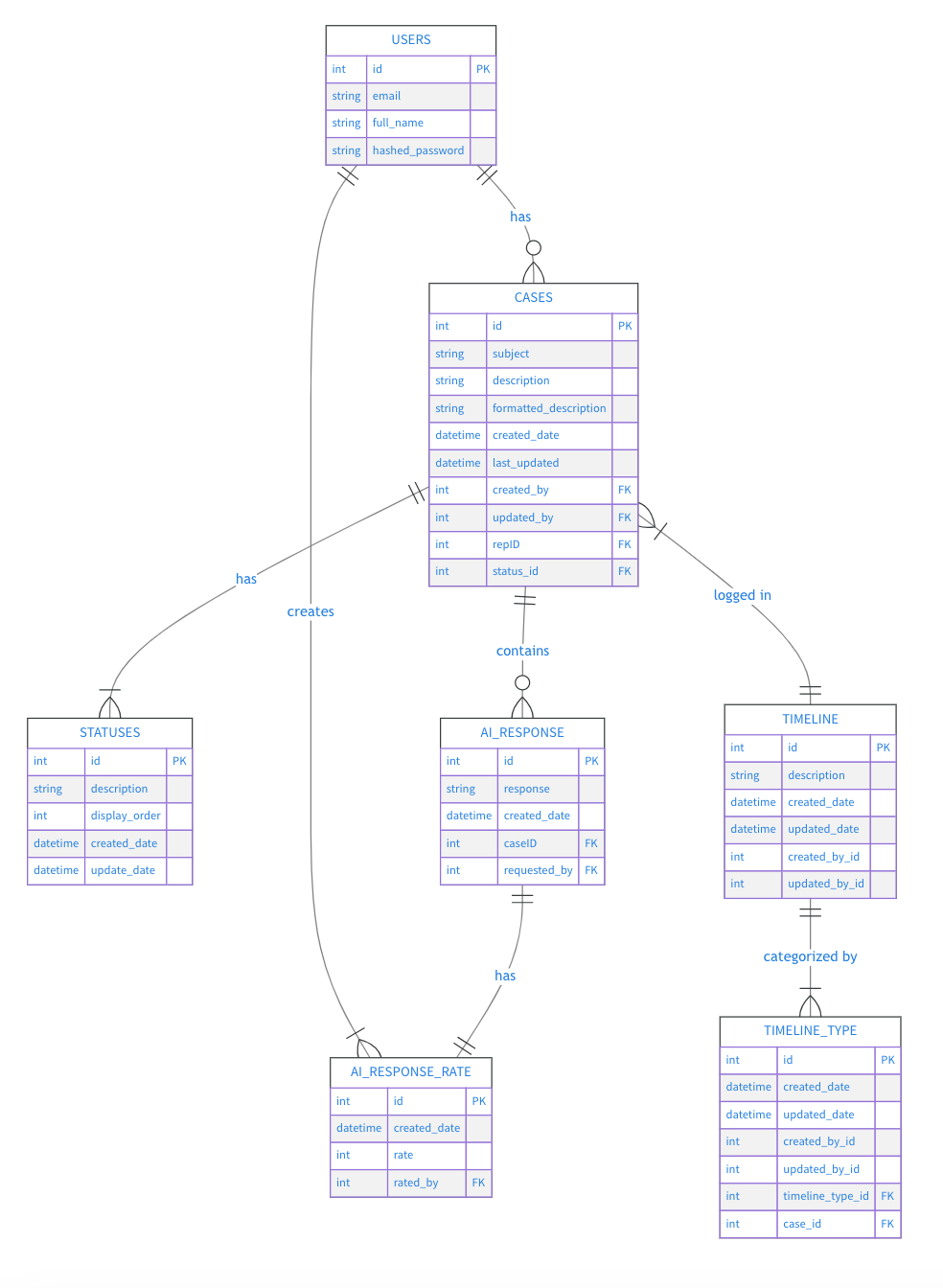
**G.1 Modeling Results**

In this subsection, we present the key models and diagrams that were essential in shaping the design and functionality of our AI-powered customer service web application. These models were pivotal in making decisions, streamlining processes, and ensuring that the system aligns with the project’s goals.

The following model is a general representation of the flow of processes in our system. It illustrates the sequence of events, from case submission to AI-generated email response, human review, and final delivery to the customer. As processes advance from one step to the next, we can also see how certain data, such as cases and AI responses are stored.

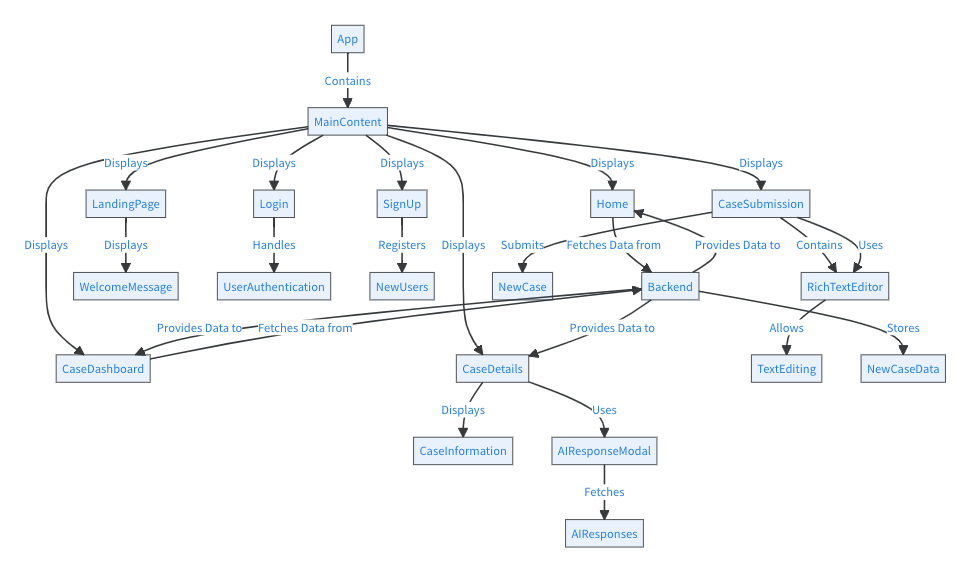


By mapping out these processes, we ensured a smooth workflow that balances automation with human oversight, preventing errors and maintaining a high standard of communication.

Using FastAPI, we stored our data in a database with the following tables: Users, Cases, Roles, Statuses, AIResponse, AIResponseRate, Messages, TimelineType, and Timeline, which are connected to our frontend to support the functionality of our application. Multiple of our tables in our database depend on each other and pull information from each other to then give to our frontend to use when producing our user interface using various CRUD functions and endpoints to, for example, submit cases, create user logins, etc.. The database schema for our project has been meticulously designed to ensure optimal performance and maintainability. Each table is tailored to store specific types of data, facilitating efficient data management retrieval. The relationships between these tables are illustrated in an Entity-Relationship diagram, which provides a clear visualization of how entities interact. 

