Course Submission Cover Sheet

Module: CS6003ES Advance Software Engineering

Assignment no: 001

Weighting: 30%

Deadline: TBC

Module Leader: **Student ID: KAN00257474**

*Please note that there are specific regulations concerning **the use of AI and Academic Misconduct**. Below are extracts from these regulations. By signing, you acknowledge that you have read and understood these extracts.*

*(signature:)*_____ *Date:* *02/02/25*

This header sheet should be attached to the work you submit.

Academic Integrity means being honest in your academic work and your studies and making sure that you acknowledge the work of others and giving credit where you have used other people's ideas as part of presenting your arguments. Your assessment submissions must therefore always be entirely your own work, based on your own learning and appropriately referenced including how you have used Generative AI. The University regards the use of Generative AI applications by students to deceive to gain unfair advantage as **academic misconduct**. This usage includes:

- **Plagiarism**, where AI tools are used to generate output and ideas that are presented or submitted as if they were the student's own work, without proper citation or references.

- Where a complete assignment is created using Generative AI and represented as a student's own work, this will be regarded as contract cheating in the same way as commissioning an 'Essay Mill' or other third party to complete your work. Further information can be found on : Guidance on the use of Artificial Intelligence.

Academic misconduct: The University takes academic misconduct very seriously and seeks at all times to rigorously protect its academic standards. Plagiarism, collusion and other forms of cheating constitute academic misconduct, for which there is an explicit range of graduated penalties depending on the particular type of academic misconduct. The penalties that can be applied if academic misconduct is substantiated range from a reprimand to expulsion in very serious cases and for repeated instances of misconduct. You are also responsible for ensuring that all work submitted is your own and that it is appropriately referenced. The University does not tolerate cheating of any kind. You are strongly advised

Contents

Chapter 1

1.1 Project Background and Motivation

1.1.1 Background

Manual procedures lead to inefficiencies in Sri Lanka's LP gas distribution, which result in inadequate tracking, delivery delays, and stock shortages. Customers frequently visit stores without guarantees of supply, and store managers find it difficult to manage inventory and schedule delivery. These difficulties demonstrate the necessity of an online system for gas distribution and requests in order to maximize efficiency. With real-time connections between customers, retailers, and dispatch centers, such a system can offer precise stock updates, fair distribution, and expedited delivery times. It turns the current fragmented method into a dependable, effective solution for homes and businesses by lowering manual errors and increasing transparency, which guarantees a smooth supply chain and increased customer satisfaction.

1.1.2 Motivation

This system's implementation is motivated by the increasing need for efficient and convenient service delivery. The widespread use of smartphones and digital platforms has led to customers expecting easy-to-use solutions for everyday problems. With its updated solution that puts the ease of its customers first, GasByGas hopes to meet these expectations. The system will also greatly increase GasByGas's operational efficiency. Technology can help the business cut expenses, cut down on waste, and increase customer loyalty by managing customer communications, scheduling delivery, and tracking inventories. Additionally, by cutting down on needless travel and paper-based procedures, the program contributes to the larger objectives of sustainability and digital transformation.

1.2 Project Scope and Objectives

1.2.1 Objectives

- Allowing end consumers to place orders online and receive tokens with an anticipated pickup time in order to streamline gas demand.
- To increase the effectiveness of distribution by giving outlets automated tools to control stock availability and delivery timetables.
- To guarantee efficient communication by sending consumers email or SMS notifications about delivery dates and other pertinent information.
- To fairly manage stock by restricting requests according to personal information (NIC, phone number, or email) and redistributing unfulfilled demands as needed.
- To provide distinct request categories for households and businesses/industries, with appropriate validation for the latter, in order to serve a variety of client segments.
- To give the dispatch office and outlets a reliable tracking system that guarantees correct record-keeping and on-time delivery.
- To provide real-time information on requests, delivery schedules, and stock levels in order to enhance decision-making.

1.2.2 Scope

All stakeholders involved in the gas distribution chain, including end consumers, outlet managers, and the central dispatch office, will be served by the system. Features including online request submission, token generation, stock availability tracking, delivery scheduling, and notifications will all be included. To guarantee seamless operations, the platform will integrate with the current supply chain infrastructure. For business and industrial clients, distinct modules will be created to manage large orders and specialized certifications.

The service will be available to clients in urban, suburban, and rural regions of Sri Lanka due to its comprehensive geographic coverage. To avoid misuse, the system will also include strong user authentication features connected to NICs, phone numbers, and emails. Reliability and customer trust will be increased through integration with SMS and email services, which will guarantee prompt communication about stock status and delivery schedules.

Chapter 2

2.1 Overview of Existing System

The current system is primarily manual and lacks an internet platform to track deliveries, facilitate gas requests, or effectively manage scheduling. Consumers have to physically visit stores to lodge requests or check availability, which frequently results in misunderstandings, delays, and a lack of transparency in the distribution process. Likewise, business and industrial customers use conventional registration and gas request processes, which makes it difficult to verify credentials and optimize their supply. Inefficiencies brought on by this lack of automation and integration include supply shortages, haphazard deliveries, and poor customer communication.

2.1.1 Drawbacks of the Existing System

The existing system of GasByGas faces several drawbacks that hinder its efficiency and customer satisfaction:

Manual Process: Customers experience delays and hassles when gas requests and availability checks are handled by physical visits to outlets.

Lack of Transparency: Customers are unclear and unsatisfied since they are unable to monitor gas supply or delivery timetables.

Inefficient Communication: The absence of automatic notifications for deliveries, stock updates, or schedule modifications causes a communication breakdown between the business, its retail locations, and its clientele.

Limited Accessibility: Customers are unable to remotely place gas requests or get tokens due to the lack of an internet platform, which limits accessibility.

Inefficient Resource Allocation: Reallocation procedures and the physical distribution of gas cylinders to clients are laborious and prone to mistakes.

Consumer Dissatisfaction: Poor customer experiences are a result of ineffective request processing, delivery delays, and a lack of real-time updates.

2.2 Proposed Solution

To improve customer satisfaction and expedite GasByGas' distribution operations, an online gas request and delivery management system is the proposed solution. This system will provide real-time gas distribution tracking, enhance communication, and automate important procedures. The following are important aspects of the proposed solution:

- By choosing the preferred outlet, gas type, and quantity, customers can place an online or mobile gas request.
- Customers will receive a token with a two-week tolerance window and an anticipated pickup period after submitting a request.
- When the gas is ready for pickup or if a rescheduled time is required because of supply limitations, the system will arrange deliveries to outlets and send customers an SMS or email.
- To avoid abuse and overbooking, the system will restrict requests based on personal identity verification, such as NIC, phone number, or email.
- To guarantee that cylinders are picked up on schedule, customers will receive automated SMS/email reminders the day before their delivery date.

By implementing this solution into practice, GasByGas will guarantee a gas distribution procedure that is more effective, transparent, and customer-friendly while lowering operational inefficiencies and stock-related difficulties.

2.3 Feasibility Study

The proposed Online Gas Requesting and Delivery Management System for GasByGas is assessed for practicality and potential success through a feasibility study. Here are the main points:

Technical Feasibility

The frontend of our system is WordPress, the database is MySQL, and the backend is Java Spring Boot. Strong security, quick API development, and effective business logic handling are all made possible by Spring Boot. With its robust content management system and plugin compatibility, WordPress offers an intuitive user interface. High-performance, dependable data storage is guaranteed by MySQL. This stack is perfect for our platform since it strikes a mix between performance, efficiency, and maintainability.

Operational Feasibility

The platform will minimize the need for intensive training by prioritizing usability with a user-friendly interface that is intended for administrators, employees, and consumers. Gas requests, scheduling, and alerts will all be automated, streamlining operations and improving customer and employee efficiency. A committed support staff will also make sure the system functions properly with frequent upgrades, resolving any problems and preserving peak performance.

Legal Feasibility

The system will be built in accordance with Sri Lanka's data protection regulations, guaranteeing the safe management of private data, including phone numbers and NICs. The platform will apply certification validation procedures for corporate and industrial demands, guaranteeing that all gas distribution complies with industry norms and governmental laws. This guarantees all transactions are safe and compliant with the law.

2.4 Requirement Collection

2.4.1 Requirement Gathering Techniques

A critical stage in system development is requirement gathering, which guarantees that the proposed solution satisfies the demands of the company and its clients. Requirements for the Online Gas Requesting and Delivery Management System will be gathered using the following methods:

Interviews: Key stakeholders, such as GasByGas management, outlet managers, and end users, will be interviewed one-on-one. The expectations, operational difficulties, and particular needs for the new online system will be determined in part by these interviews. End users can share their experiences with the current manual process, outlet managers can draw attention to logistical and inventory-related issues, and management can offer insights into the overall distribution strategy.

Questionnaires: Customers, store employees, and other pertinent parties will receive surveys to collect organized input on the current system. Aspects including stock shortages, communication breakdowns, service delays, and expectations for an online request and delivery tracking system will all be covered in these questionnaires. The system design can be more closely matched to the real needs of users by gathering both quantitative and qualitative data.

Observations: Daily operations at different GasByGas locations and service facilities will be directly observed. This approach will assist in identifying inefficiencies in the current workflow, including order processing delays, customer service problems, and stock management difficulties. Potential areas for improvement can be identified by examining customer-staff interactions in real time.

Document Analysis: To learn more about the current system's flaws, existing records such as process flows, manual logs, and customer complaints will be examined. By examining these documents, important information about persistent issues like missed deliveries, inventory

discrepancies, and communication breakdowns will be obtained. This approach will also assist in comprehending the standard operating procedures and regulatory requirements that need to be incorporated into the new system.

2.4.2 Justification for Requirement Gathering

Key stakeholders, including GasByGas management, outlet managers, and customers, can be interviewed to better understand their particular needs and concerns. This will help the project develop a more focused and efficient solution.

By using questionnaires, customers and outlet employees can provide more detailed feedback on present problems and their expectations for the new system.

Interviews alone might not always reveal real-world inefficiencies and pain areas, but observations at outlets might help.

