Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Project Report

on

"Queue-Based Banking Management System"

[Code No: COMP 202]

For partial fulfillment of Second Year/first Semester in Computer Engineering

Submitted By:

Sanat Giri [17]

Submitted To:

Mr. Sagar Acharya

Department of Computer Science and Engineering

Submission Date: 2026-02-18

Bona fide Certificate

This is to certify that the project entitled

" Queue-Based Banking Management System"

is a bonafide work carried out by the student in partial fulfillment of the requirements for the Second Year /First Semester of the Bachelor of Engineering in Computer Engineering, Department of Computer Science and Engineering.

Name of Student :

Sanat Giri

The project work has been carried out with sincerity, dedication, and to the satisfaction of the department, adhering to the academic guidelines.

Project Evaluator

_____

Sagar Acharya

Lecturer

Department of Computer Science and Engineering

Date: 2026-02-18

Acknowledgement

I would like to express my sincere gratitude to my teacher, **Mr. Sagar Acharya**, for his wonderful introduction to Data Structures and Algorithms. His teaching inspired me to build this project and helped me understand the core concepts that made this project possible.

I am also grateful to the Department of Computer Science and Engineering for providing me the platform to convert my ideas into a practical project.

Thank You

Abstract

Customer service management is one of the most crucial considerations in any banking situation. This project outlines the development and implementation of a Queue-Based Banking Management System in C++ that simulates real-world banking situations using Data Structures and Algorithms.

The system employs a linked list queue to manage customer service requests on a First-In-First-Out (FIFO) basis, ensuring all customers receive equal and efficient service. The key operations include account opening, deposit, withdrawal, transaction history, and safe admin login with password protection. The system is developed in C++ with the Qt framework, offering a professional graphical user interface for convenience.

This project aptly demonstrates the usage of queue data structures, dynamic memory allocation, and modular programming techniques based on object-oriented programming in C++. The system efficiently automates customer queue management and banking operations using an easy-to-use Qt-based GUI with admin login authentication.

Keywords: Queue, Data Structures, Banking System, C++, Qt Framework, Linked List, FIFO, File Handling, GUI.

Table of Contents

List of Figures

Acronyms/Abbreviations

The list of all abbreviations used in the documentation are as follows :

DSA– Data Structures and Algorithms

FIFO – First In First Out

OOP – Object Oriented Programming

GUI – Graphical User Interface

IDE –Integrated Development Environment

QT– Qt Framework

Chapter 1 Introduction

1.1 Background

Data Structures and Algorithms are the constituents of effective software design. Among the most basic is the queue, which functions on the First-In-First-Out (FIFO) concept and finds numerous applications in practical implementations such as banking, where customers are served in the order they arrive.

The traditional banking system faces numerous challenges in efficiently handling customer services, leading to delays and mistakes. The manual process of handling customer queues and transactions is time-consuming and prone to mistakes, thus establishing a definite need for an automated system that can efficiently handle customer queues and process transactions accurately.

The proposed project seeks to create a Queue-Based Banking Management System using C++ and the Qt Framework to automate the handling of customer queues using a linked list-based queue, handle basic banking operations such as deposits and withdrawals, and record transaction history using file handling operations, all of which are to be done from a secure admin-controlled graphical user interface.

1.2 Objectives

The primary objectives of " **Queue-Based Banking Management System**" are :

- Design and develop an efficient queue using a linked list to handle customer service requests based on the FIFO principle.

- Develop basic banking operations such as account opening, deposit, withdrawal, and account closure.

- Implement a secure admin login system with password management to manage access to the banking system.

- Design and implement file handling to permanently store and retrieve account and transaction information.

- Develop a professional and user-friendly Graphical User Interface using the Qt Framework for ease of use.

1.3 Motivation and Significance

The reason for pursuing this project is based on the need to apply the concepts of Data Structures and Algorithms taught in class on a real-world application. Banking systems are a traditional example of queue-based processes, where customers are served on a first-come, first-served basis, making it an ideal example for the application of the FIFO principle.

The current manual banking system is inefficient, error-prone, and difficult to handle. This project will overcome these drawbacks by developing an automated queue-based system for efficiently handling customer service requests, performing core banking operations correctly, and maintaining permanent transaction records through file handling methods.

The importance of this project is based on its ability to show the relevance of basic DSA concepts such as linked list-based queues, dynamic memory allocation, and modular object-oriented programming in C++. Moreover, the addition of the Qt Framework to the project makes it more than a terminal-based application, as it provides a professional graphical user interface with admin authentication, which is much closer to a real-world banking application.