The Users table, for example, is linked to the Cases table and AIResponseRate table, highlighting that users can create multiple cases and rates. Similarly, the Cases table has associations with Statuses, AIResponse, and Timeline, indicating various stages and actions related to a case. These relationships help streamline the process of tracking and managing customer service inquiries. By implementing this database design, we will have achieved significant improvements in managing customer service cases. The structured approach to storing data ensures consistency and reliability, while the comprehensive schema supports complex queries and data analysis. The ER diagram serves as an essential tool for understanding the data flow and interactions, making it easier for developers to navigate and modify the database as needed. Overall, creating this model has enabled us to deliver a robust and scalable solution that enhances the efficiency and effectiveness of customer service operations.

In addition to the backend database schema, our frontend design is crucial for delivering a user-friendly interface. Modern web technologies were used to create an intuitive and responsive React-based application. We created a frontend component diagram to illustrate the structure and functionality of our application. This diagram provides a detailed overview of the various components used in the frontend and how they interact to deliver a seamless user experience.



Within our app, we implemented various components, including LandingPage, Login, SignUp, Home, CaseSubmission, AIResponseModel, CaseDashboard, CaseDetails, and RichTextEditor. The diagram shows Home, LandingPage, Login, SignUp, CaseDashboard, CaseDetails, and CaseSubmission grouped together under Main Content to show everything that works together to dynamically display different views based on user interactions. The AIResponseModel component enhances the case management process by providing AI-generated responses that are reviewed and rated by employees. The RichTextEditor component allows users to craft detailed and formatted case descriptions, improving the quality of information submitted. By implementing this structured approach, we ensure that our frontend components work together harmoniously, facilitating efficient data management and user interaction. This design not only meets the project’s objectives but also enhances the overall user experience, making the application intuitive and easy to navigate. Overall, the frontend component diagram is a crucial element of our project, showcasing how we achieved a solution that aligns with our goal of improving customer service operations.

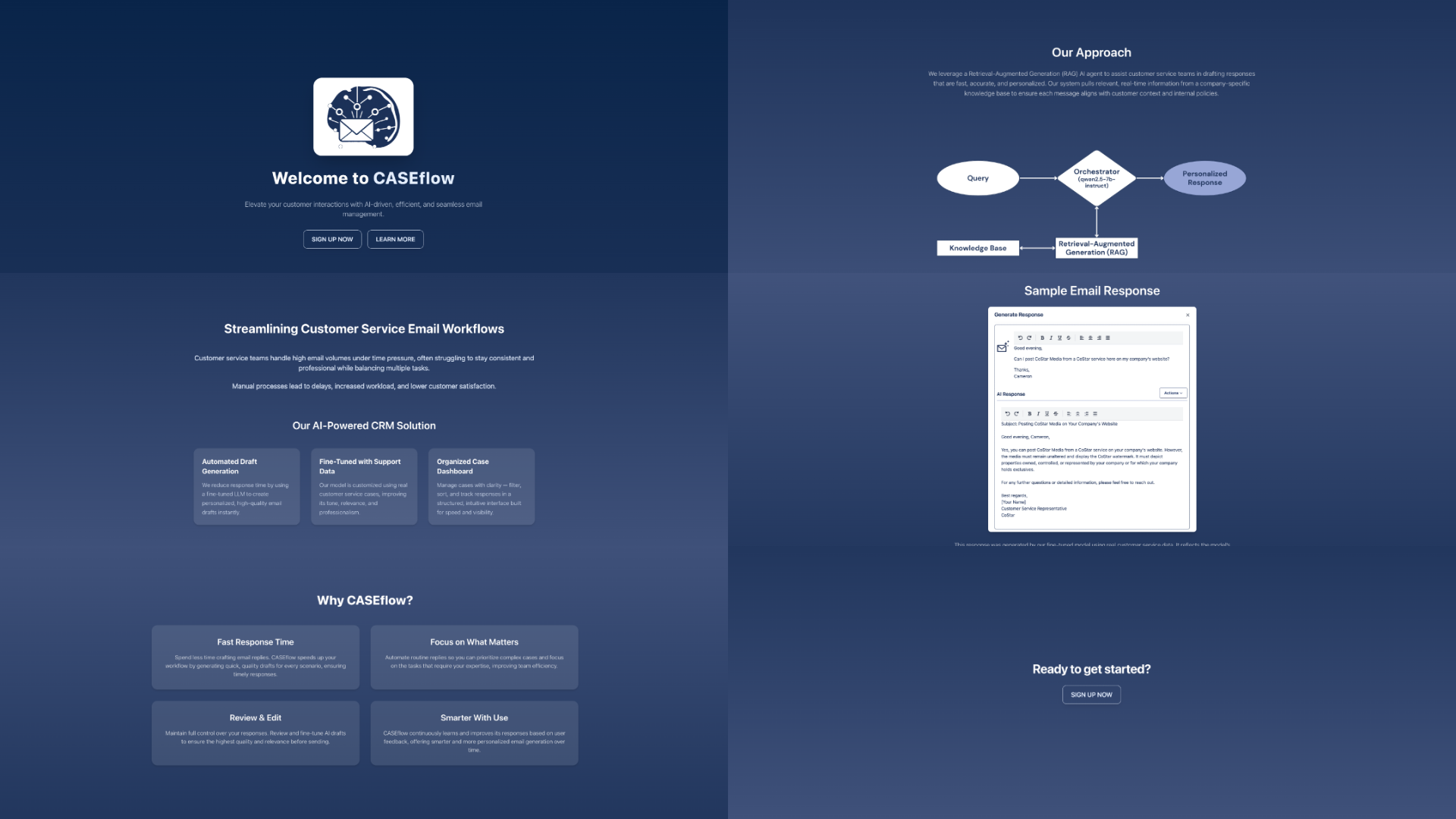
**G.2 Experimental and Testing Results**

The experimental phase involved rigorous testing of various components and algorithms to ensure that our AI model meets the desired performance standards. Further testing was done to fine-tune the LLM, using a diverse dataset of customer service interactions. The dataset was cleaned and preprocessed to find matches between incoming issues and outgoing responses. The LLM was trained on this dataset, so that the model would consistently generate high-quality responses with contextual relevance. After fine-tuning, generated text wording matched much more closely with the dataset, demonstrating that fine-tuning was successful. On the RAG-side of text generation we tested for correct information retrieval from the vector database. After many iterations of testing embedding models and varying text chunk sizes, our RAG tool was able to retrieve correct information nearly every time. Aside from fine-tuning our LLM, we developed a prototype of our web application and conducted extensive testing to validate its functionality and user experience. Our prototype includes key features such as a hero page, case submission, user authentication, a case details page, a case timeline, an AI response modal, and message history between customers and representatives. The frontend was developed using React to create a user-friendly interface. We conducted functional testing to ensure that all components of the application worked as intended. This included testing the case submission process and making sure that cases were able to be viewed once they were submitted. Additionally, there was extensive testing done with the AI response generation to ensure that the model was producing quality responses. Testing was also done to ensure that user authentication was working properly. We used JWT tokens for authentication and successfully created a functional login page. On the backend, we used FastAPI to test various endpoints to ensure their functionality. For each table we tested all CRUD operations and confirmed they were working as intended. Other endpoints, such as login, submit case, and generate/send response endpoints, were tested through the frontend with a successful login, case submission, generation of AI responses, and those responses sending as intended. Upon send, we also recorded that messages between users were successfully stored and displayed in the message history.

