Restaurant Store System

Maya Sheriff and Sami Shaban

Division of Science, Mathematics and Technology, Governors State University

CPSC-4900: Senior Project and Seminar

Professor Freddie Kato

12/2/2022

Table of Contents

[Abstract 3](#_Toc389161619)

[System Documentation 6](#_Toc639753068)

[System Overview 7](#_Toc780553406)

[System Diagram 7](#_Toc281409494)

[System Inputs: 7](#_Toc1438662655)

[Program Documentation 12](#_Toc1736621063)

[Program Name: Main.java 13](#_Toc1974578491)

[Program Input: 13](#_Toc1674425829)

[Program Output: 13](#_Toc1614259869)

[Program Description: 13](#_Toc72107438)

[Program Name: Database.java 13](#_Toc1235682843)

[Program Input: 14](#_Toc684076571)

[Program Output: 14](#_Toc159449985)

[Program Description: 14](#_Toc430787486)

[Program Name: SceneController.java 15](#_Toc650620334)

[Program Input: 15](#_Toc731058639)

[Program Output: 15](#_Toc83103576)

[Program Description: 15](#_Toc668441605)

[Data Structure or Algorithm Analysis 16](#_Toc10721776)

[Description 17](#_Toc2116187368)

[Analysis 17](#_Toc125443734)

[DB Documentation 18](#_Toc75695810)

[DB Description: 18](#_Toc657231115)

[DB Type: 18](#_Toc84042877)

[DB Tables 18](#_Toc1508959567)

[DB Relationship Diagrams 20](#_Toc720262233)

[Forms and Screens 20](#_Toc462440982)

[Known Issues 27](#_Toc1195341407)

[Future Enhancements 28](#_Toc1164663307)

[Other Resources 29](#_Toc1932528514)

# **Abstract**

***Introduction.*** As time continues, the world is slowly evolving to rely more on software. The software created is used on so many platforms with many different intended purposes. One place where software is implemented is at restaurants. For example, there is software that is created just to be used at restaurants to take orders, take payment and keep track of sales. It is intended to make a more smooth and efficient work environment. However, they are outdated and not always the most effective. ***Problem.*** Many restaurants nowadays still write down orders. We intend to create software that takes orders and saves everything to a database that can be retrieved later. There is software that does this however their UI systems tend to be very confusing. Ours will not. We intend to cut the time to learn down and create places to aid the waitress. This is how our solution to these problems revolutionize how restaurants operate. ***Interdisciplinary.*** This project will affect culinary arts and business administration. This system will allow for more affective transactions for food services. Our system allows the businesses to handle everything with our software from the waitresses/waiters to the chefs and then finally to the managers to oversee everything. Our system is specifically designed for businesses that practice culinary arts. Our system will display all the orders to a screen for the chefs to know what to make. ***Body.*** This system will be used for Surface pro or any windows touchscreen, our system will be developed using Java as a language, and its GUI will be made by java swing. Our system will first allow employees to log in using our clock-in feature. When the restaurant is up and running the waitresses/waiters will be able to take orders; this includes dine in orders, and pickups orders, which can be taken by phone or through walk-ins. Dine in orders will be taken via tablet with our system software installed; orders will be identified with the table number displayed on it. Pickup orders can only be taken by a computer assigned with a register; this order will need a name and phone number to be placed for easy finding later. Once the order is complete, it is then sent to the server which is then sent to the kitchen to be displayed on the screens. When the food is done the kitchen staff sends a notification to the waitress/waiter that the order is done so it can be picked up and taken to the correct table. There will be no confusion as to what order goes to what table because the table number will be on the order from when it was initially placed. Finally, customers who dine in and are done with their meals will then pay; they can either pay by card or by cash. If they choose to pay with card, the waitress/waiter will be able to enter the card information with the tablet without moving a inch from the table. However, if the customer wants to pay using cash, the waitress/waiter will have to go to a cash register station, pull up the order, and ring them out getting proper change then returning it to the customer. On the other hand, for pick up orders the waitress/waiter at the register will have to search the order by name or phone number, then ring the customer out with their desired payment type. Lastly if any device becomes faulty it doesn’t matter because once there is a new device it can pick up from where it left off using the database features. ***Results.*** The users of our system will be the kitchen staff, the waitresses/waiters, and the manager. Our system is designed differently for each user’s needs; each user type will be given a UI that corresponds to their job’s roles. This means that the chef who logs in won't be able to make an order like a waitress/waiter. However, the manager is given access to all features as the overseer. The manager is also allowed to modify pay, see everyone's clock in/out logs, order history, and customer info. ***Conclusion.*** As software is becoming increasingly dominant in the world it's important to keep up with the times. Most restaurants today still use outdated methods, like writing down orders, to accomplish their goals. Our system, as explained, is designed to completely overhaul this and make everything simpler and easier.