Furthermore, examining current documentation such as process flows and customer feedback provides insightful information about the deficiencies of the current system, which aids in improving design choices and guaranteeing that the new system fixes these issues.

Building a system that is user-centric, efficient, and successful requires a thorough approach to requirements collecting, which is ensured by combining these several approaches.

Chapter 3

3.1 Functional and Non-Functional Requirements

3.1.1 Functional Requirements

Gas Request Management:

Gas requests are easily submitted by customers using a mobile application or website. They receive a token with an anticipated pickup time upon successful submission, guaranteeing transparent communication. In order to guarantee effective operations and client satisfaction, the system intelligently verifies stock availability and scheduled delivery times, blocking requests when resources are inadequate or unavailable.

Delivery Scheduling:

Using the system, dispatch in the head office may effectively organize and oversee delivery schedules for every outlet. Token holders are instantly alerted by email as soon as a delivery date is established. In order to ensure efficient inventory management and minimize waste, the system also allows outlet managers to reallocate gas supply in the event that a customer does not meet the pickup conditions.

Notifications and Alerts:

One day prior to the scheduled delivery date, customers will receive automatic reminders by email to make sure they are ready. To ensure transparency and improve the customer experience, the system will promptly notify consumers of rescheduling or cancellations in the event of product shortages or delays.

Identity-Based Request Management:

The system uses personal identifiers like NIC, phone number, or email to enforce limits on customer requests. This promotes effective inventory management and equitable service distribution by guaranteeing equitable access to resources, preventing abuse, and avoiding overbooking.

Industrial and Business Client Management:

The system has distinct registration and validation procedures for business and industrial clients, and certification verification is necessary to guarantee regulatory compliance. Requests from the industrial sector are prioritized, and specific notification and delivery systems are set up to quickly and effectively meet their particular needs.

Inventory Management:

The solution streamlines the replenishment process by allowing outlets to request replenishing directly from central office. In order to assist avoid shortages and guarantee steady availability for clients, the system additionally offers real-time tracking of stock levels at every shop.

Admin Controls:

Head office administrators have comprehensive control to monitor, update, and manage system-wide operations, ensuring seamless coordination across outlets. Outlet managers are granted specific permissions tailored to managing local operations, including stock handling and customer interactions, enabling efficient and localized decision-making.

3.1.2 Non-Functional Requirements

Performance:

The system will be built to efficiently support at least 1,000 users at once without experiencing any performance issues.

Scalability:

Scalability will be considered throughout the system's development to guarantee that it can accommodate future growth. As the company expands, it will manage higher user traffic, support the opening of new locations, and enable the smooth integration of new services.

Availability:

A 99.9% uptime will be maintained by the system's design, guaranteeing clients continuous service and reducing interruptions. Reliable infrastructure, routine maintenance, and backup systems to guarantee continuous accessibility will all help achieve this high availability.

Security:

To preserve user privacy and comply to data protection laws, customer information, including NICs, phone numbers, and emails, will be safely saved and encrypted. Only authorized staff will be able to access sensitive data and carry out particular tasks according to their roles within the system thanks to the system's strong user authentication and role-based access controls.

Usability:

The system will have user-friendly interfaces that lower the learning curve for new users. The platform will reduce the amount of time needed for training and enhance the user experience by emphasizing simplicity, straightforward navigation, and user-friendly design, which will allow users to easily comprehend and utilize the system.

Reliability:

The system will be built to recover from malfunctions in less than a minute, guaranteeing clients minimal downtime and uninterrupted service. With safeguards in place to preserve data during system updates or unplanned shutdowns, data integrity will be given high priority, avoiding any loss or corruption of important information.

Chapter 4

4.1 Architecture Designing

A client-server architecture would be used in the proposed online gas request and delivery management system. Clients, or users who access the system through online or mobile applications, and a server, or central system that organizes data, processes requests, and facilitates communication between various entities, make up this model.

Centralized Control & Management:

A central server will stored and managed all scheduling, inventory, and gas request data, guaranteeing data consistency, simple monitoring, and effective coordination amongst all outlets. Administrators will be able to monitor stock levels, oversee deliveries, and quickly address consumer demands thanks to the centralized system's real-time data access capabilities.

To preserve data integrity and guarantee continuous service, strong backup and security procedures will also be put in place.

Scalability:

The system is designed to scale effectively, meaning that as the number of clients, outlets, and requests grows, performance won't suffer. Its adaptable design makes it possible to add more users and outlets without requiring significant changes to the main system. Optimized database administration, cloud-based architecture, and load balancing guarantee seamless operations even as demand increases. This scalability guarantees long-term dependability and flexibility for upcoming company growth.

Security & Data Integrity:

Sensitive information, such phone numbers and NICs, should be kept on a centralized, secure server to improve security, stop illegal access, and guarantee that data protection laws are followed. This information is further protected by encryption and frequent security checks. In order to protect privacy and system integrity, role-based access control also makes sure that only authorized users such as administrators, outlet managers, and customers can access particular data in accordance with their permissions.

Efficient Communication:

Serving as a central hub, the server makes it easier for customers, outlets, and the dispatch office to communicate with one another. It guarantees easy coordination between all parties, effective request processing, and real-time data synchronization. The server also controls

automated email and SMS notifications, which minimizes manual communication efforts and guarantees prompt updates on request status, delivery timetables, and any required alarms.

Better Performance & Load Handling:

Optimized performance and a seamless user experience are guaranteed by the system's efficient workload distribution between the client (frontend) and server (backend). While the backend handles intricate tasks like request processing, inventory monitoring, and authentication, the frontend manages user interactions and reduces the processing burden on consumer devices. This architecture guarantees smooth operation on both web and mobile platforms while improving response times and lowering latency.

Client-Server Architecture's many benefits in terms of security, scalability, performance, and centralized management make it the perfect choice for the Online Gas Requesting and Delivery Management System. The system is divided into two primary parts by this architecture: a server (which houses centralized processing and data storage) and clients (which are programs that interact with users). The smooth and dependable running of the entire system is ensured by this separation, which permits effective resource allocation and optimum operations.

The centralized control and data management of client-server architecture is one of its main advantages. According to this concept, a central server houses and processes all important data, such as delivery schedules, inventory information, and client requests. Data administration is made simpler by this centralized approach, which guarantees that data is current and consistent for all users and outlets. Applying security mechanisms, including role-based access control and data encryption, to safeguard private user data, like NIC numbers, contact information, and transaction history, is also made more simpler by this design.

Additionally, one of the main benefits of the Client-Server approach is its scalability. The server can be expanded or upgraded to accommodate increased traffic and data loads when GasByGas grows its business and adds more clients or locations without affecting system performance. Because of its effective scalability, GasByGas can meet increasing demand without needing a

major overhaul or modifications to its infrastructure. As the user base grows, this adaptability guarantees the system's long-term sustainability.

The performance and load balancing capabilities of the client-server architecture are also important factors in the decision. The client (user interface) in this configuration is in charge of communicating with the user and displaying information, while the server manages the more laborious tasks, such as processing and storing data. The system optimizes response times and lessens the stress on individual user devices by allocating jobs between the client and the server. Customers will be able to place orders and track deliveries more quickly and effectively without their devices becoming overloaded.

Finally, the Client-Server paradigm makes effective communication possible. In order to ensure real-time communication and timely updates regarding gas supply, delivery schedules, and order statuses, the server may promptly transmit updates, notifications, and alerts to all clients (customers and outlet staff). By eliminating the need for manual procedures, this centralized communication minimizes delays and human error.

The Client-Server architecture is the ideal option for the Online Gas Requesting and Delivery Management System since it offers efficiency, scalability, security, and dependability. GasByGas can guarantee seamless and continuous service delivery by utilizing this model, which also allows it to grow and adjust to future expansion.