**G.3. Final Design Details/Specifications**

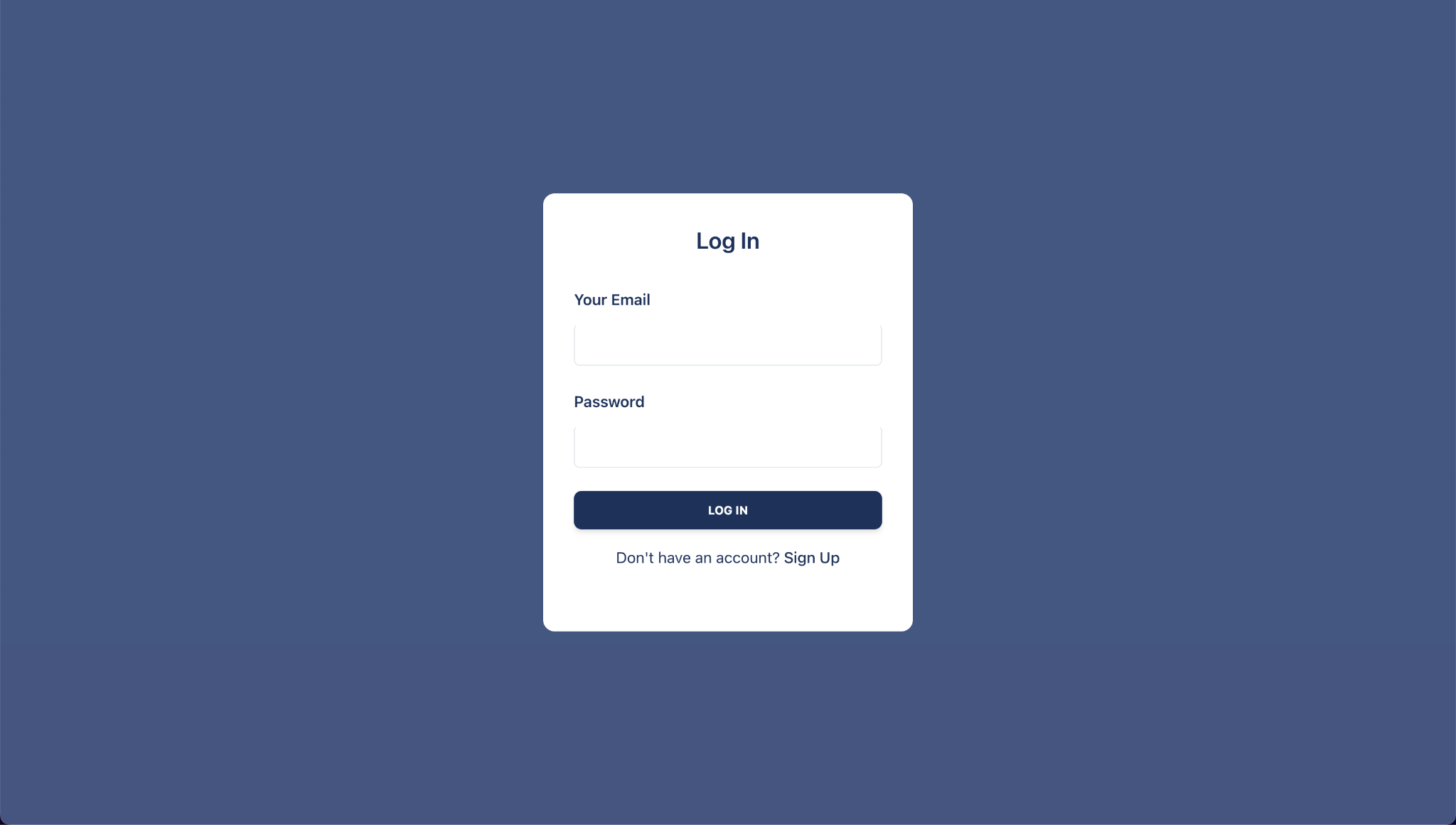
In this section, we provide an in-depth look at the final design of our AI-powered customer service web application. The following storyboard showcases the user interface and key components of the application. Each screenshot is accompanied by a brief description, highlighting the design choices, functionality, and user experience considerations.

*Landing Page*



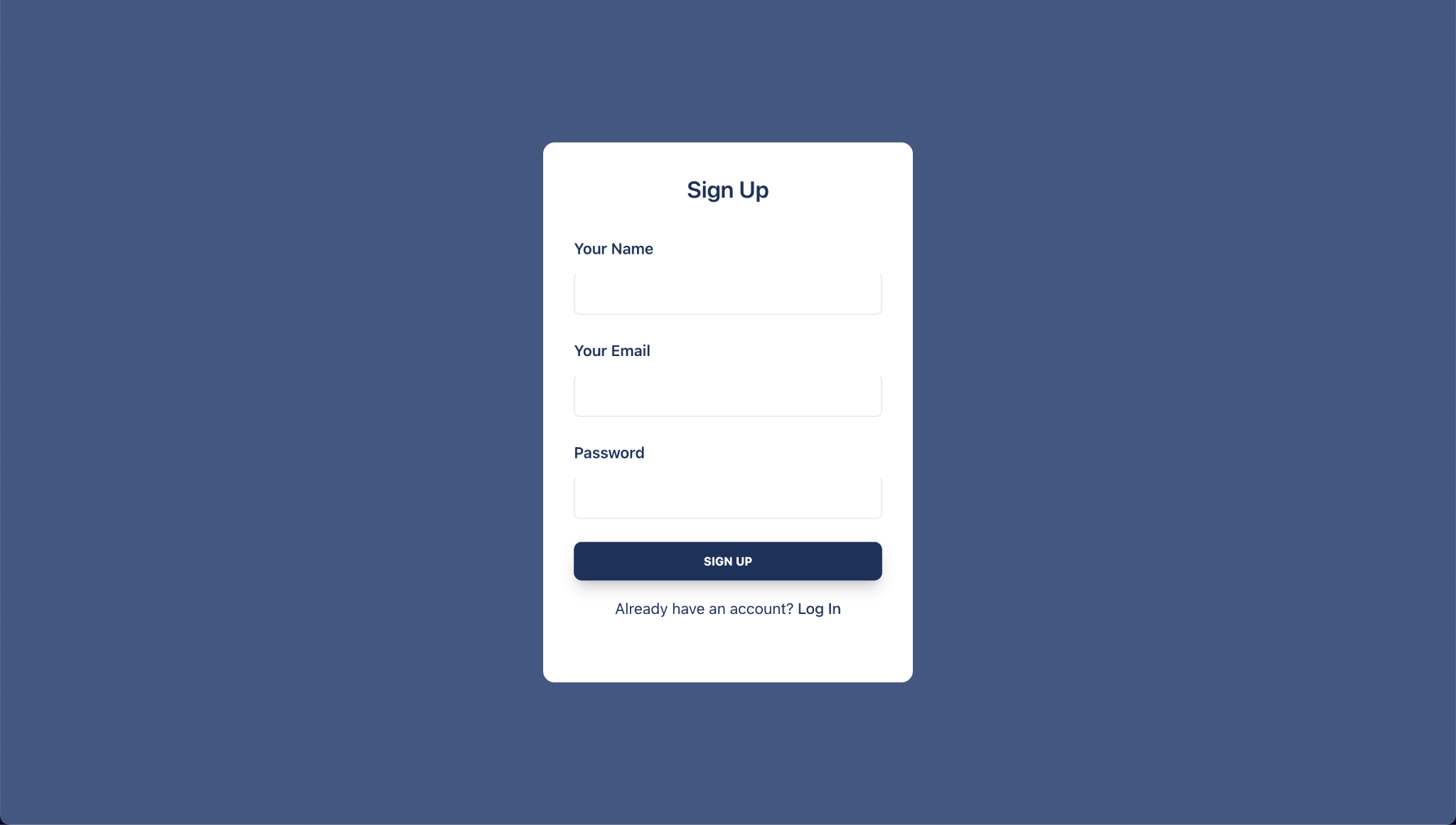
The landing page serves as the entry point for users visiting our application. The design is clean and simple, with clear navigation options for new and returning users. There are six screens that, together, ensure an intuitive and seamless experience: the welcome screen, that gives users the option of creating a new account, logging in, or learning more about our application; the problem/solution screen, which describes the problem that was presented and features of our application that we have proposed as solutions to the problem; the “Why CASEflow” screen, which details specific reasons as to why our application would be useful; the approach screen, detailing the approach behind the model we used; the example screen, displaying an example email and AI response; and the final screen, prompting the user to sign up.