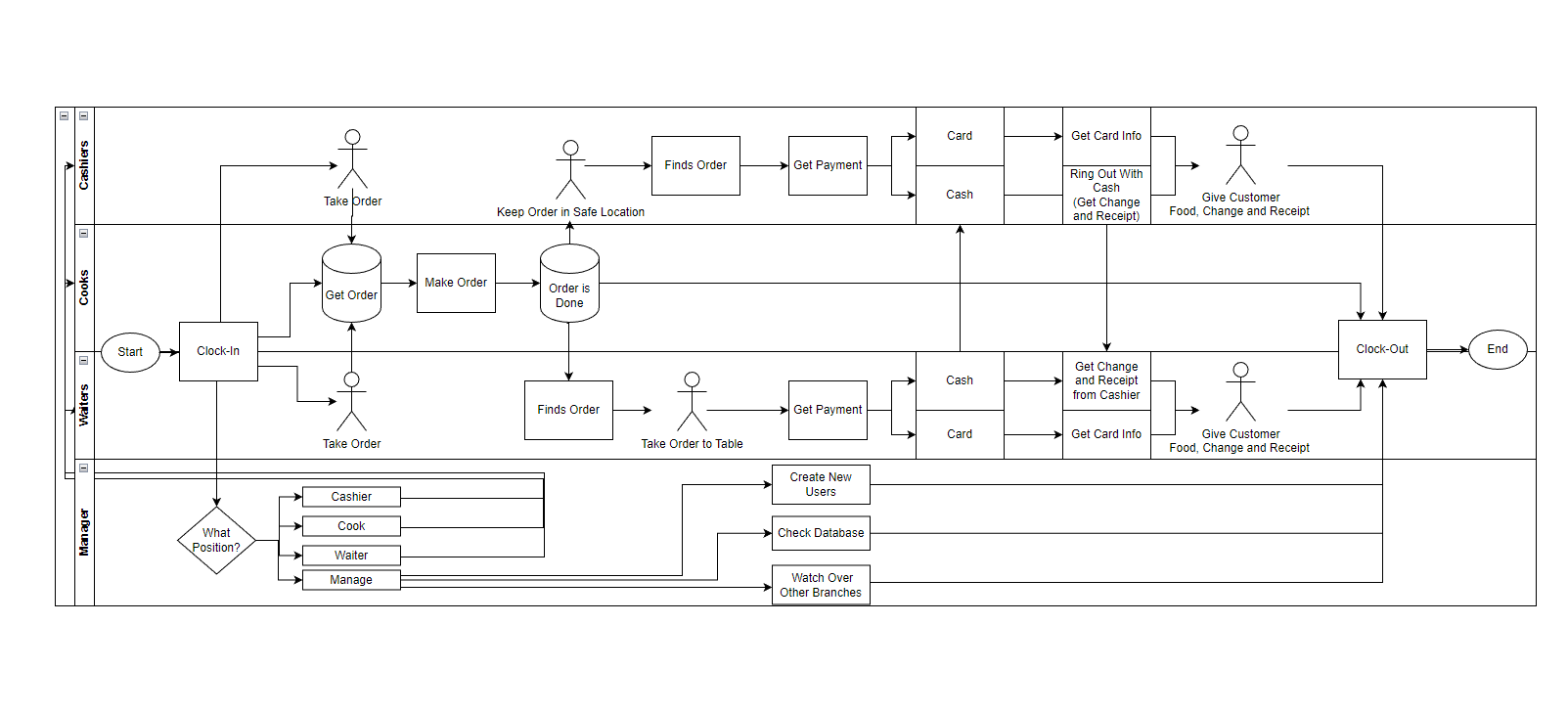
# System Documentation

## System Overview

Describe your system and explain the design approach/principles taken to define your solution.

Our system allows Waitstaff to log in orders and send them to the database. Once in the database the kitchen staff then makes the order. Once it deletes it from the table and adds it into another table. In the waitress UI they are able to search for orders that are completed by the kitchen and grab them and serve customers. Once served and done the waitstaff can only pay using credit card for the order. If the customers want to use cash, then the order must be searched by the cashier to authorize the cash transaction. Mangers are able to assist the staff by logging in to their own screens.

## System Diagram



## System Inputs:

Login Interface:

* Login
  + Kitchen Login
  + Waitstaff Login
  + Manager Login
* Clock in
  + Kitchen Clock in
  + Waitstaff Login Clock in
  + Manager Login Clock in

Kitchen Interface:

* Click order
* Mark item as done
* Mark order as complete

Waitstaff Interface:

* Add item
* Delete item
* Add note
* Clear order
* Take payment
* Complete order

Manager Interface:

* Use kitchen interface
* Use waitstaff interface
* Create new employee user interface
  + Enter name, job title, wage, id number, and hours
* Edit employee user interface
  + Enter employees name, employee ID, wage, job type, hours, and employment status

System Outputs:

Login Interface:

* Login:
  + Kitchen Login: checks database for correct password for a user then displays kitchen interface if the password belongs to a kitchen staff member. If password is incorrect an error message will display explaining why (no user with the entered login, incorrect format, etc.).
  + Waitstaff Login: checks database for correct password for a user then displays waitstaff interface if the password belongs to a waitstaff member. If password is incorrect an error message will display explaining why (no user with the entered login, incorrect format, etc.).
  + Manager Login: checks database for correct password for a user then displays manager interface if the password belongs to a manager. If password is incorrect an error message will display explaining why (no user with the entered login, incorrect format, etc.).