Chapter 2 Related Works

There have been a few projects and researches that have tried to implement data structures in banking systems. Queue-based systems have been widely used in customer service simulation models, where the efficiency of the FIFO algorithm has been demonstrated in managing waiting lines.

The previous implementations of banking systems in C and C++ have been focused on simple account management using arrays and linked lists. However, there have been few attempts that have tried to integrate customer queue management with file handling. Even fewer have tried to integrate a graphical user interface.

This project is an improvement over the previous ones because it tries to integrate a linked list-based queue with the basic banking operations, transaction history management, and admin authentication, all of which are provided through a professional-looking Qt-based graphical user interface.

Chapter 3 Design and Implementation

3.1 Implementation

The system was designed with a modular approach using C++ to make it maintainable and have a clear division of labor. The project is divided into separate modules for queue management, account management, and transaction management, with separate header and source files for each.

The system consists of four major components that communicate with each other: the queue module, which handles customer service queries using a linked list-based FIFO queue; the account module, which handles banking transactions; the transaction module, which logs and displays transaction history; and the main window module, which offers the graphical user interface implemented using the Qt Framework.

The GUI is implemented using Qt Widgets, offering a professional and user-friendly interface with admin login security, password management, and navigation to different banking modules via a stacked widget layout. Data is stored using file handling, ensuring that records are retained after closure.

3.1.1 Implementation Planning

The original plan was to develop the project as a simple array-based banking system. Later, the plan was changed to include the development of a linked list-based queue system to more accurately model real-world banking operations. The scope was further increased to include the development of a graphical user interface with the Qt Framework, in place of the terminal-based interface for a more professional and useful approach.

The planning phase included identifying the goals of the system, choosing suitable data structures, and identifying requirements for fundamental functionality of the system, which include customer queue handling, account handling, transaction handling, and secure admin login with password handling.

The system was designed in a modular way, with separate modules for handling queues, accounts, transactions, and GUI handling, to keep the code organized and provide a clear illustration of DSA concepts in C++.

3.1.2 Requirement Analysis

The system requirements were identified to ensure proper functionality and performance.

Functional requirements include the management of the customer queue based on the FIFO principle, account deletion and creation, deposit and withdrawal transactions, the storage of transaction history, the secure login of the admin using password management, and the saving and loading of data using file handling.

Non-functional requirements include system reliability, proper memory management, modular programming, fast response time, and usability with a clean and intuitive graphical user interface built using the Qt Framework.

3.1.3 System Design

The system was designed with a modular approach to ensure that it is maintainable, scalable, and has a clear separation of responsibilities. The system was broken down into four major modules: queue management, account management, transaction processing, and GUI management, each implemented in its own header and source files.

The system is object-oriented in design, where the queue module is responsible for handling customer service requests using a linked list-based FIFO data structure, the account module is responsible for handling all banking transactions, the transaction module is responsible for recording and displaying transaction history, and the GUI module is responsible for implementing the graphical user interface using Qt Widgets. All modules communicate with each other through the main window class, which acts as the central controller of the system.

The system has a graphical user interface that uses a stacked widget layout to enable smooth transitions between different pages such as the login page, main menu, queue management, account management, deposit and withdrawal, transaction history, and password management, all of which are managed solely by the admin.