*Login Page*



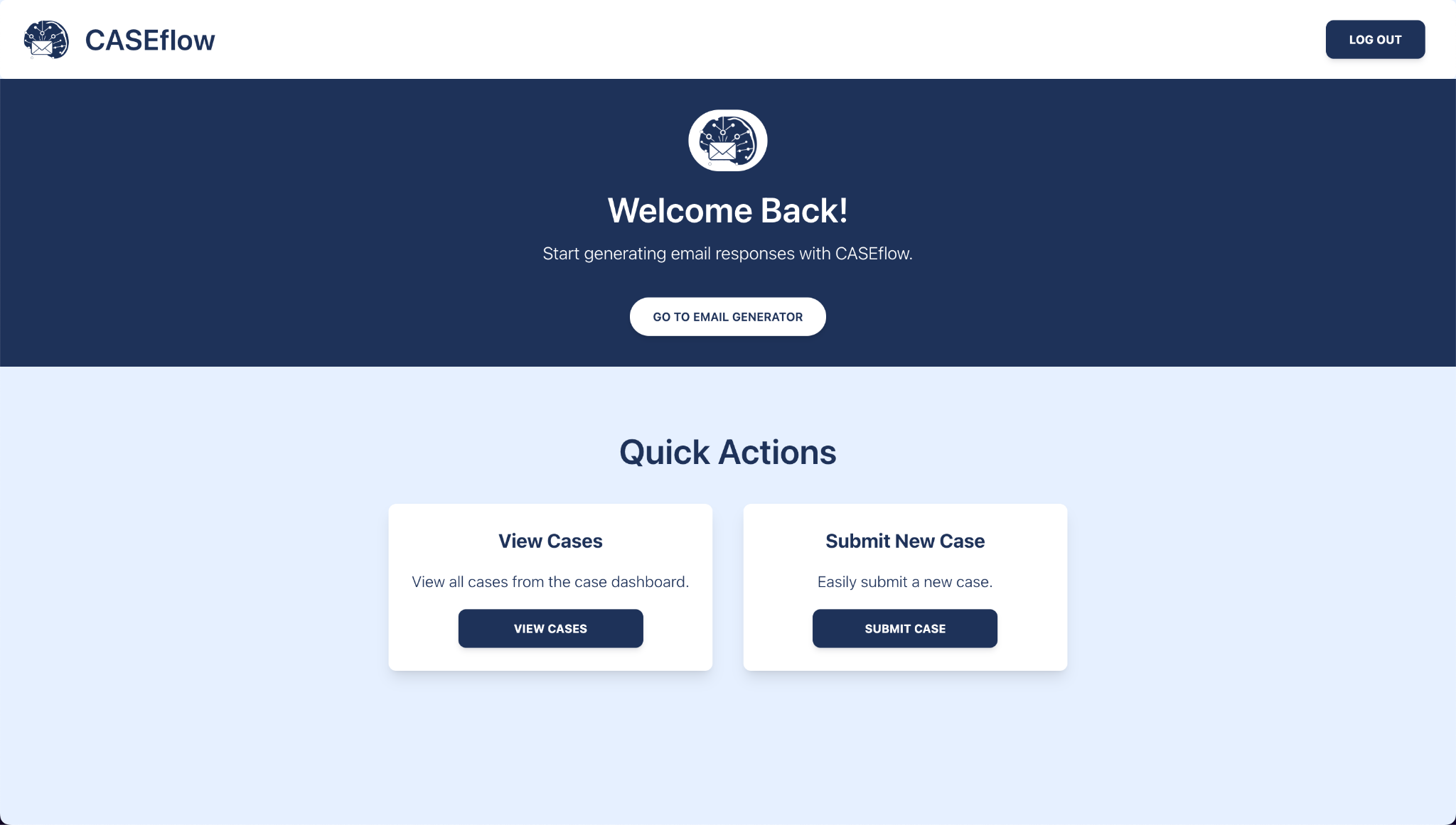
The login page is designed to provide secure access to the application. Users can enter their credentials to log in, or if you are a new user, there is an option provided to reroute them to the sign-up page. The layout is simple and user-friendly, focusing on functionality and security. This page is crucial for ensuring that only authorized users can access the system.

*Sign-up Page*



The sign-up page allows new users to create an account. The form collects essential information, such as name, email, and password. If the user already created an account, there is an option provided to reroute them to the login page. The design emphasizes ease of use, with clear instructions and validation to guide users through the registration process. This ensures a smooth onboarding experience for new users.