* Clock In:
  + Kitchen, Waitstaff and Manager Clock in: will be the same as logging in but will also send the users name and clock in time to the database.

Kitchen Interface:

* Logout: will send user back to login interface.
* Clock Out: will send user back to login interface and send the users name and clock out time to the database.
* Click order: will display the order clicked in a ticket format.
* Mark item as done: will mark that specific item done (can only be seen by other kitchen staff).
* Mark order as complete: will make the entire order complete and will be displayed as done in database so waitstaff will know.

Waitstaff Interface:

* Logout: will send user back to login interface.
* Clock Out: will send user back to login interface and send the users name and clock out time to the database.
* Add item: will add the item, price, and option to add a note.
* Delete item: will delete item and its additional information.
* Add note: will save a note corresponding to the item it refers to.
* Clear order: will delete all items, pricing, notes, table number/customer information from ticket
* Take payment: will take card payment or prompt to give proper change.
* Complete order: will end order to database for the kitchen staff to recall later (this can be done before payment is taken).

Manager Interface:

* Logout: will send user back to login interface.
* Clock Out: will send user back to login interface and send the users name and clock out time to the database.
* Use kitchen interface: will display kitchen interface along with full usability listed above.
* Use waitstaff interface: will display waitstaff interface along with full usability listed above.
* Create new employee user interface
  + Enter name, job title, wage, id number, and hours: when this information is entered correctly, it will great a new user in the database which can be used when logging in. However, if information is entered incorrectly a error message will display explaining why (missing name, wrong format, etc.).
* Edit employee user interface
  + Enter employees name, employee ID, wage, job type, hours, or employment status: when this information is entered, it will update the according information in the database.