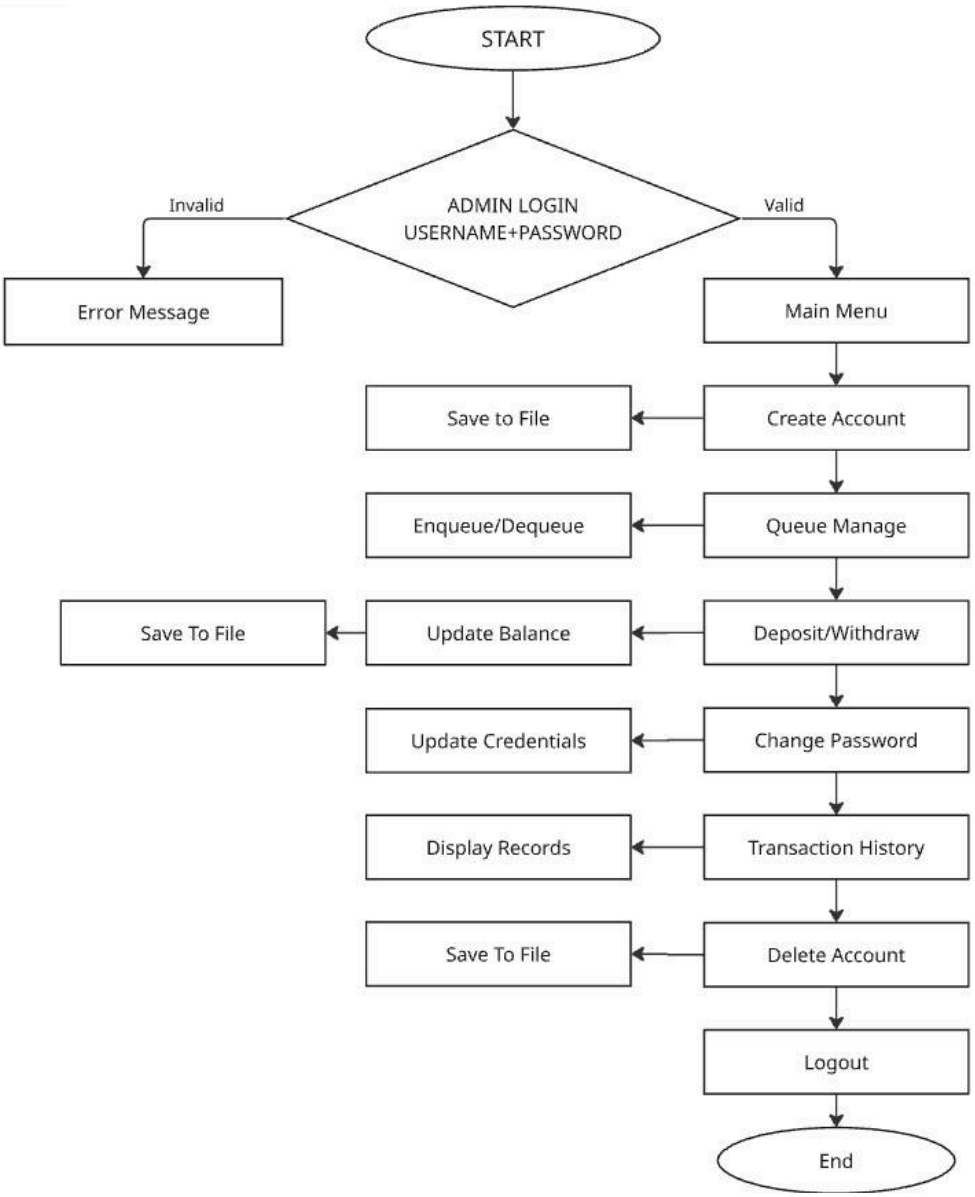

Fig: 3.1 : System Flow Diagram

The system loads existing data from files and presents the admin login screen. Upon successful authentication, the admin can create accounts, manage the customer queue, perform deposits and withdrawals, view transaction history, delete accounts, and change the admin password. All changes are automatically saved to files after each operation.