*Home Page*



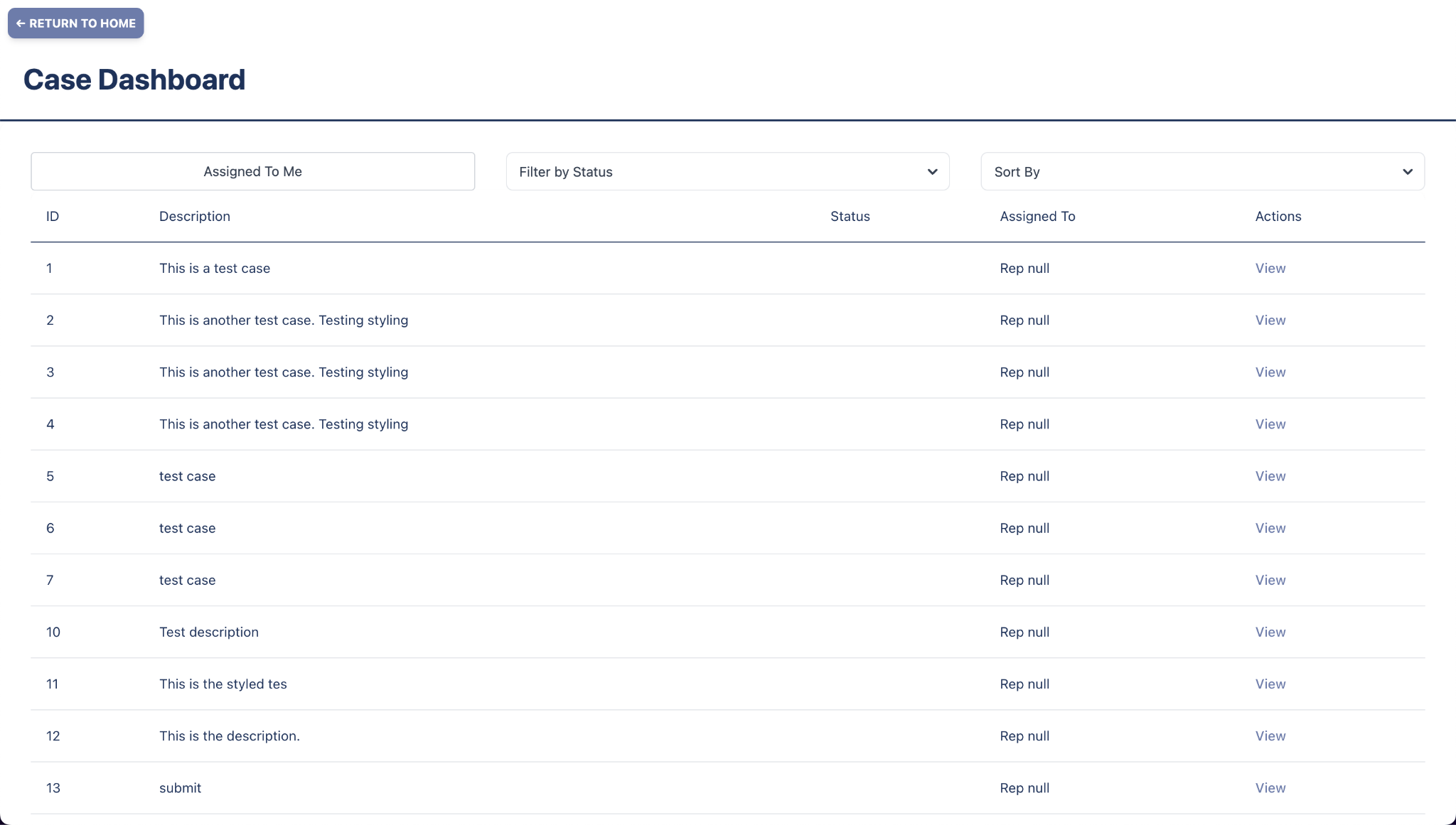
The home page serves as the central hub for users after logging in. It provides quick links to the main features, such as the case submission page and case details page. The design is very user-friendly and leaves little room for misinterpretation. Once a user is finished using the application, there is a provided logout button to securely close out of the application.

*Case Submission Form*



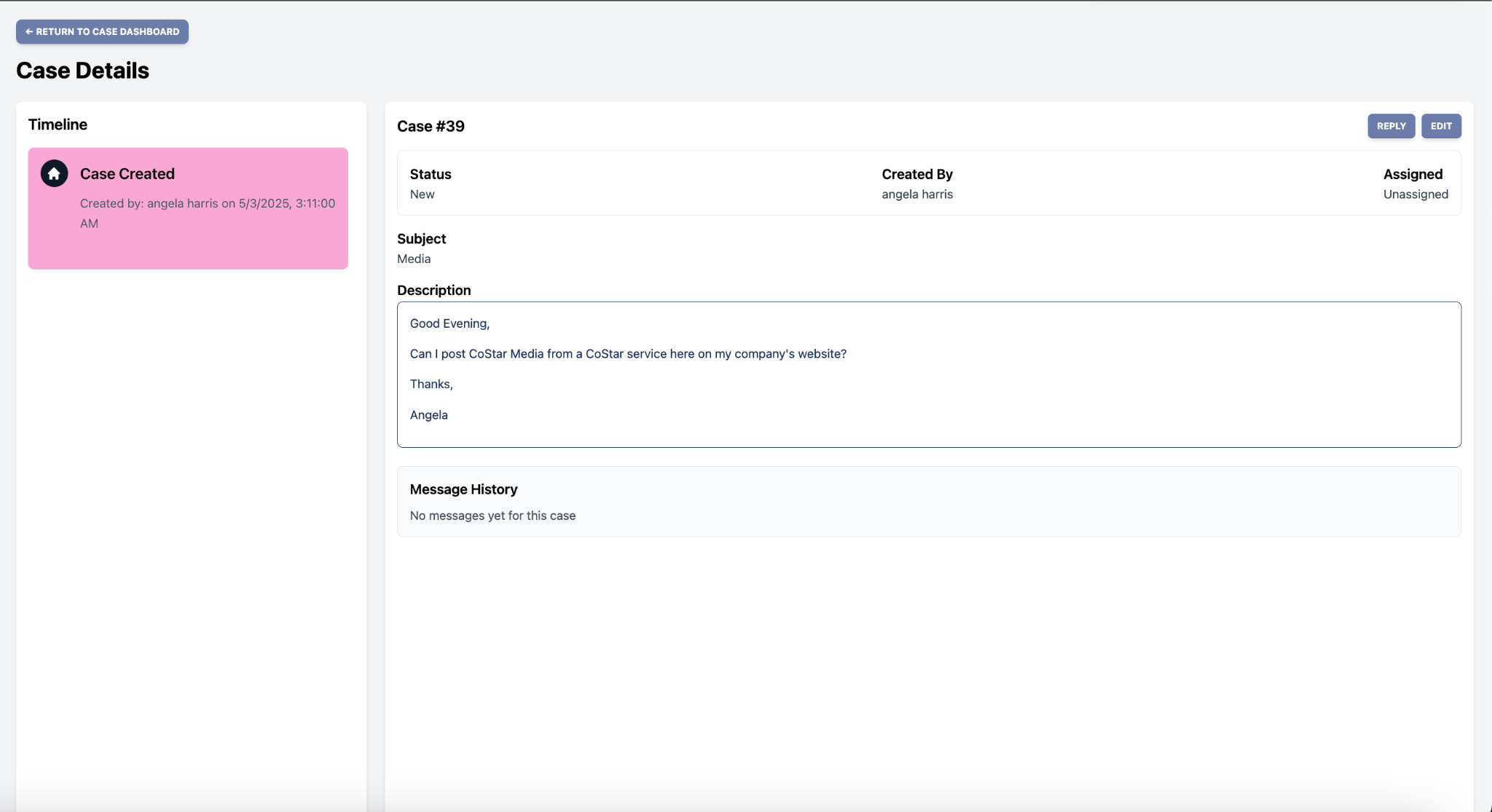
The case submission form is designed to allow users to submit new customer service cases with detailed descriptions. It includes a rich text editor, providing users with the tools to format their text as needed. The form also includes validation to ensure that all necessary fields are completed before submission, such as the case subject field. Once a case has been submitted, there is also the option to go to the case dashboard and view all cases. The design of the case submission form is simple and provides a familiar appearance to other email applications providing comfort to the user in usability.