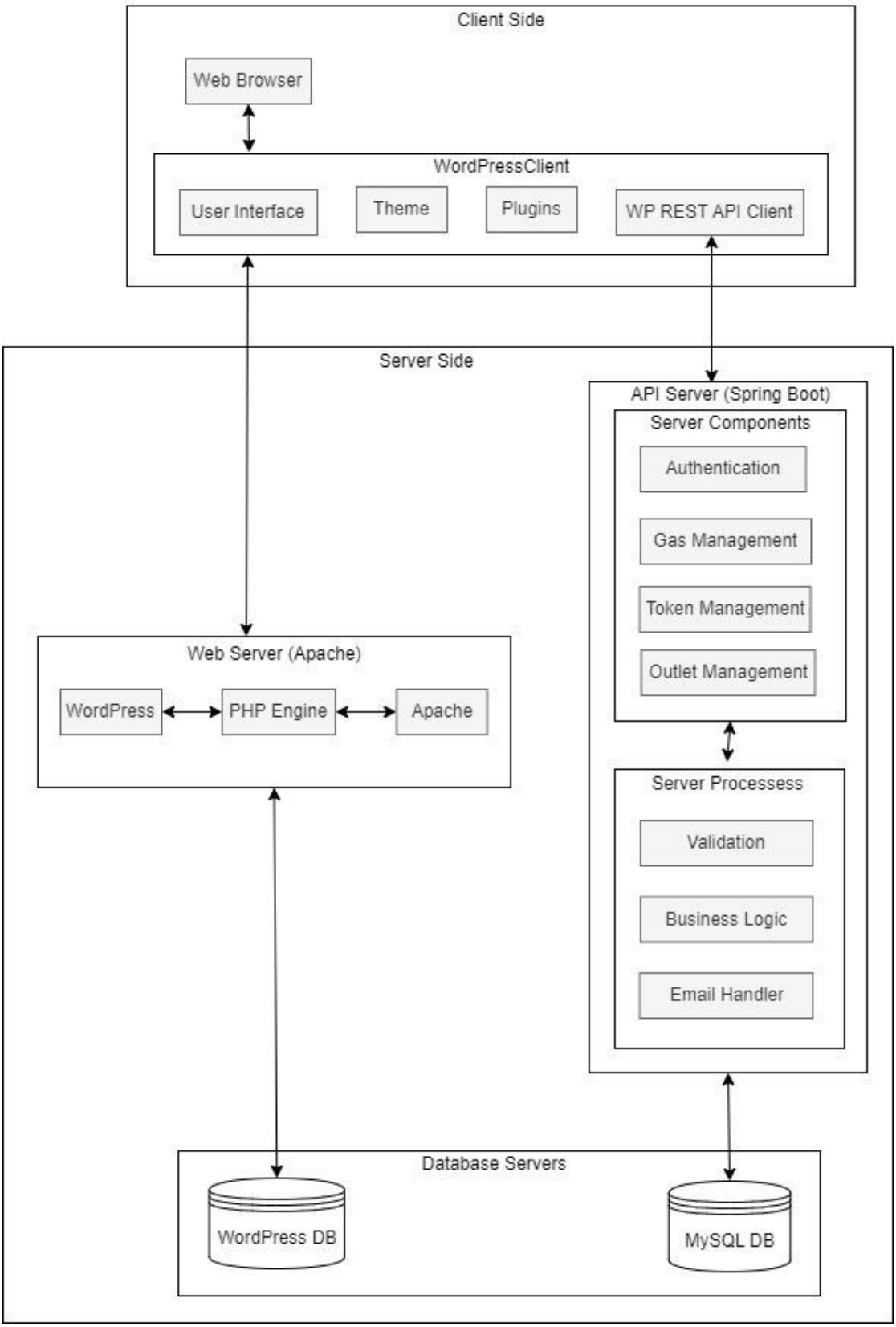


Figure 1_Client-Server architecture

The web application with client-side, server-side, and database servers is depicted in this system architecture diagram. On the client side, users engage with the system via a web browser that connects to a WordPress client, which consists of a WP REST API client, themes, plugins, and an interface. A Spring Boot API Server manages authentication, gas management, token management, and outlet management, while an Apache Web Server runs WordPress and handles PHP queries. Email processing, business logic, and validation are also handled by the API server. A MySQL database for business-related data and a WordPress database for content storage are both part of the database layer. The technology ensures effective data handling and operation by facilitating smooth connection between WordPress and the API server.

4.2 User Interface Designing

4.2.1 Data Input Designing

The Consumer Portal has a number of essential forms to improve the user experience. First, the Registration Form has fields for entering the consumer's name, NIC, email address, phone number, and address, among other important details. To make sure the data is entered accurately, each of these fields will have validation criteria (e.g., confirming the email format, checking the NIC validity, and verifying the phone number format). Tooltips explaining each field's needs will be provided to help users, particularly for fields like address and NIC. A consent to terms and conditions button will also be available for users to confirm that they accept the privacy and data usage policy.

Customers will be able to choose their preferred outlet, gas type, and amount from dropdown menus on the Gas Request Form. Additionally, a search function will be included in the form to help customers locate the closest store. If the customer is already registered, their address will automatically fill in to speed up the process and reduce entry time. In order to provide customers greater choice, the form will also let them select their desired delivery time or slot.

The outlet manager can manage scheduled deliveries, change the status of customer tokens, and enter available goods on the goods change Form included on the Outlet Manager Dashboard. Real-time data will be displayed on the form, along with alerts for low stock or impending deliveries, to improve the efficiency of stock management. If changes are required immediately, there will also be a manual override option.

Outlet managers can check the associated request data (gas type, amount, and customer details) and validate the token ID in the customer Verification section prior to confirming the delivery. The manager will be able to report any inconsistencies through the system, guaranteeing that only legitimate orders are fulfilled.

A delivery scheduling input mechanism will be available on the head office dashboard. In order to help arrange deliveries more effectively, this system will gather information such the outlet, petrol type, and quantity. It will also have a calendar interface. In order to avoid scheduling conflicts, the calendar will provide available timeslots. In order to handle urgent demands, the system will also enable the defining of delivery priority.

In order to assist the head office in creating comprehensive reports, the dashboard will have filters for outlets, date ranges, and request status. In order for the team to assess the system's performance, it will also have more sophisticated filtering choices, like payment status, delivery success rate, and customer feedback.