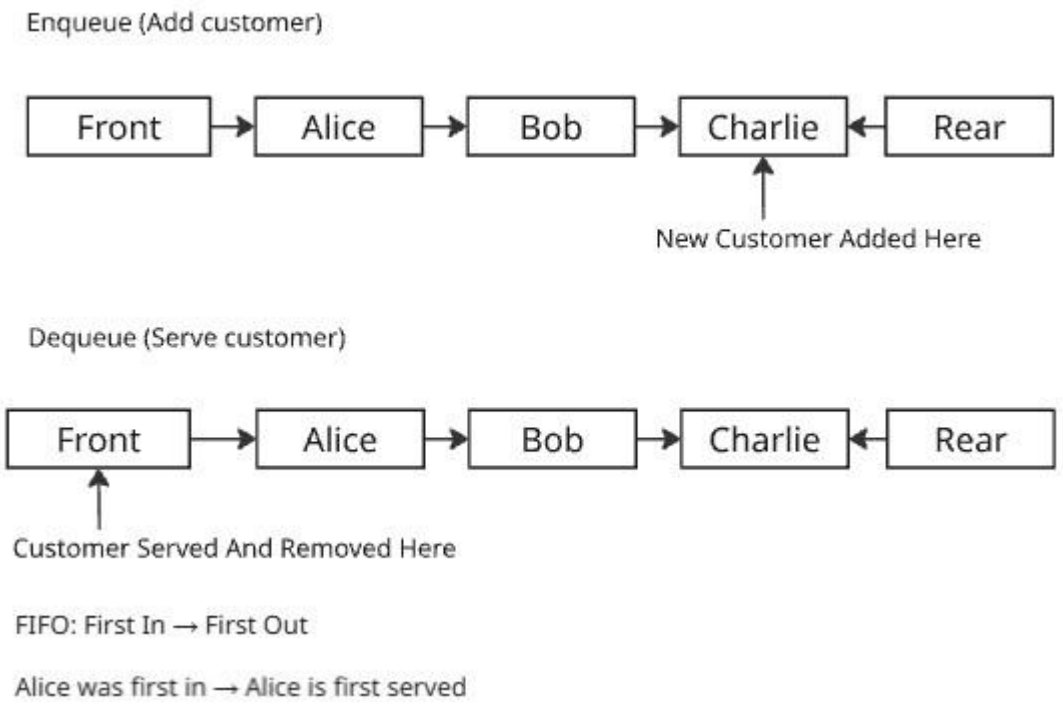

Fig: 3.2 : Linked List Based Queue Operations

The above diagram illustrates the queue operations used in the system. In the enqueue operation, new customers are added at the rear of the queue. In the dequeue operation, customers are served and removed from the front. This ensures that the first customer to arrive is the first to be served, following the FIFO principle.

3.1.4 Development

The initial plan was to develop a terminal-based banking system using C++. Later, the plan was expanded to include a GUI-based system using the Qt Framework. This was done to make the system more useful and professional. The development was done individually, with the aim of developing the basic concepts of DSA, along with a functional GUI.

During the course of development, some technical issues arose. These included issues related to the integration of the Qt widget, handling the cin buffer in the terminal-based system, and combining multiple modules. These issues were overcome by performing rigorous debugging and problem-solving.

The main development stages of the project were as follows:

- Requirement Planning and Gathering
- System Design and Module Planning
- Implementation of DSA (Queue, Account, Transaction)
- Qt GUI Development and Integration
- Implementation of Admin Authentication
- File Handling and Data Persistence
- Testing and Debugging

After completing the planning stage, a number of design diagrams were developed. These diagrams represented various aspects of the system. These included a system flow diagram and a queue operation diagram. These diagrams were developed to explain the functioning of the system.

3.1.5 Testing and Debugging

The Queue-Based Banking Management System was tested extensively for functionality, reliability, and accuracy. Functional testing ensured that all modules of the system, such as queue management, account opening, deposit and withdrawal transactions, transaction history, and admin login, were functioning properly

.Integration testing ensured that there were no glitches in the interaction between the queue module, account module, transaction module, and GUI module, and that data was properly stored and retrieved using file handling techniques. Performance testing was done to check the system's response time during multiple queue operations and transactions, and the system functioned properly.

Debugging was done to remove errors in infinite loop programming in cin buffer handling, Qt widget integration, and module linking errors in multiple modules. Error handling techniques were incorporated into the system to ensure that it remains stable, such as checking for insufficient funds, empty queue messages, invalid inputs, and incorrect login details.

3.2 System Requirement Specifications

The system was developed and tested with specific software and hardware requirements to ensure proper functionality, performance, and compatibility. These requirements are outlined as follows.

3.2.1 Software Specifications

The software components necessary for developing and operating the system include:

- Programming Language: C++
- Framework: Qt Framework (Qt Widgets)
- Integrated Development Environment (IDE): Qt Creator
- Compiler: G++ (GCC)
- Version Control: Git / GitHub

These software components provide the foundation for implementing the DSA logic, graphical user interface, file handling, and modular programming structure.

3.2.2 Hardware Specifications

The hardware requirements for running the system effectively are as follows:

- Processor: Intel Core i3 or equivalent (minimum); Intel Core i5 or higher recommended
- Memory (RAM): 4 GB minimum; 8 GB or more recommended
- Storage: Minimum 1 GB free space for Qt installation and project files
- Display: 1366×768 resolution minimum; 1920×1080 recommended for better GUI experience
- Input Devices: Standard keyboard and mouse

Chapter 4 Results and Discussion

4.1 Project Overview

The project is a Queue-Based Banking Management System implemented in C++ with the Qt Framework. It models a real-world banking scenario by handling customer service requests using a linked list-based queue on the FIFO principle. The system provides support for basic banking activities such as account opening, deposit, withdrawal, transaction history, and account closure.