*Case Dashboard*



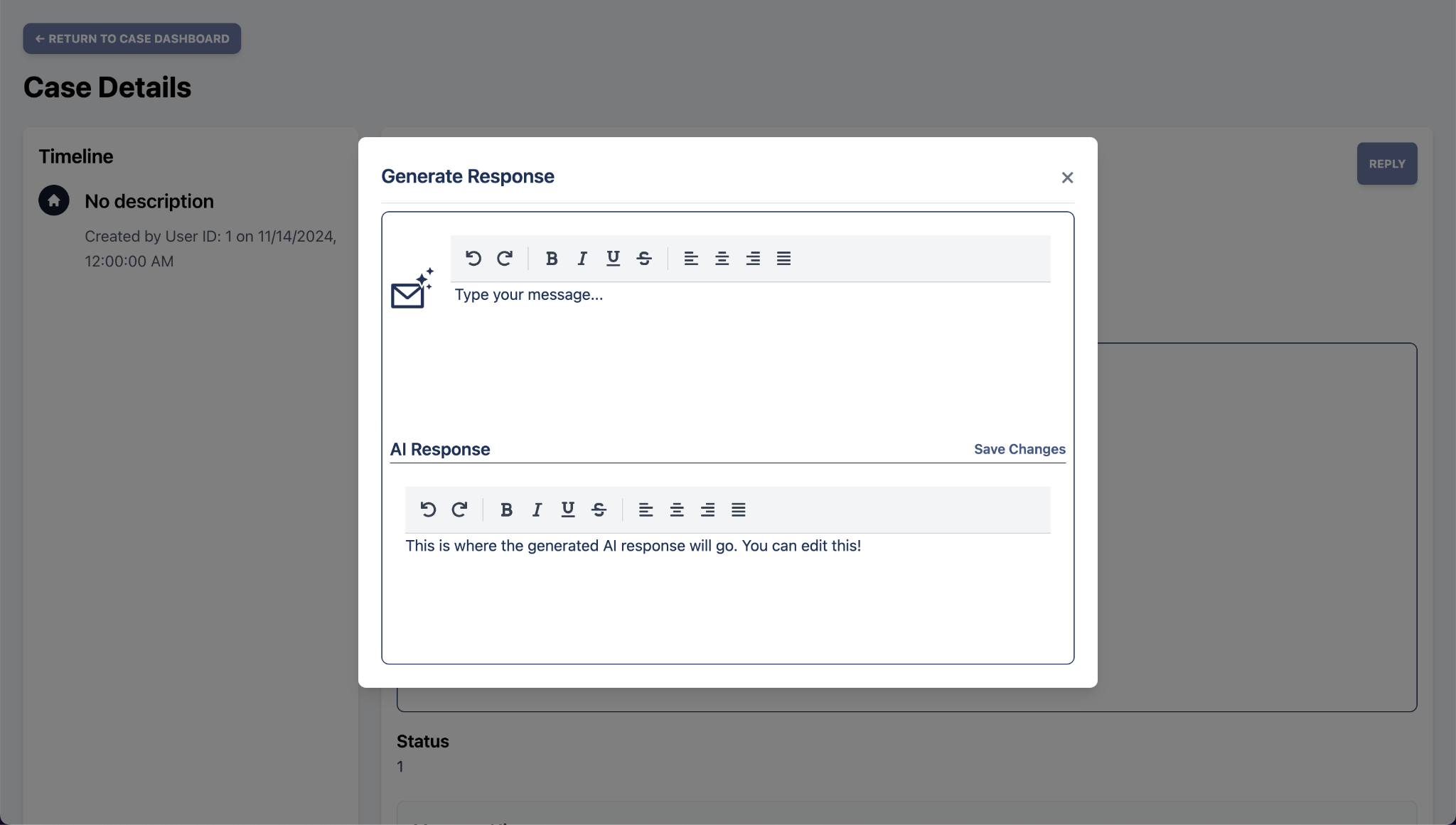
The case dashboard displays a list of all customer service cases. It provides filters and sort options to help users quickly find specific cases. Each case includes key details such as case ID, description, status, who it was assigned to, and actions taken. The design ensures that users can efficiently manage and navigate through a high volume of cases, enhancing productivity.

*Case Details Page*



The case details page provides comprehensive information about a selected case. This includes the case history/timeline, who the case was created by, the case subject, the description of the case, and the status. There is also the option to reply to the case. The layout is designed to present information clearly and concisely, ensuring that users have all the necessary details to handle complex cases effectively.

*AI Response Modal*



The AI response page allows users to view and interact with AI-generated email responses. Users can review the response with the option of making any necessary edits before sending them. The model also includes a rich text editor for styling purposes meant to improve user experience.The design focuses on usability and accessibility, ensuring that users can easily review and customize their responses as they see fit.

**Section H. Societal Impacts of Design**

**Section H. Societal Impacts of Design**

**H.1 Public Health, Safety, and Welfare** In designing our AI powered email generation and case management system, safeguarding public health, safety, and welfare has been a guiding priority. To that end, we have implemented robust authentication using JSON Web Tokens (JWT) to ensure that only duly authorized representatives can access customer data. This measure not only prevents unauthorized intrusion and social engineering attacks but also upholds regulatory requirements for data protection. All case records, draft email responses, and knowledge base embeddings are stored in a secure Chroma DB backend, encrypted at rest and in transit so that even in the unlikely event of infrastructure compromise, data integrity and confidentiality remain intact.

Our retrieval augmented generation pipeline, orchestrated via LangGraph and powered by Qwen 2.5 7b, further promotes safety by sourcing contextually relevant, up to date information from a vetted company knowledge base before drafting any correspondence. Because miscommunication can have serious consequences when advice involves legal, financial, or health related guidance, each AI generated draft must pass through a human in the loop review in a Lexical powered rich text editor. Only after explicit human approval is a response sent, thereby minimizing the risk of unsafe or misleading instructions reaching customers.

To reinforce ongoing safety assurances, we maintain a continuous feedback loop where representatives rate each AI suggestion and these ratings feed into our monitoring dashboards. By tracking metrics such as semantic similarity to approved templates and user reported concerns, we can detect and correct systematic errors, refine model behavior, and ensure that our system continually aligns with the highest standards of public welfare.

**H.2 Societal Impacts** The introduction of our advanced case management system is reshaping how customer service representatives engage with their work. By automating routine drafting tasks and surfacing relevant case history, the system enables staff to focus on complex problem solving, deepen customer relationships, and reduce burnout. From the customer perspective, response times shrink and communication quality improves, which fosters greater trust and satisfaction. As AI driven workflows become commonplace, organizations will invest in building AI literacy through training in prompt engineering, critical evaluation of language model outputs, and understanding vector database retrieval to ensure responsible adoption. While our solution is designed to augment rather than replace human roles, it will inevitably shift the skill set required for customer service positions and place greater emphasis on oversight, data governance, and strategic problem resolution.

**H.4 Economic Impacts** By streamlining initial email drafting and case data retrieval, our system can reduce average handling time by thirty to fifty percent, delivering significant labor cost savings and allowing companies to reallocate resources to high value activities. Early adopters gain a competitive edge in customer service metrics such as customer satisfaction and net promoter score, which can translate into market share growth. The demonstrable return on investment in vector databases, tool calling frameworks, and domain specific language model fine tuning is likely to spur further capital influx into these technologies and fuel innovation across the AI ecosystem. At the same time, enterprises will incur costs associated with workforce development as they design and deliver training programs on AI oversight, prompt design, and data privacy compliance to fully leverage the system’s capabilities.