**System Processing:**

The system starts at the login frame. The user enters an ID and the system checks if that came from the database. If not an errors shows. The manager can always make a new user with a unique ID which the system checks to make sure its not a duplicate ID. Once the user is given ID by the manager then the manager gives them a job title with a salary. Depending on the title, the user can only access certain parts of the system whether it be the kitchen, waitstaff, manager, or cashier. The waitstaff can log in orders or pull in orders from the database. The purpose is to either pay for the order or enter orders made from customers.

**System Requirements:**

Describe the resources necessary to implement your system. This may include required data files, software (***not the IDE***), DBMS, and OS under which the system runs.

* Microsoft Access
* Java
* Text File

**System Implementation:**

Describe the setup procedure to follow in order to implement your system. What steps did you take in order to setup the environment to be able to run your system? What are the required configuration parameters needed to duplicate the setup?

* This should be a step by step process that one of your peers could follow.

Our system was built using open-source software such as “UcannAccess”, javaFX, Scene builder, and GitHub. Scene builder runs off of javafx code and was convenient. Scene builder allows the software engineer easily to create containers like Vboxes, panes, text fields, and buttons. “Ucanaccess” allows the programmer to store any data generated from code to be stored in access, which is Microsoft database. Our java code was mostly ran off of eclipse which is also free but any other IDE that can run java will do. GitHub allowed me and my partner to share code to each other without copying and sending files. Git allows us to fix a mistake and go back to a commit that was made.

# Program Documentation

## Program Name: Main.java

### Program Input:

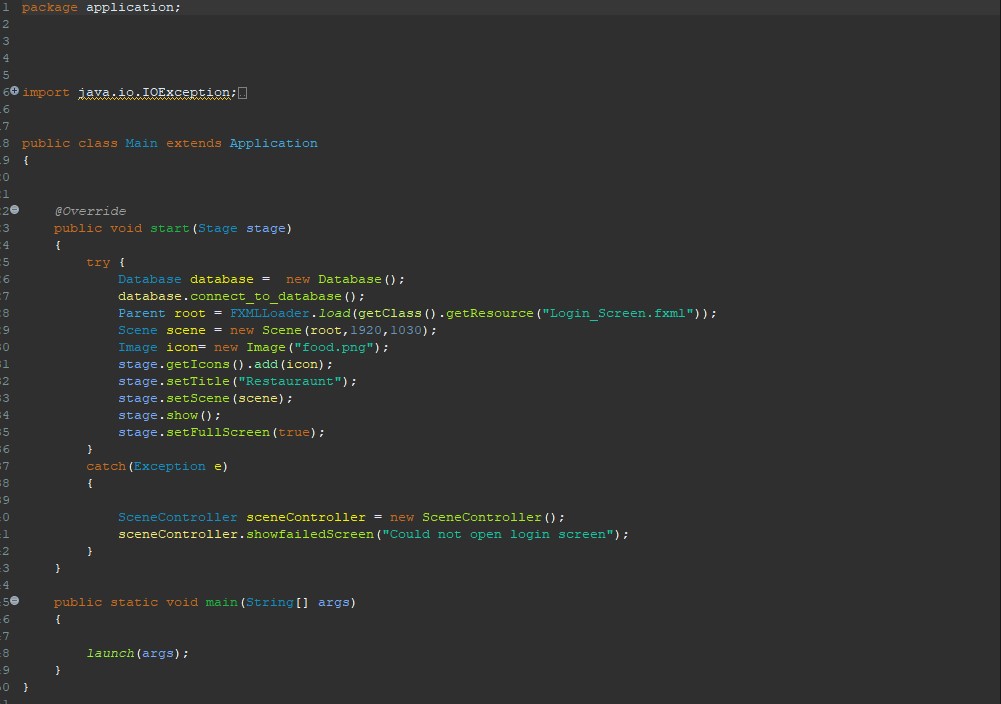
* Takes login information.