The system is secured by an admin login and password management system, which ensures that only authorized personnel have access to and control over the banking activities. The data is stored using a file handling system, which saves and loads the account and transaction information.

The GUI is implemented using the Qt Widgets library, which provides a professional and user-friendly interface with smooth navigation between the various banking modules using a stacked widget layout. The system perfectly illustrates the application of Data Structures and Algorithms concepts such as linked list-based queues, dynamic memory allocation, and object-oriented programming in C++.

4.2 Achievements

In the course of developing the Queue-Based Banking Management System, a number of technical and practical skills were learned in several fields. A complete functional linked list-based queue system was successfully designed and implemented, which facilitates effective customer service management based on the FIFO principle.

Basic banking system operations such as account generation, deposit, withdrawal, transaction history management, and account deletion were successfully implemented and tested. A secure admin login system with password management was developed, which allows only authorized personnel to manage the banking system.

The Qt Framework was incorporated to develop a professional graphical user interface with smooth navigation between various banking system modules based on a stacked widget arrangement. File management was implemented to store account and transaction information between sessions, ensuring data integrity and reliability.

In the course of this project, a number of technical difficulties were encountered and overcome, including problems with cin buffer handling, Qt widget integration, and module linking. These difficulties greatly improved debugging and problem-solving skills.

In conclusion, this project has given a solid grounding in Data Structures and Algorithms, object-oriented programming with C++, Qt graphical user interface programming, modular software development, and file management, which culminated in the development of a

Chapter 5 Conclusion and Future Development

The project has successfully met its primary objectives of designing a Queue-Based Banking Management System that efficiently handles customer service requests using a linked list-based queue structure based on the FIFO principle. The basic banking operations like account opening, deposit, withdrawal, transaction history, and account closure have been completely implemented and tested. The admin authentication mechanism with password management has been implemented to provide a secure and managed environment for banking operations.

The Qt Framework has been successfully integrated into the project, providing a professional graphical user interface that makes the system usable and friendly. The file handling mechanism has been implemented to provide data persistence between sessions, ensuring data integrity and reliability. The system successfully illustrates the real-world application of DSA concepts such as linked list-based queues, dynamic memory allocation, and modular object-oriented programming in C++.

However, some additional features such as money transfer between accounts, account balance search, and multi-admin support have not been completely implemented due to time limitations.

For future development, it is suggested that money transfer between accounts be implemented, data encryption for security purposes be added, a database instead of file handling be integrated, a customer interface be developed, and cloud-based deployment for remote access be explored.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA: MIT Press, 2009.

[2] B. Stroustrup, *The C++ Programming Language*, 4th ed. Upper Saddle River, NJ: Addison-Wesley, 2013.

[3] Qt Documentation, "Qt Widgets," Qt Project, 2024. [Online]. Available: https://doc.qt.io/qt-6/qtwidgets-index.html

[4] Qt Documentation, "Qt Creator IDE," Qt Project, 2024. [Online]. Available: https://doc.qt.io/qtcreator/

[5] M. A. Weiss, *Data Structures and Algorithm Analysis in C++*, 4th ed. Upper Saddle River, NJ: Pearson, 2014.