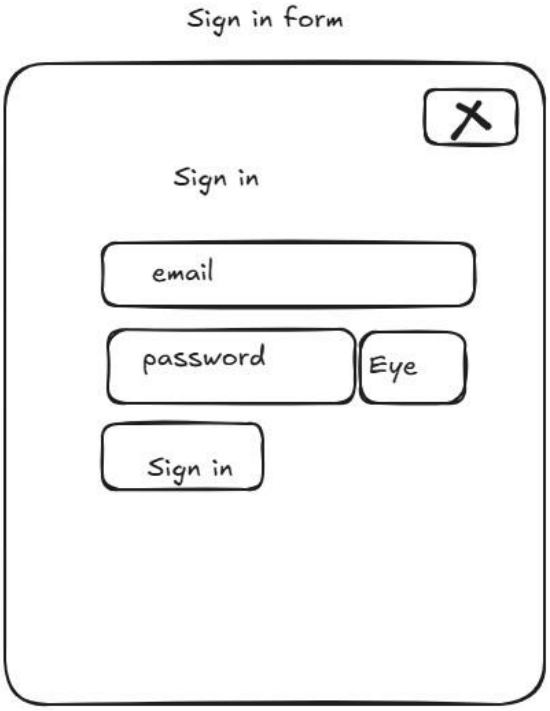


Figure 2_Sign in form

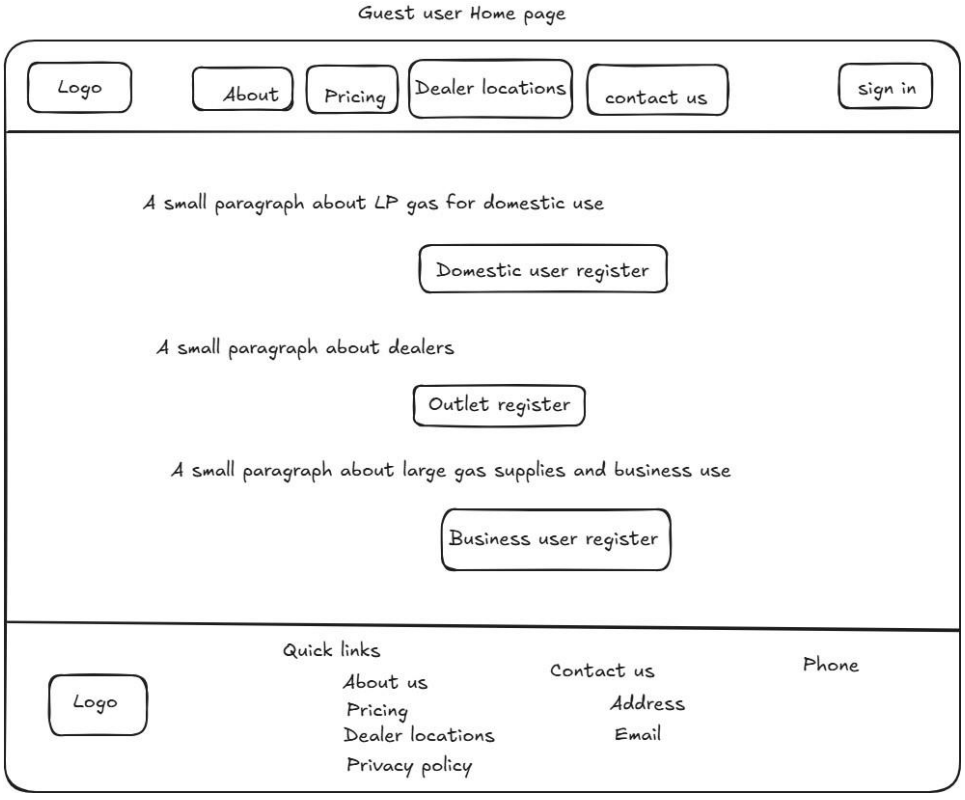


Figure 3_Guest user home page

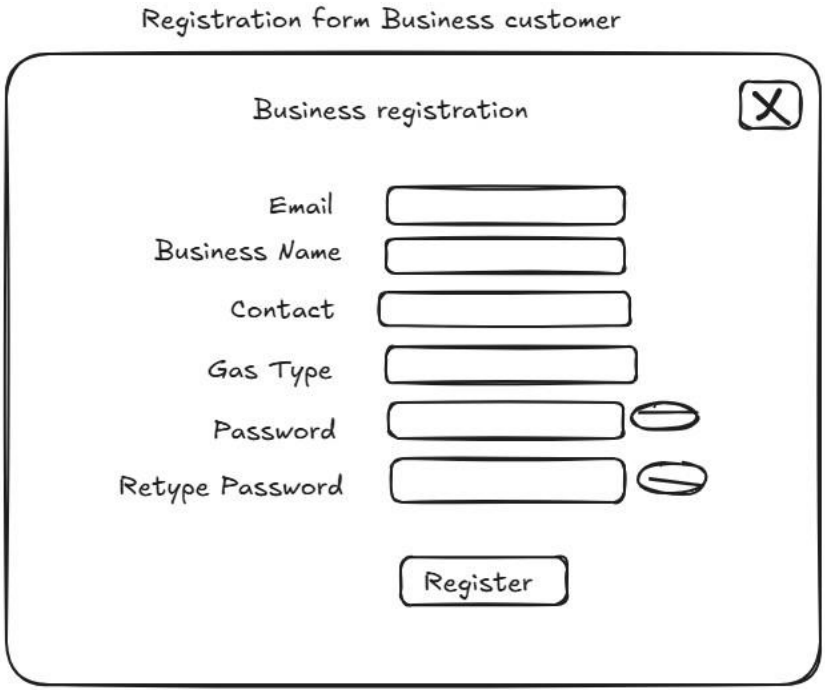


Figure 4_ Registration form business customer

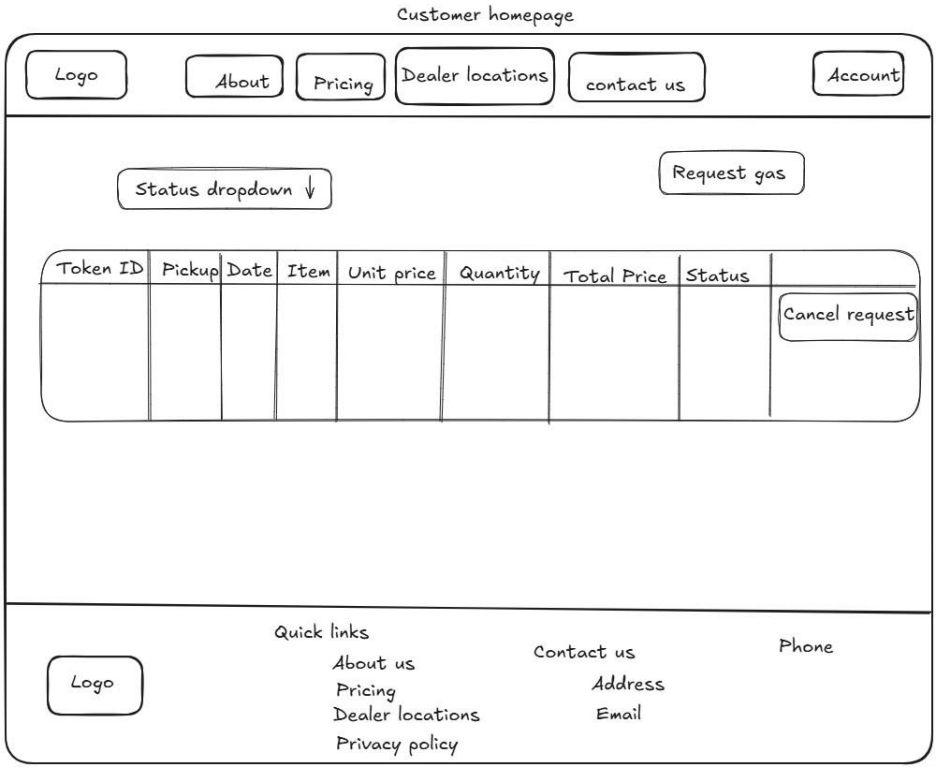


Figure 5_ Customer home page

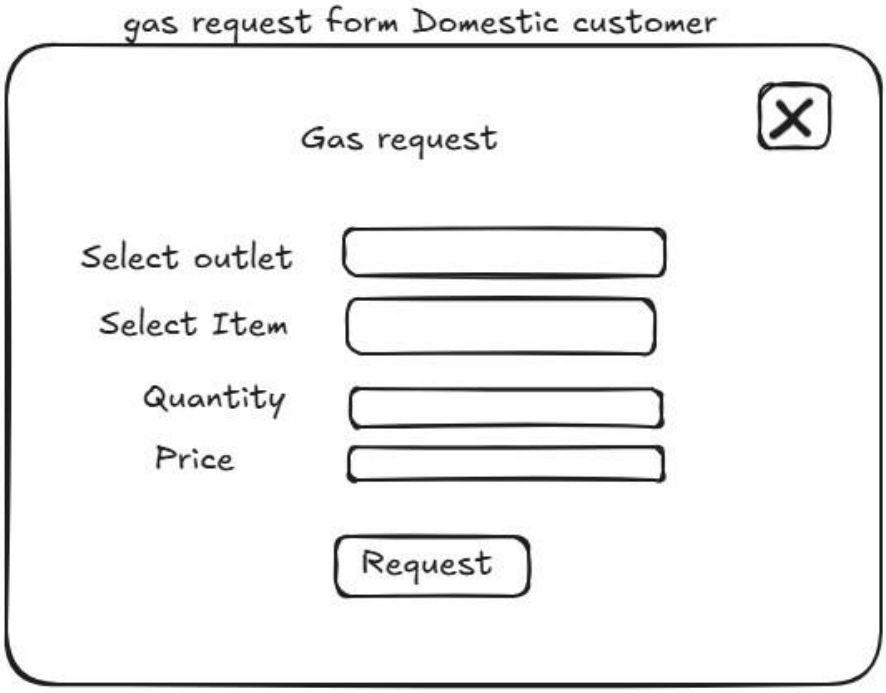


Figure 6_ Gas request form domestic customer

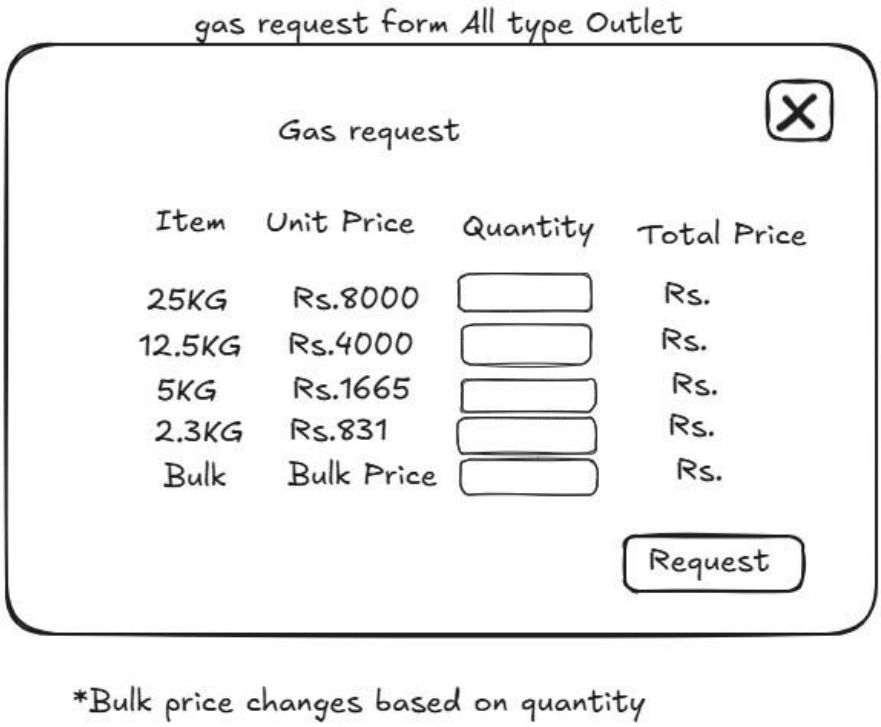


Figure 7_ Gas request form all type outlet

4.3 Output Designing

The output will be centered on giving customers fast and pertinent information about their needs. Important details including the token ID, pickup time, and delivery status will be shown in the Token Details. To help customers better understand when to expect their gas supply, an approximate delivery time will be offered. The status of the delivery will also be displayed using a delivery progress indicator, which will employ labels like In Progress, scheduled or delivered.

Customers will also be notified via email and SMS of a variety of events, including reallocation notices, delivery updates, and token confirmation. In addition to these automated messages, the system will notify users one day prior to the scheduled delivery to make sure they are ready. Push notifications will be delivered to offer real-time updates if the system has a mobile app.

The system will give outlet managers access to real-time stock status information, including each outlet's current petrol availability and other pertinent data. To make it easier to understand, graphs or charts will be used to visually show the stock levels. A list of active tokens will also be produced by the system, together with customer information like name, contact details, gas type desired, and other pertinent details. In addition to having the ability to reallocate tokens in the event of delivery problems, managers will be able to update token statuses such as Ready for pickup, delivery Scheduled straight from the dashboard.

The dashboard will give the head office a summary of the performance of the outlets, including measures like customer happiness, delivery success rates, and the effectiveness of stock management. Graphs or charts will be used to display this performance, making it simple to see patterns and problem areas. With statuses like scheduled, in progress, or completed, the delivery schedules will be shown in a centralised system. The central office will be able to supervise delivery for every store as a result. Along with a shortage signal to point out any problems, the dashboard will also provide a summary of the total stock levels across all shops. In order to predict future demand, predicted stock needs will also be computed using historical data.

Monthly Business Customer Analysis

Monthly Business Customer Analysis

Period: [Month Year]

1. Business Customer Overview

Total Active Business Customers: [Number]
New Registrations: [Number]
Pending Verifications: [Number]
Average Order Value: [Amount]

2. Consumption Analysis

Business Type	Total Orders	Average Order Size	Total Volume
[Type1]	[Number]	[Number]	[Number]
[Type2]	[Number]	[Number]	[Number]

Customer token receive email template

Dear [customer name]

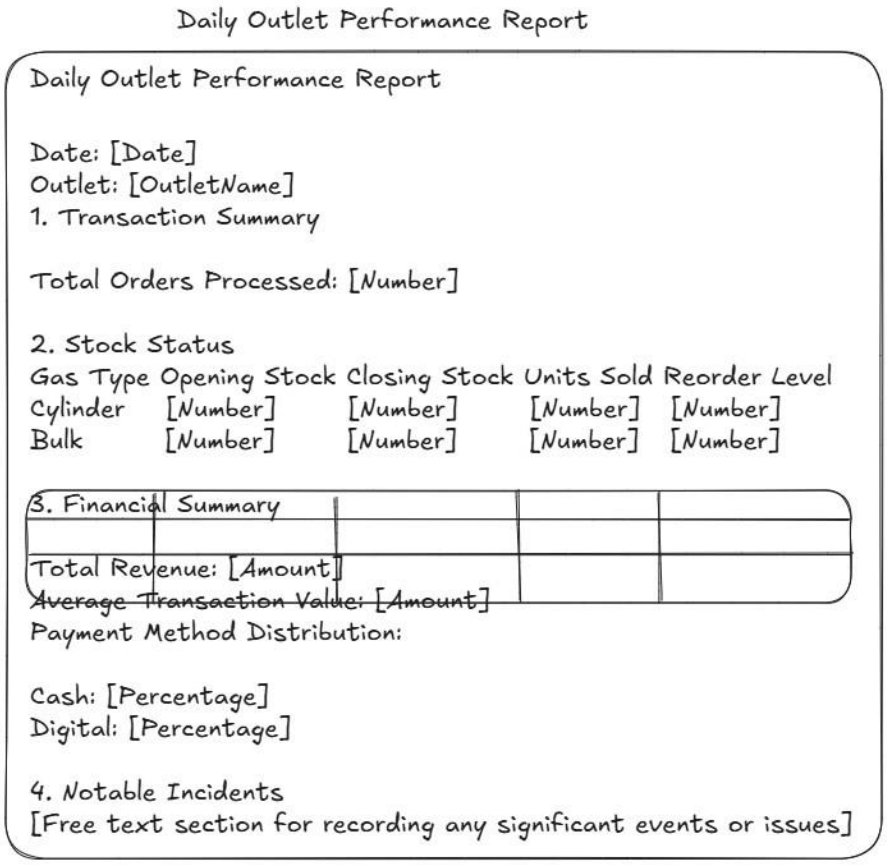
Your request recived by our GasbyGas online distribution system.