**H.7 Ethical Considerations** Several ethical dimensions must be carefully managed as we deploy this AI assisted platform. We enforce rigorous data privacy policies by tagging, versioning, and subjecting every document and customer communication to strict access controls and audit logs to ensure full traceability of data usage. To promote transparency, each AI draft includes metadata such as retrieved source identifiers and confidence scores so that representatives understand the basis of every suggestion. Crucially, no message can be sent without explicit human review, which prevents blind reliance on AI outputs. We also conduct regular fairness audits on our embedding space and model responses to detect and mitigate bias and guard against skewed recommendations. Finally, both employees and, where appropriate, end customers are informed that responses are AI assisted, which preserves trust and agency. Although today’s design augments human roles, we remain committed to studying long term workforce impacts and providing reskilling pathways for any staff whose job profiles evolve.

### Section I. Cost Analysis

Our project will be incorporating AWS SageMaker and LLaMa to clean our data and fine-tune our LLM. AWS offers a pay-as-you-go pricing model, where training instances is $0.27 per hour, data processing is $0.92 per hour, and inference is $0.40 per hour. The cost of fine-tuning LLaMa models depends on the model size and training duration. We will be fine-tuning an 8 billion parameter model, which may cost around $2000 for the entire process. Based on these components, the estimated total cost for fine-tuning the LLaMa model and cleaning the data will be approximately $2000, more or less. This includes costs for training instances, data processing, and real-time inference. Fine-tuning the LLaMa model will take up the overall expense. These costs ensure that our AI-powered customer service web application operates efficiently and effectively, providing high-quality, automated email responses.

### Section J. Conclusions and Recommendations

The design team's journey toward the final design of the AI-powered email generation and case management system for CoStar Group reflects a thorough application of the engineering design process. This evolution involved iterative development, user feedback, and a commitment to addressing the specific needs of Case Management Employees.

**Evolution of the Design** The design process began with the goal of easing routine email drafting for Case Management Employees while also providing a structured way to track and resolve cases. Early prototypes used a basic Qwen 2.5 7b model to generate draft messages and a simple text editor for edits. Although this confirmed the value of AI assistance, representatives found the templates too rigid and there was no unified view of case information. User feedback made clear that a more integrated approach was needed.

**Progression Through Design Concepts** In the second iteration, a central case management dashboard was introduced alongside the AI drafting tool. Representatives could now view their open cases, check status and assignment, and generate responses in the same interface. This improved workflow visibility and efficiency, but editing remained limited to plain text and there was no formal mechanism to refine the AI model over time. The final design concept resolves these shortcomings by combining retrieval augmented generation with LangGraph querying a Chroma DB vector database (using the mxbai-embed-large embedding model) and the Qwen 2.5 7b LLM. Drafts are then reviewed and polished in a Lexical powered rich text editor. Secure access is enforced through JSON Web Tokens, and every representative action is logged and encrypted to protect sensitive customer data.

**Summary of Final Design Features** The completed platform automatically retrieves relevant company documents, FAQs and historical case records to ground each draft in up to date knowledge. The case dashboard allows creation, editing, sorting by status or assignee and an interactive timeline that shows every milestone in a case’s lifecycle. True rich text editing ensures that final messages meet corporate style guidelines. Behind the scenes, a closed-loop feedback process captures representative ratings and their final edits, feeding that data into scheduled retraining cycles so the model continually adapts to real world usage.

**Future Considerations** As the system gains traction, scaling strategies such as distributed caching or data partitioning may be needed to handle larger volumes and concurrent users without performance degradation. Regular retraining with expanded and diverse datasets will further improve the model’s understanding of nuanced customer inquiries. Structured training programs for representatives will help them master advanced features such as prompt refinement and feedback submission workflows. Integrating analytics dashboards to visualize feedback trends and service metrics will accelerate identification of improvement opportunities and guide prioritization of new capabilities.

**Unanswered Questions** Two critical challenges remain. First, how can user feedback be collected, categorized and analyzed in a standardized way that yields the most actionable insights for model refinement? Second, what monitoring and optimization techniques will best ensure that message generation and dashboard interactions maintain consistently low latency as new features and data sources are added?

**Conclusion** Through a series of focused, user centered iterations, the team has delivered an AI assisted case management and email generation platform that meets CoStar Group’s operational requirements while laying a robust foundation for future innovation. Comprehensive documentation of testing procedures, architectural designs and user guides has been archived to support ongoing development. This work positions the organization to deliver faster, more accurate and more personalized customer service as the AI powered solution continues to evolve.