Appendix

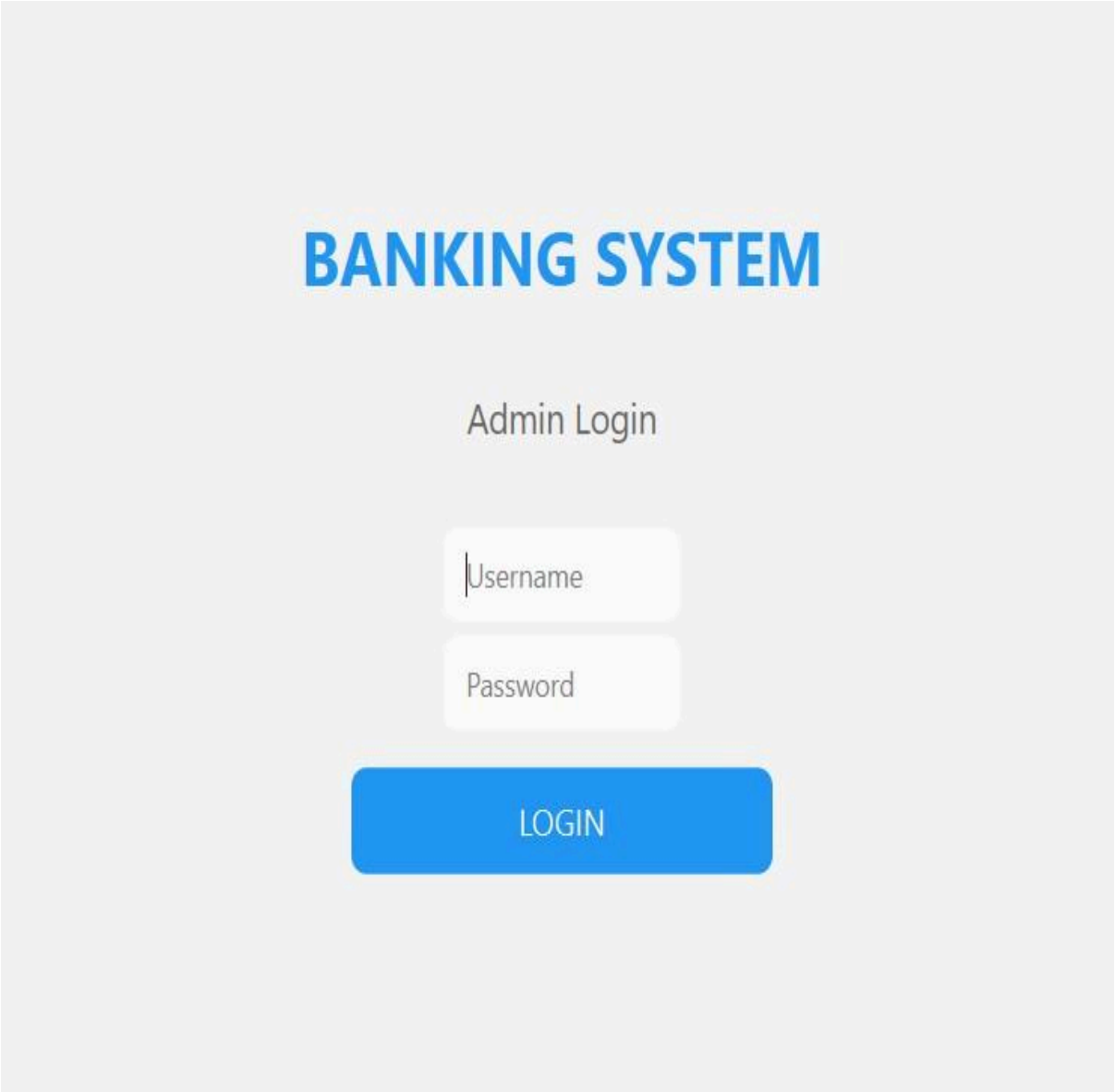

Fig: 3.3 : Login Section