Your order number is [Order No]

Your token is [Token Id]

Your request pickup date is [Date]

This is a system generated email.



4.4 Identify the objects and data and file structures

The system's operations will be centred around a number of items. Features such as the consumer's name, NIC, email, phone number, address, and request history will be included in the Consumer object. Additional features that can be utilised to tailor interactions and enhance customer service include the date of account creation, loyalty status, and feedback ratings.

The Outlet object will have features such as the location, inventory, management information, delivery schedules, and contact data of the outlet. Data like customer happiness and delivery success rate will also be included, along with a rating system that allows customers to comment on services.

Request ID, customer ID, outlet ID, gas kind, quantity, token status, delivery status, and notifications are among the elements that will be included in the Gas Request object. To enable

improved administration and more individualised service, other features including request priority, payment status, and discount eligibility will also be added.

The gas type, quantity, delivery date, delivery status, and outlet ID will all be monitored by the Delivery Schedule object. Additional information will also be kept, including the delivery window (morning or afternoon), priority level, and delivery confirmation receipts.

Data structures-wise, a token will be kept in a key-value structure that connects the token ID to customer and request information for quick access. Delivery schedules will be managed according to priority by implementing a delivery queue utilising a priority queue. This will make it possible to process high-volume requests or urgent deliveries more quickly.

A framework for customer feedback will also be part of the system, and it will be connected to the customer ID and request history. This input will be used to evaluate the level of service quality and assist with platform improvement.

4.5 UML Diagrams

4.5.1 Use Case Diagram

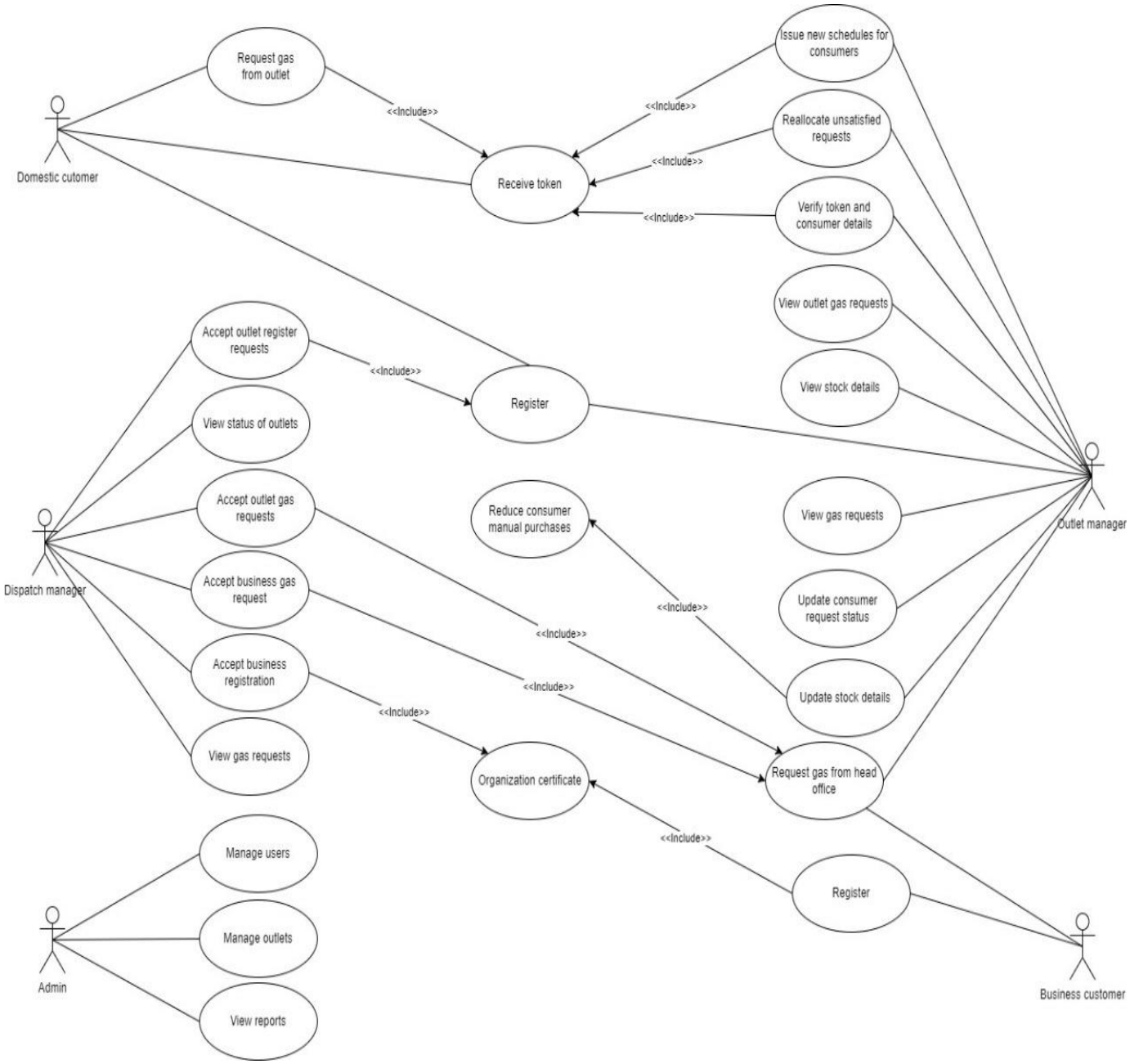


Figure 8_ Use Case Diagram

A gas management system is represented by this use case diagram, which shows how various user roles communicate with the system. With distinct roles, the main players are dispatch

managers, outlet managers, business customers, domestic customers, and administrators. Both business and domestic consumers can register and make gas requests; domestic users are given tokens that require scheduling and validation. By taking registrations, handling gas requests, and keeping track of outlet status, the dispatch manager manages gas distribution. By confirming tokens, updating stock information, and responding to customer requests, the outlet manager makes sure that gas allocation runs smoothly. The administrator oversees users, outlets, and reports in the interim. The diagram illustrates the ways in which different roles work together to enable effective gas management and distribution.

4.5.2 Activity Diagram

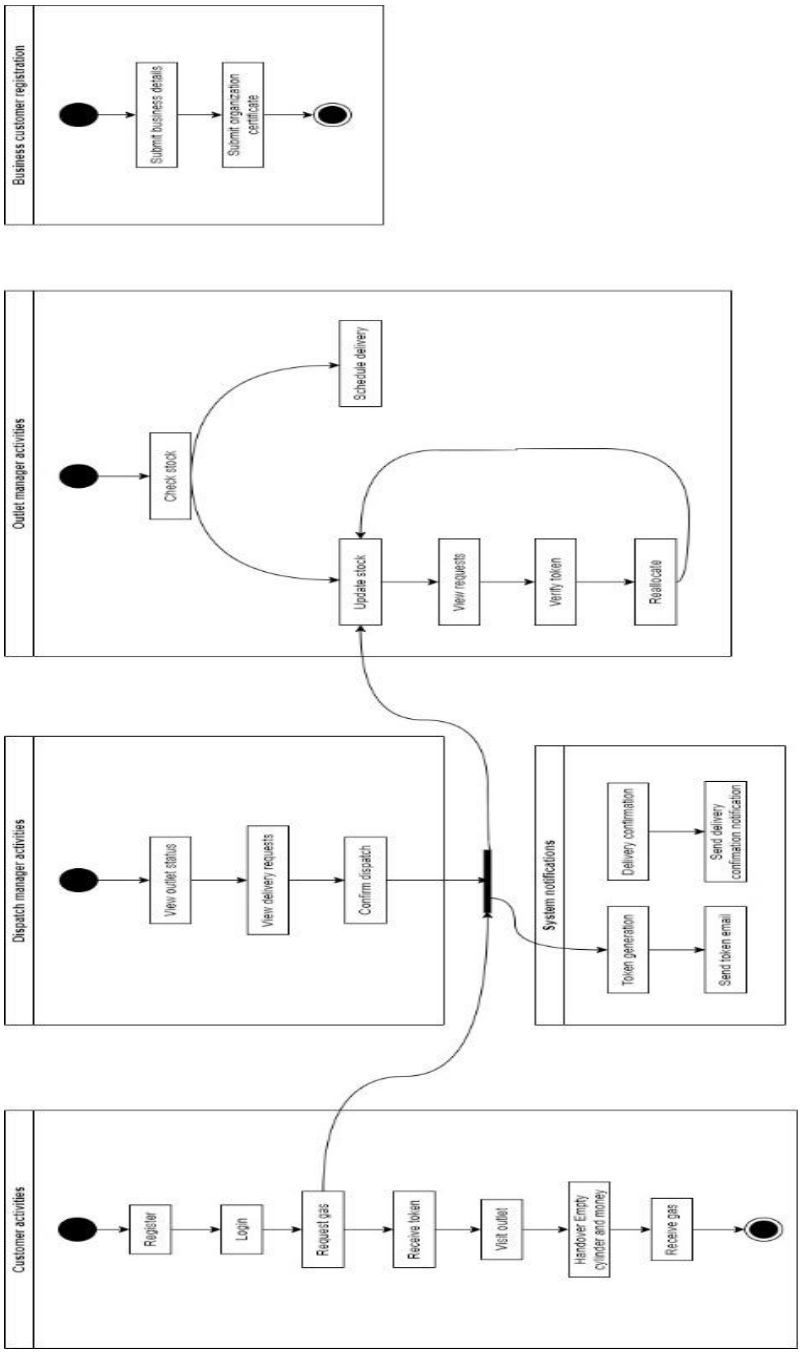


Figure 9_ Activity Diagram

The interactions between customers, dispatch managers, outlet managers, and business customers are depicted in this activity diagram, which illustrates the workflow of a gas distribution system. Consumers sign up, log in, make a gas request, get a token, go to the outlet, pay for the empty cylinder, and pick up the fresh gas. The dispatch manager views delivery

requests, verifies dispatches, and keeps an eye on outlet status. When necessary, the outlet manager reallocates stock, schedules deliveries, verifies tokens, and checks stock. Business clients sign up by sending in their information and credentials. Token creation, email delivery, and dispatch confirmations are managed by system notifications. Effective gas distribution, inventory control, and customer interactions are guaranteed by this methodical procedure.