### 

### 

### 

### 

### 

### 

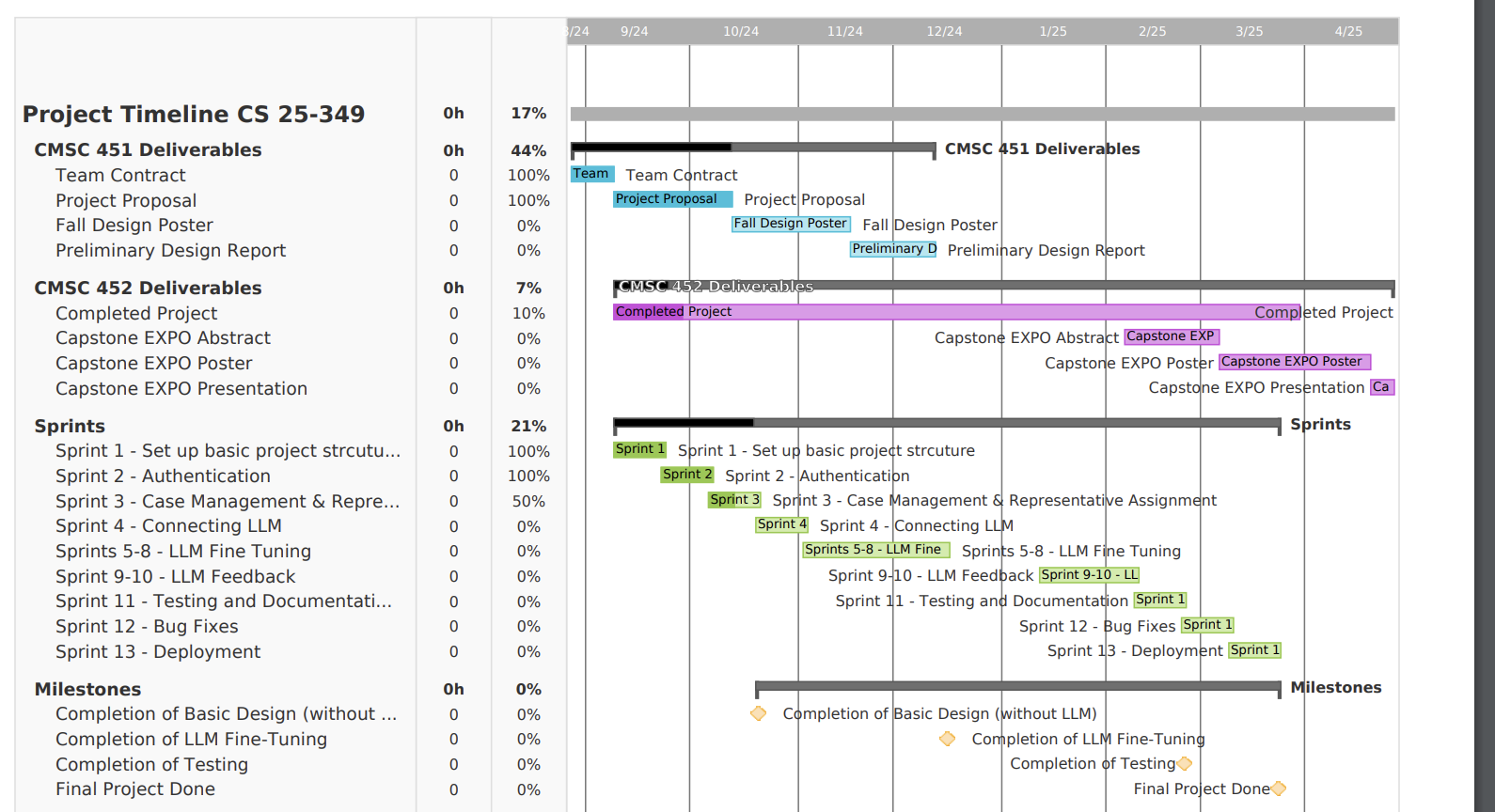
### 

### 

### 

### Appendix 1: Project Timeline

The figure below illustrates the timeline of our project in Gantt Chart form. The categories of tasks are as follows: CMSC 451 Deliverables, CMSC 452 Deliverables, Sprints, and Project Milestones. The sprints section will change as the team is assigned sprints.



### Appendix 2: Team Contract (i.e. Team Organization)

#### Step 1: Get to Know One Another. Gather Basic Information.

| ***Team Member Name*** | ***Strengths each member bring to the group*** | ***Other Info*** | ***Contact Info*** |
| --- | --- | --- | --- |
| Angela Harris | Organization, quick learner, communication, flexibility | Strong believer in the idea that there are no “dumb” questions | harrisam2@vcu.edu |
| Emma Smith | Problem-solving, React experience, leadership | I am passionate about learning new things and enjoy collaborating with others. | smither3@vcu.edu |
| Sohil Marreddi | Eager to learn from others, previous project experience | Intrigued to build real world products and create that product from scratch. | marreddiss@vcu.edu |
| Cameron Clyde | Adaptable, team-work, quick learner, front-end experience. | I'm always looking for ways to improve my knowledge and skills. | clydecp@vcu.edu |

| ***Other Stakeholders*** | ***Notes*** | ***Contact Info*** |
| --- | --- | --- |
| Faculty Advisor: Preetam Ghosh,  VCU Engineering |  | pghosh@vcu.edu |
| Sponsor:  Keroles Hakem, CoStar Group |  | khakem@costar.com |

#### Step 2: Team Culture. Clarify the Group’s Purpose and Culture Goals.

| ***Culture Goals*** | ***Actions*** | ***Warning Signs*** |
| --- | --- | --- |
| Attend every meeting | * Set up meetings in shared calendar * Send reminder message in group chat day of meeting * Discuss when there are schedule conflicts/reschedule as needed | * Student misses meeting without communication |
| Holding each other accountable to stay on top of work | * Work on project weekly * Discuss goals for the week and add them to meeting notes * Set reasonable deadlines for goals | * Student shows up for weekly meeting with no considerable work done |
| Creating a safe environment with open communication | * Talking through problems/conflicts as they arise * Communicate schedule * Open to new ideas | * Shutting down other teammates ideas * Passive aggression |

#### Step 3: Time Commitments, Meeting Structure, and Communication

| ***Meeting Participants*** | ***Frequency***  ***Dates and Times / Locations*** | ***Meeting Goals***  ***Responsible Party*** |
| --- | --- | --- |
| *Students Only* | *As Needed, On Discord Voice Channel or Zoom* | *Discuss updates on progress and challenges*  *Work through challenges as group and discuss potential solutions*  *(Angela will record notes and upload them to shared Google Drive)* |
| *Students Only* | *Every Monday 7-9pm in library room* | *Actively work on project, discuss any difficulties*  *Review previous weeks progress and discuss goals for upcoming week*  *(Emma will take notes and upload them to shared Google Drive)* |
| *Students + Faculty advisor* | *As Needed* | *Update faculty advisor and get answers to our questions*  *(Sohil will take notes on shared Google Docs; Angela will organize and lead meeting)* |
| *Students + Project Sponsor + Faculty advisor* | *Thursday at 6PM on Zoom* | *Update project sponsor to ensure we are on the right track*  *Ask questions and discuss challenges*  *(Cameron will take notes; Emma will organize and lead meeting; Sohil will demo prototype so far and give updates)* |

# 

#### 

#### 

#### Step 4: Determine Individual Roles and Responsibilities

| ***Team Member*** | ***Role(s)*** | ***Responsibilities*** |
| --- | --- | --- |
| Angela Harris | Logistics Manager | * Primary contact for communication with faculty advisor * Documenting meeting times * Obtaining information for the team * Following up on communication of commitments |
| Emma Smith | Project Manager | * Primary contact for communication with sponsor * Create a welcoming environment at meetings * Document and organize team goals for week * Schedule weekly meetings with sponsor * Book library rooms for student meetings |
| Cameron Clyde | Test Engineer/Financial Manager | * Monitors team budget. * Oversees experimental design, test plan, procedures and data analysis * Leads presentation of experimental finding and resulting recommendations |
| Sohil Marreddi | Systems Engineer | * Takes notes of client/sponsor requirements to put together solutions * Works with team members to design architecture of the software |

# 

#### Step 5: Agree to the above team contract

*Team Member:* Angela Harris *Signature: Angela Harris*

*Team Member:* Sohil Marreddi *Signature: Sohil Marreddi*

*Team Member:* Cameron Clyde *Signature: Cameron Clyde*

*Team Member:* Emma Smith *Signature: Emma Smith*

### 

### 

### 

### References

[1] A global leader in the digital transformation of the $300+ trillion real estate industry. Home | CoStar Group. (2024). https://www.costargroup.com/

[2] Gil-Gomez, H., Guerola-Navarro, V., Oltra-Badenes, R., & Lozano-Quilis, J. A. (2020). Customer relationship management: Digital Transformation and Sustainable Business Model Innovation. *Economic Research-Ekonomska Istraživanja*, *33*(1), 2733–2750. https://doi.org/10.1080/1331677x.2019.1676283

[3] Mesquita, T., Martins, B., & Almeida, M. (2022, October). Dense template retrieval for customer support. In Proceedings of the 29th International Conference on Computational Linguistics (pp. 1106-1115). https://aclanthology.org/2022.coling-1.94/

[4] Sheth, J. N., Jain, V., & Ambika, A. (2024). Designing an empathetic user-centric customer support organization: Practitioners’ perspectives. European Journal of Marketing, 58(4), 845–868. https://doi.org/10.1108/ejm-05-2022-0350

[5] Malik, R., Subramaniam, L. V., & Kaushik, S. (2007, January). Automatically selecting answer templates to respond to customer emails. In IJCAI (Vol. 7, No. 1659, p. 3015). https://www.researchgate.net/profile/Lv-Subramaniam/publication/220815935\_Automatically\_Selecting\_Answer\_Templates\_to\_Respond\_to\_Customer\_Emails/links/55b6580e08aec0e5f436fe0e/Automatically-Selecting-Answer-Templates-to-Respond-to-Customer-Emails.pdf