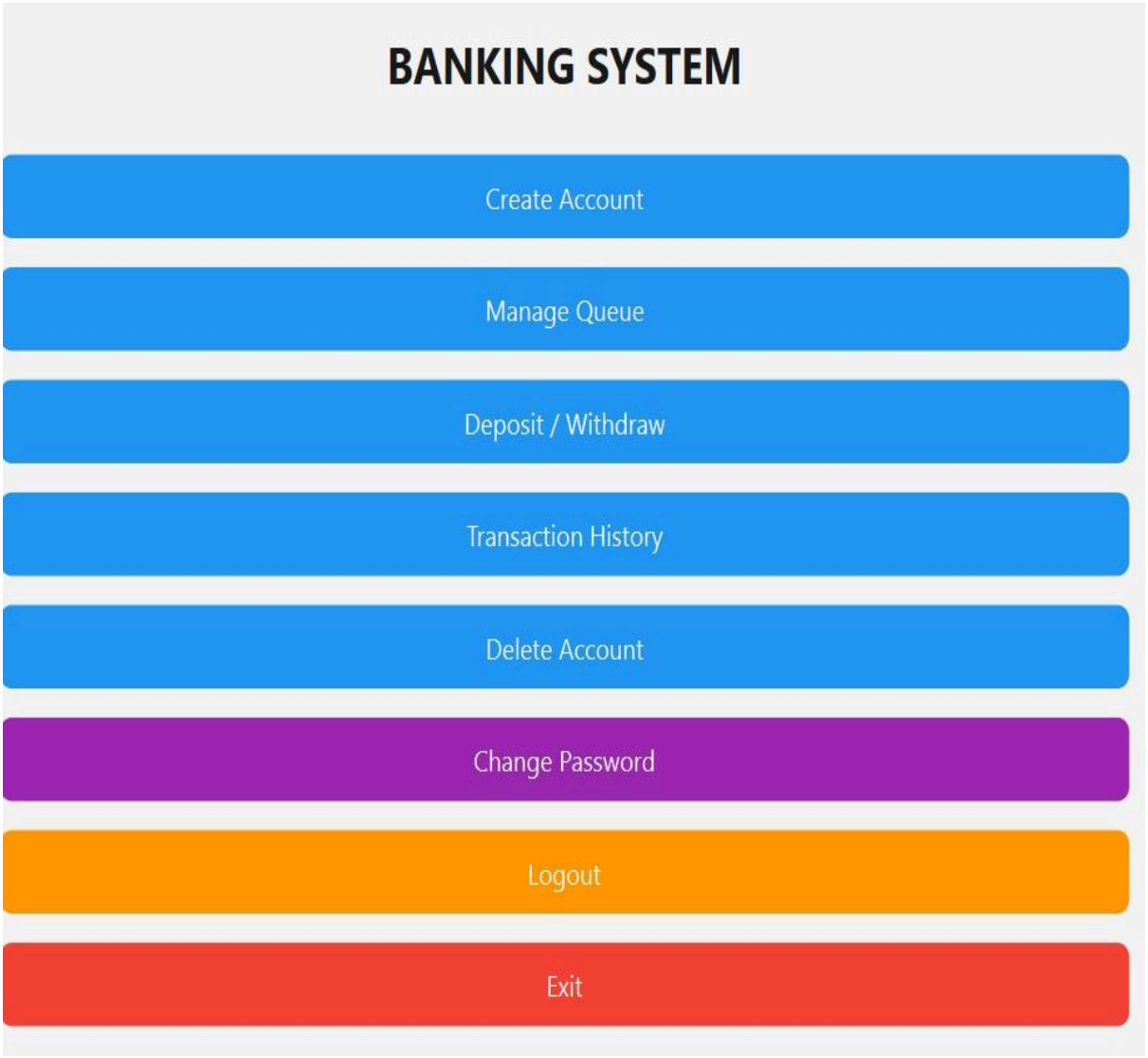

Fig: 3.4 : Main Menu

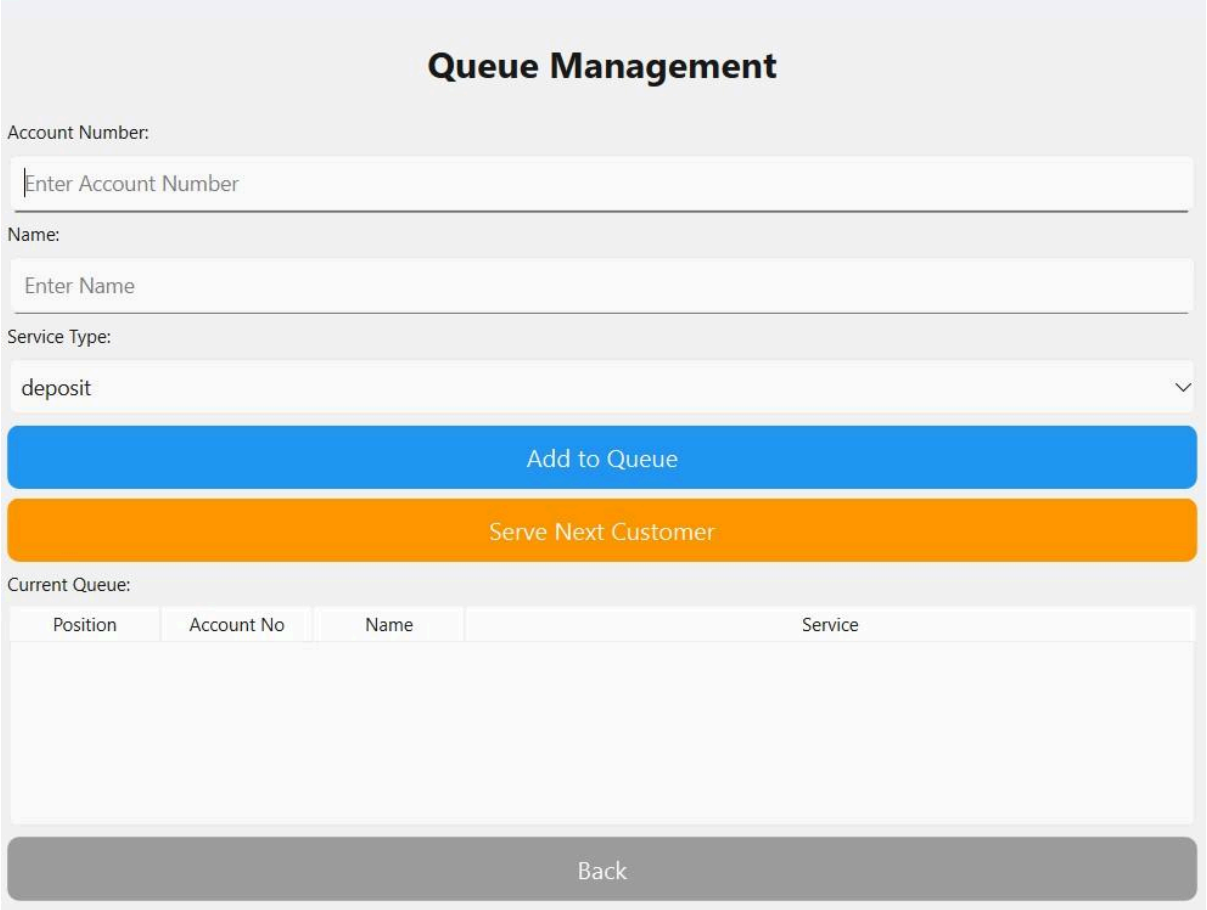