4.5.3 Class Diagram

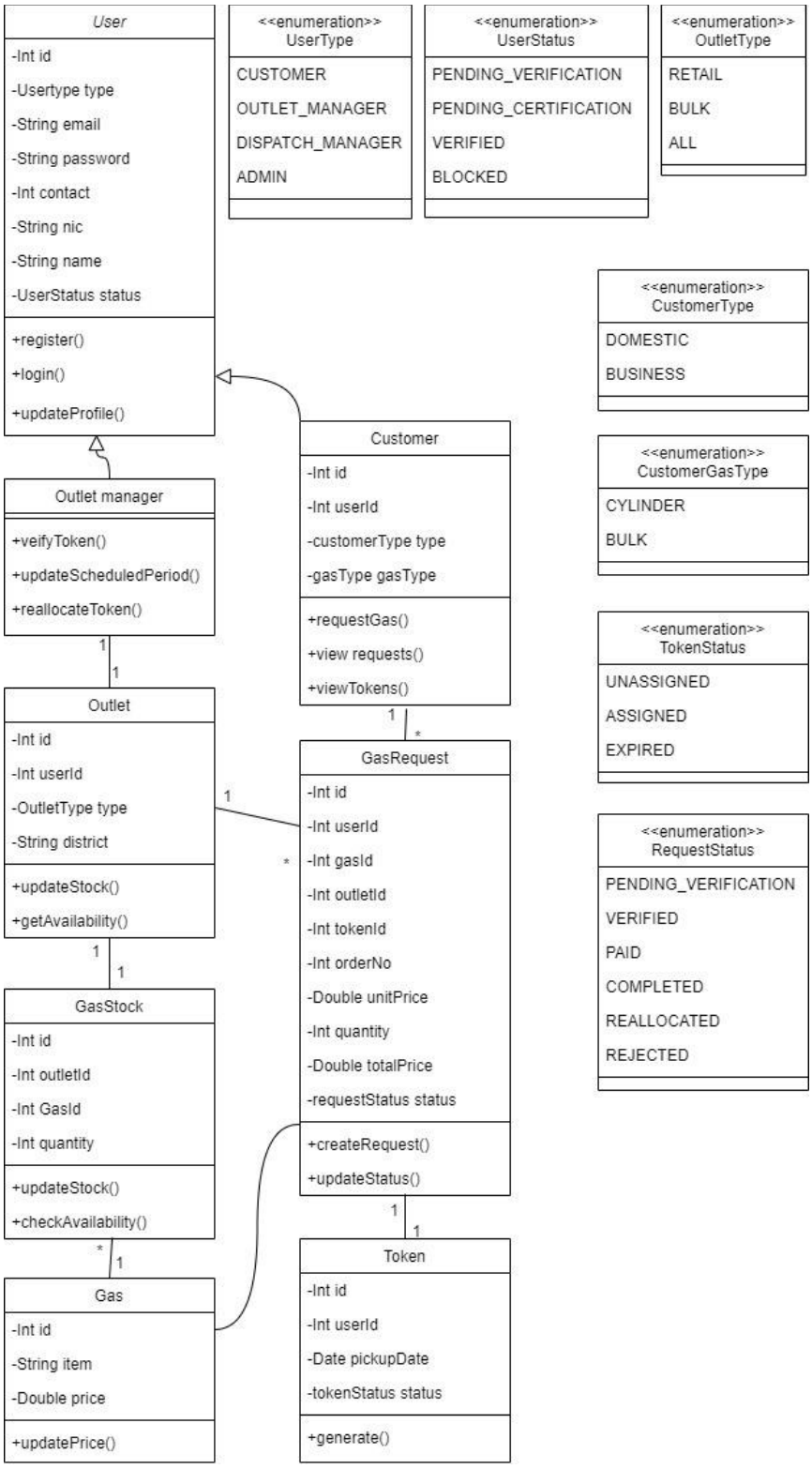


Figure 10_ Class Diagram

A gas management system is modeled by this class diagram, which displays entities, attributes, and how they interact. Roles with distinct functions, including Customer, Outlet Manager, Dispatch Manager, and Admin, are part of the system's core User class. The GasRequest class, which keeps track of information like price, quantity, and status, allows customers to request gas. Schedules for gas pickups are managed by the Token class, which makes sure that requests are fulfilled. While dispatch managers keep track of deliveries, outlet managers manage inventory, make schedule updates, and redistribute tokens. Admins are in charge of managing outlets and users. To ensure seamless operations, enumeration categories specify various user statuses, customer types, and request states. This structure improves efficiency and system reliability by streamlining customer service, stock control, and gas distribution.

4.5.4 Entity Relationship Diagram

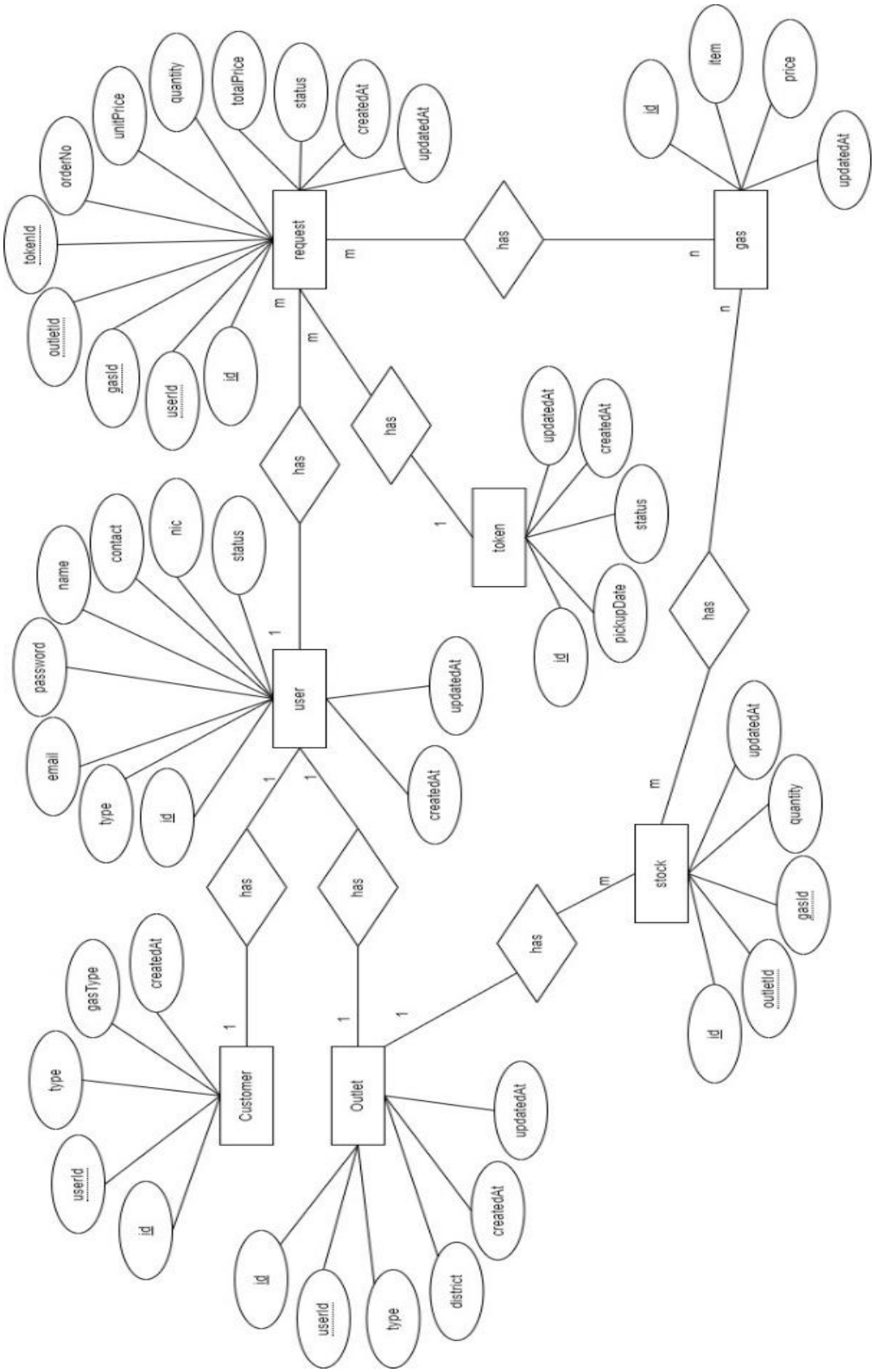


Figure 11_ Entity Relationship Diagram

The structure of a gas management system is depicted by this Entity-Relationship Diagram (ERD), which lists important entities along with their characteristics and connections. The main element is the User entity, which connects to Customers, Outlets, Requests, and Tokens. Tokens that store pickup information and status are used to track gas requests made by customers. Outlets oversee inventory, which includes various gas varieties with corresponding prices. Users are connected to gas transactions through the request entity, which records information about quantity, price per unit, total price, and status. The ERD guarantees effective gas distribution, tracking, and management within the system by charting these relationships.

Chapter 5

5.1 Software Tools and Technologies

5.1.1 Justification about System Tools and Technology

Frontend Development:

- Draw.io: Used for creating flowcharts
- Excalidraw: UI wireframes, and system architecture visuals to design frontend structures.
- Google Docs: Supports collaboration and documentation of frontend design requirements.

Backend Development:

- Postman: A tool for testing APIs to ensure seamless communication between frontend and backend systems.
- IntelliJ IDEA: A comprehensive IDE used for Java-based backend development.
- Java: The primary programming language for developing backend application logic.
- Spring Boot: A framework that simplifies the creation of APIs and backend microservices.

Database:

- XAMPP: Provides a local development environment, including MySQL, for database testing.
- MySQL: A relational database management system for storing and managing structured data.
- Flyway: A tool for managing and versioning database schema changes during development and deployment.

Notifications:

- Google Sheets: Used to organize and track notification systems and integrate basic alert automations.

Hosting and Deployment:

- GitHub: A platform for hosting repositories, managing code changes, and integrating with CI/CD pipelines for deployment.

Version Control:

- Git: Tracks code changes, enabling efficient collaboration and version management across development stages.

5.2 Implementation Environment

5.2.1 Hardware Environment

Local Development Machine:

- CPU: Multi-core processor (Intel i5/i7 or equivalent)
- RAM: Minimum 8 GB (16 GB recommended)
- Storage: SSD with at least 256 GB
- Operating System: Windows, macOS, or Linux

Web Server:

- CPU: Server-grade processor

- RAM: Minimum 2 GB
- Storage: SSD
- Software: Apache/Nginx with PHP and MySQL/MariaDB

Application Server:

- CPU: Server-grade processor
- RAM: 4–8 GB
- Storage: SSD
- Operating System: Ubuntu, CentOS, or Windows Server with Tomcat

Database Server:

- CPU: Moderate to high-performance server processor
- RAM: Minimum 4 GB
- Storage: SSD
- Operating System: Linux or Windows Server

Client Devices:

- Consumers: Smartphones or PCs with basic internet connectivity.
- Outlet Managers: Tablets or PCs with internet access.

5.2.2 Software Environment

Operating Systems

- For Development: Windows, macOS, or Linux.
- For Servers: Linux (Ubuntu/CentOS) for hosting or Windows Server, if required.

Web Hosting

- WordPress: Apache/Nginx with PHP and MySQL.
- Spring Boot: Standalone Java application or Docker containerized setup.

Databases

- MySQL: Used for application data storage.

Version Control

- Git: Used for version control with GitHub.

Development Tools

- IntelliJ IDEA: For Spring Boot back-end development.
- Postman: For API testing.
- Flyway: Manages database migrations.

5.2.3 Collaborative tools

For documentation: Microsoft Word

For task and project tracking: Google Sheets

For system architecture diagrams: Draw.io:

Communication: WhatsApp and google meeting for team collaboration.

5.3 Module Structure

User Management Module

- User registration, login, and profile management.
- Personal identity validation (NIC, phone, email).
- Notification management (SMS/email notifications).

Outlet Management Module

- Managing outlet details (location, available gas stocks).
- Handling gas availability checks at the outlets.
- Communicating stock availability and schedule changes.

Gas Request Module

- End consumers can request gas.
- Requests are not allowed if no stock is available at the outlet.

Delivery Schedule and Dispatch Module

- Confirms delivery schedules for outlets.
- Sends notifications (SMS/email) to token holders with delivery details.
- Reallocates deliveries if customers cannot meet the delivery conditions.

Notification System Module

- Sends SMS and email notifications to customers.
- Alerts for gas request status, payment confirmation, and delivery updates

5.4 Important Codes

Customer Registration

```
public User registerCustomer(CustomerRegistrationRequest request) {
    validateNewUser(request.getEmail(), request.getNic());

    User user = new User();
    user.setEmail(request.getEmail());
    user.setPassword(passwordEncoder.encode(request.getPassword()));
    user.setName(request.getName());
    user.setContact(request.getContact());
    user.setNic(request.getNic());
    user.setType(UserType.CUSTOMER);
    user.setStatus(UserStatus.PENDING_VERIFICATION);

    user = userRepository.save(user);

    Customer customer = new Customer();
    customer.setUser(user);
    customer.setType(request.getCustomerType());
    customer.setGasType(request.getGasType());
    customerRepository.save(customer);

    emailService.sendVerificationEmail(user.getEmail());
```

```
    return user;
  }
```

Login

```
public LoginResponse login(LoginRequest request) {
    User user = userRepository.findByEmail(request.getEmail())
        .orElseThrow(() -> new AuthenticationException("Invalid credentials"));

    if (!passwordEncoder.matches(request.getPassword(), user.getPassword())) {
        throw new AuthenticationException("Invalid credentials");
    }

    if (user.getStatus() != UserStatus.VERIFIED) {
        throw new AuthenticationException("Account not verified");
    }

    String token = jwtService.generateToken(user);

    return new LoginResponse(token, user.getType(), user.getName());
  }
```

Send Token Email

```
public void sendTokenEmail(String toEmail, Token token) {
```

```
        try {
            Template template = freemarkerConfig.getTemplate("token-email.ftl");
            Map<String, Object> model = new HashMap<>();
            model.put("tokenId", token.getId());
            model.put("pickupDate",
token.getPickupDate().format(DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm")));

            String content = FreeMarkerTemplateUtils.processTemplateIntoString(template,
model);

            MimeMessage message = mailSender.createMimeMessage();
            MimeMessageHelper helper = new MimeMessageHelper(message, true);

            helper.setFrom(fromEmail);
            helper.setTo(toEmail);
            helper.setSubject("Gas Pickup Token Details");
            helper.setText(content, true);

            mailSender.send(message);
        } catch (Exception e) {
            throw new EmailException("Failed to send token email", e);
        }
    }
```

Domestic Customer Request

```
public GasRequestResponse createDomesticRequest(GasRequestDTO request) {
```

```
        validateStockAvailability(request.getOutletId(), request.getGasId(),
request.getQuantity());

        GasRequest gasRequest = new GasRequest();
        BeanUtils.copyProperties(request, gasRequest);
        gasRequest.setStatus(RequestStatus.PENDING_VERIFICATION);

        gasRequest = gasRequestRepository.save(gasRequest);

        LocalDateTime pickupDate = calculatePickupDate(request.getOutletId());
        Token token = tokenService.generateToken(gasRequest.getId(), pickupDate);

        emailService.sendTokenEmail(gasRequest.getUser().getEmail(), token);

        return mapToResponse(gasRequest, token);
    }
```

Outlet Bulk Gas Request

```
public List<GasRequestResponse> createBulkOutletRequest(BulkGasRequestDTO request)
{
        String orderNo = generateOrderNo(request.getUserId());

        return request.getRequests().stream()
            .map(req -> {
```

```
            GasRequest gasRequest = new GasRequest();
            BeanUtils.copyProperties(req, gasRequest);
            gasRequest.setOrderNo(orderNo);
            gasRequest.setStatus(RequestStatus.PENDING_VERIFICATION);
            return gasRequestRepository.save(gasRequest);
        })
        .map(this::mapToResponse)
        .collect(Collectors.toList());
    }
```

Chapter 6

6.1 System Test Plan

6.1.1 Test Plan for

Version History

Version	Revised By	Summary	Approval Date
1.0	QA Team	Initial Test Plan	December 28, 2024

Table of Contents

Introduction

This Test Plan outlines the testing activities for the GasByGas Online Gas Requesting Delivering System. It defines the scope, approach, resources, and schedule of all testing activities to ensure the system meets its technical, functional, and business requirements.

Scope

In Scope

1. Consumer Web Application Testing

- Registration and authentication system
- Gas request management

- Token generation and management
- Notification system (Email)

2. Outlet Management System

- Stock management
- Token verification
- Customer request handling
- Delivery scheduling

3. Dispatch Management System

- Distribution management
- Outlet status monitoring
- System administration

4. Admin System

- Business customer validation
- User management
- Role and permission management
- System configuration
- Reports generation

Out of Scope

- Minor edge cases
- Physical delivery process testing
- Email service provider internal systems

Test Objectives

1. Verify user registration and authentication

- Prevent multiple registrations per identity
- Validate NIC/Phone/Email uniqueness
- Test business organization registration process

2. Validate gas request system

- Token generation and management
- Schedule management
- Stock availability checks
- Request limitations per user

3. Test notification system

- Delivery schedule notifications
- Payment reminders
- Token status updates
- Request status updates

4. Verify outlet management functions

- Token verification
- Stock management
- Customer request processing
- Delivery scheduling

5. Test head office monitoring system

- Outlet status tracking
- Distribution management

6. Verify admin functionality

- Business customer validation User management capabilities
- System configuration management
- Report generation functionality

Detailed Test Approach

1. Requirement Analysis

- Review and validate all functional requirements
- Identify derived requirements
- Validate business rules and workflows

2. Functionality Testing

A. Registration and Authentication

- User registration validation
- Business customer registration
- Identity verification
- Login/logout functionality
- Password management

B. Gas Request Management

- Token generation
- Schedule management
- Stock availability checks
- Request limitation validation

C. Notification System

- Email notifications
- Delivery updates
- Payment reminders

D. Outlet Management

- Stock tracking
- Token verification
- Customer management
- Delivery scheduling

E. Database Testing

- Data integrity
- CRUD operations
- Data validation

F. Admin System Testing

- User CRUD operations
- System configuration changes
- Report generation and export

3. Integration Testing

- Web server and database integration
- Client application and server communication
- Email service integration

4. Security Testing

- Authentication

- Authorization
- Data encryption
- Session management
- Input validation

5. Performance Testing

- Load testing
- Stress testing
- Response time

6. Usability Testing

- Navigation
- User interface
- Error messages

7. Compatibility Testing

- Browser compatibility

Test Strategy

- Implement both manual and automated testing approaches
- Follow bottom-up integration testing
- Conduct regular regression testing