Fig: 3.5 : Queue Management

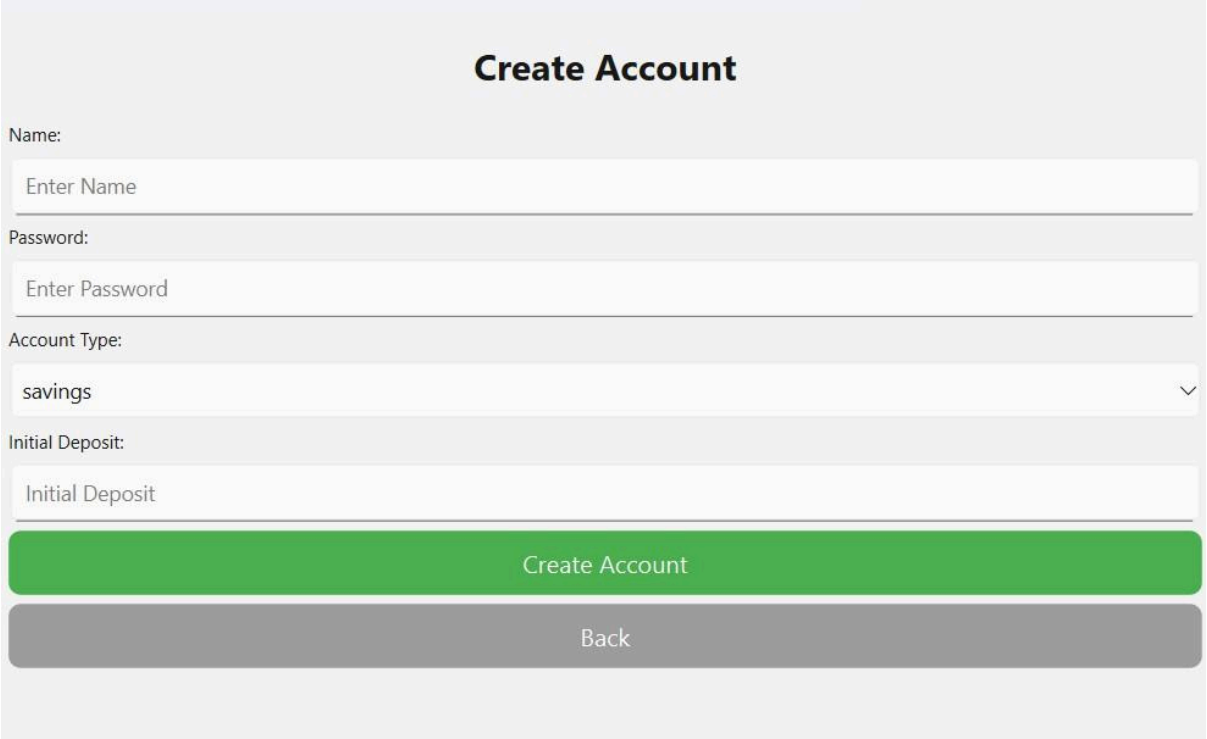

Fig: 3.6 : Account Creation