Test Schedule

Phase	Duration	Start Date	End Date
Requirement Analysis	4 days	December 29, 2024	January 1, 2025

Test Planning	2 days	January 2, 2025	January 3, 2025
Test Case Development	4 days	January 4, 2025	January 7, 2025
Functional Testing	6 days	January 8, 2025	January 13, 2025
Integration Testing	4 days	January 14, 2025	January 17, 2025
Security Testing	2 days	January 18, 2025	January 19, 2025
Performance Testing	2 days	January 20, 2025	January 21, 2025
UAT	4 days	January 22, 2025	January 25, 2025
Bug Fixes and Retesting	4 days	January 26, 2025	January 29, 2025

Problem Severity Classification

1. Severity 1 (Critical)

- System crash
- Data loss
- Security breaches
- Complete feature failure

2. Severity 2 (High)

- Major functionality issues
- Incorrect token generation
- Failed notifications
- Payment processing errors

3. Severity 3 (Medium)

- UI/UX issues
- Minor functionality issues
- Performance degradation
- Non-critical validation errors

4. Severity 4 (Low)

- Cosmetic issues

- Minor UI inconsistencies
- Documentation errors
- Enhancement requests

Test Resources

1. Testing Team

- 1 Test Lead
- 3 QA Engineers

2. Environment

- Test Environment Setup
- Testing Tools and Licenses
- Test Data

Pass/Fail Criteria

Pass Criteria:

- All critical and high-priority test cases pass
- No severity 1 or 2 bugs remain open
- 95% of all test cases pass
- All security vulnerabilities addressed
- Performance metrics meet requirements

Fail Criteria:

- Critical functionality failures
- Unresolved security issues
- Data integrity issues
- Performance below acceptable threshold
- Multiple severity 1 or 2 bugs

Environment

1. Testing Environments

- Development
- QA
- Staging
- Production

2. Infrastructure Requirements

- Web Servers
- Database Servers
- Email Gateway
- Load Balancers

Tools And Defect Tracking

- Test Management: Google sheets
- Defect Tracking: Google sheets
- Automation: Selenium WebDriver
- Performance Testing: Google chrome dev tools/Postman
- API Testing: Postman
- Version Control: Git

Exit Criteria

1. All test cases executed
2. No critical or high-severity defects open
3. All planned features tested
4. Performance criteria met
5. Security requirements satisfied
6. UAT sign-off received

7. All documentation completed

6.2 Test case and Test results

User registration validation

Test ID	Description	Priority	Pre requisties	Steps	Input data	Expected output	Actual output	Pass/Fail	Comment
1	Domestic customer registration	High	Domestic customer account	1. Click Domestic user register button		User should navigate to customer dashboard	As expected	Pass	
				2. Input email	gayalvtcdiv@gmail.com				
				3. Input customer name	Ravindu				
				4. Input NIC	992020245V				
				5. Input	710175988				

				Conatct					
				6. Input Password	Password1!				
				7. Input retype password	Password1!				
				8. Click Register button					

Test ID	**Description**	**Priority**	**Pre requisties**	**Steps**	**Input data**	**Expected output**	**Actual output**	**Pass/Fail**	**Comment**
2	Business customer registration	High	Business customer account	1. Click Business user register button		**User should navigate to customer**	**As expected**	**Pass**	

				2. Input email	gayalvtcdiv@gmail.com	**dashboard**			
				3. Input business name	Test Resturant				
				4. Input Contact	710175988				
				5. Select Gas Type	Cylinder				
				6. Input Password	Password1!				
				7. Input retype password	Password1!				
				8. Click Register button					

Test ID	Description	Priority	Pre requisties	Steps	Input data	Expected output	Actual output	Pass/Fail	Comment
3	Outlet registration	High	Outlet account	1. Click outlet register button		User should navigate to outlet dashboard	As expected	Pass	
				2. Input Email	gayalvtcdiv@gmail.com				
				3. Input Outlet name	Test outlet				
				4. Input district	Badulla				
				5. Input Contact	710175988				
				6. Input Outlet type	Retail				

				7. Input Password	Password1!				
				8. Input Retype password	Password1!				
				9. Click Register button					

Login/logout functionality

Test ID	Description	Priority	Pre requisties	Steps	Input data	Expected output	Actual output	Pass/Fail	Comment
1	Validate valid email with valid password	High	Valid email with valid password	1. Enter email	gayalvtcdiv@gmail.com	Login without errors	As expected	Pass	
				2. Enter password	Pasword1!				

				3. Click login button					

Test ID	Description	Priority	Pre requisties	Steps	Input data	Expected output	Actual output	Pass/Fail	Comment
2	Validate domestic customer sign in	High	Domestic customer account	1. Enter email	gayalvtcdiv@gmail.com	Login without errors	As expected	Pass	
				2. Enter password	Password1!				
				3. Click login button					

Test ID	Description	Priority	Pre requisties	Steps	Input data	Expected output	Actual output	Pass/Fail	Comment

3	Validate business customer sign in	High	Business customer account	1. Enter email	gayalvtcdiv@gmail.com	Login without errors	As expected	Pass	
				2. Enter password	Pasword1!				
				3. Click login button					
				2. Enter password	Gasbygasdm1!				
				3. Click login button					

Test ID	**Description**	**Priority**	**Pre requisties**	**Steps**	**Input data**	**Expected output**	**Actual output**	**Pass/Fail**	**Comment**
4	Validate outlet manager sign in	High	Outlet account	1. Enter email	gayalvtcdiv@gmail.com	Login without errors	As expected	Pass	
				2. Enter password	Password1!				
				3. Click login button					

Test ID	Description	Priority	Pre requisties	Steps	Input data	Expected output	Actual output	Pass/Fail	Comment
5	Validate dispatch manager sign in	High	Dispatch account	1. Enter email	dispatch@manager.com	Login without errors	As expected	Pass	
				2. Enter password	Gasbygasdm1!				
				3. Click login button					

Test ID	Description	Priority	Pre requisties	Steps	Input data	Expected output	Actual output	Pass/Fail	Comment
6	Validate admin sign in	High	Admin account	1. Enter email	admin@admin.com	Login without errors	As expected	Pass	
				2. Enter password	Adminadmin1!				
				3. Click login button					

Test ID	Description	Priority	Pre requisties	Steps	Input data	Expected output	Actual output	Pass/Fail	Comment
17	Sign out	Low	Test ID 1	Click sign out button		User should sign out and go to home page	As expected	Pass	

Chapter 7

7.1 Conclusion

For GasByGas, this project created an online gas ordering and delivery system with the goal of increasing gas distribution efficiency. The solution improves communication via email notifications, guarantees on-time delivery, and automates the gas request procedure. The platform can be scaled and adjusted to meet different company demands by managing domestic and industrial requests independently.

Although the system has shown itself to be effective, it might be further improved with improved payment integration and real-time delivery tracking. In order to further enhance the quality of services, future advancements might concentrate on automating customer service and feedback collection. both customers and the company gain from the GasByGas system's dependable and easy-to-use solution for expediting gas requests and delivery.

References

Tanenbaum, A. S. and Van Steen, M. (2017) Distributed systems: Principles and paradigms. 3rd edn. Pearson.

Appendix

1. Group Contribution

Group Member	Contribution
Ravindu	Chapter 4, 6 Outlet dashboard, Admin dashboard, Outlet Gas request forms
Sachini	Chapter 3, 7 Domestic customer signup, Domestic customer dashboard, Domestic customer gas request
Ahinsala	Chapter 1, 4 Sign in form, Guest user homepage, Dispatch dashboard, Outlet signup
Dewumini	Chapter 2, 5 Business customer signup, Business customer dashboard, Business customer gas request

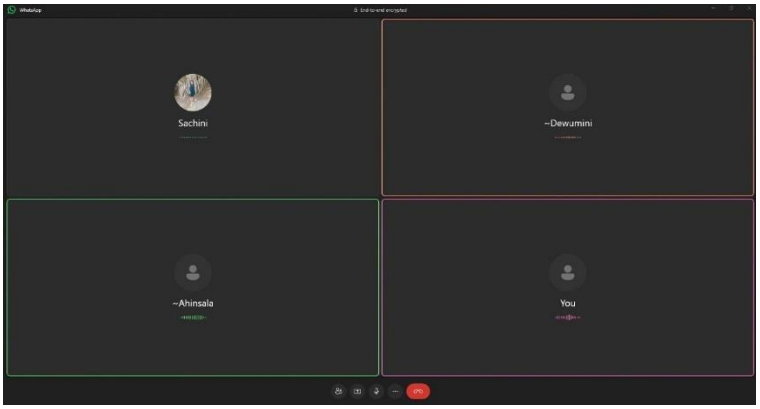


Figure 12_ WhatsApp meeting

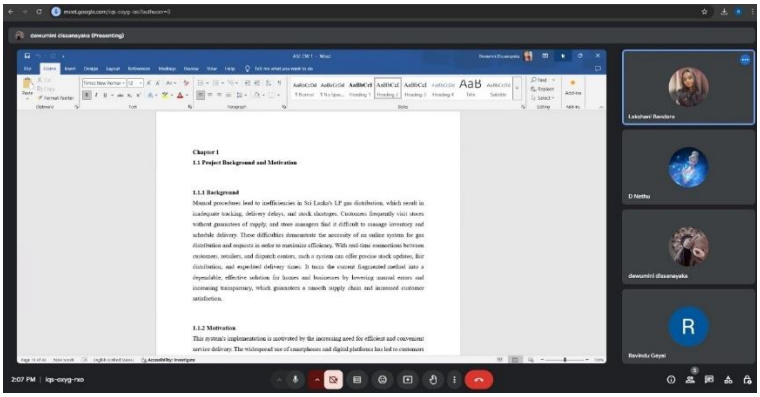


Figure 13_ Google meeting