### Program Output:

* Displays the corresponding interface for login.

### Program Description:

Runs login screen.



## Program Name: Database.java

### Program Input:

* Connects to the database automatically.
* Queries data in the tables from the database and stores it in a result set
* Closes the database

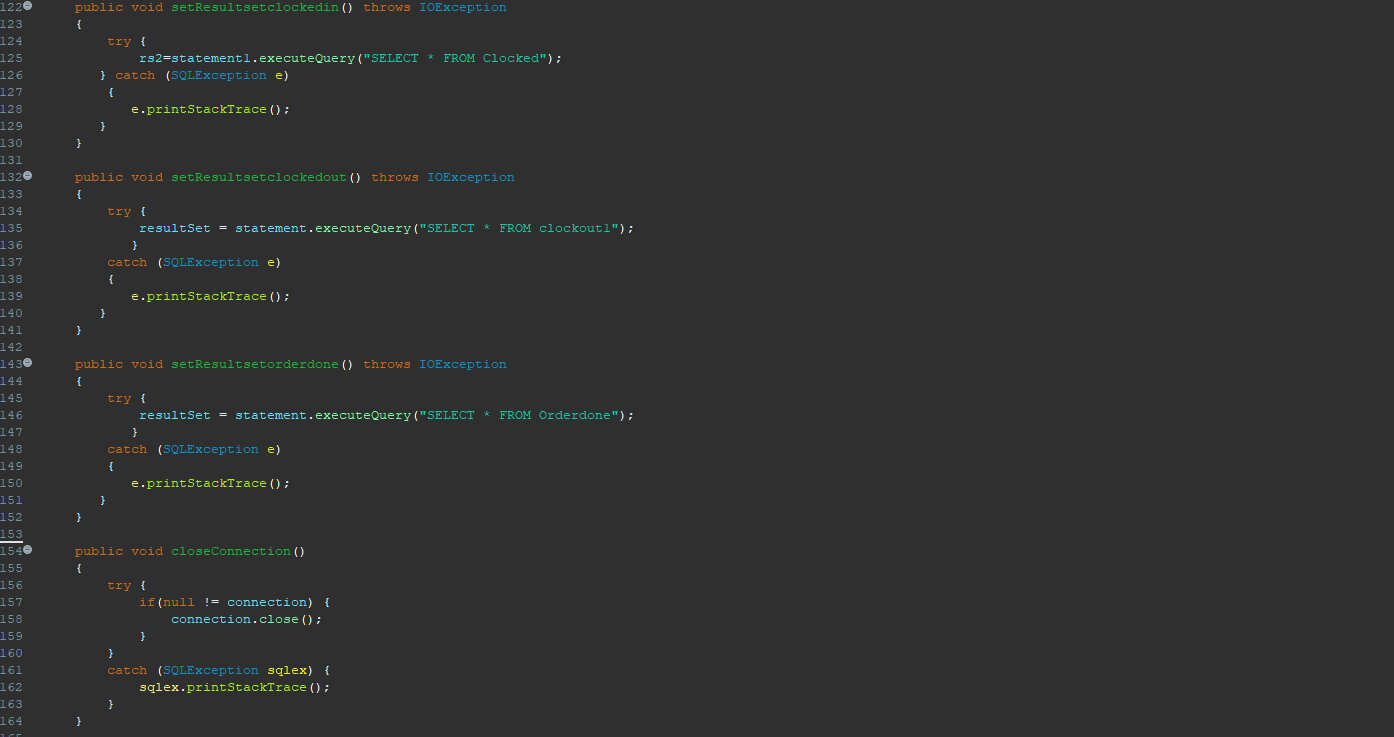
### Program Output:

* The result set is then used to store data and be recalled when needed.
* Uses a prepared statement that takes in SQL code and uses it to update or insert in the database

### Program Description:

Connects to database, retrieves and sends queues to and from the database.





## Program Name: SceneController.java

### Program Input:

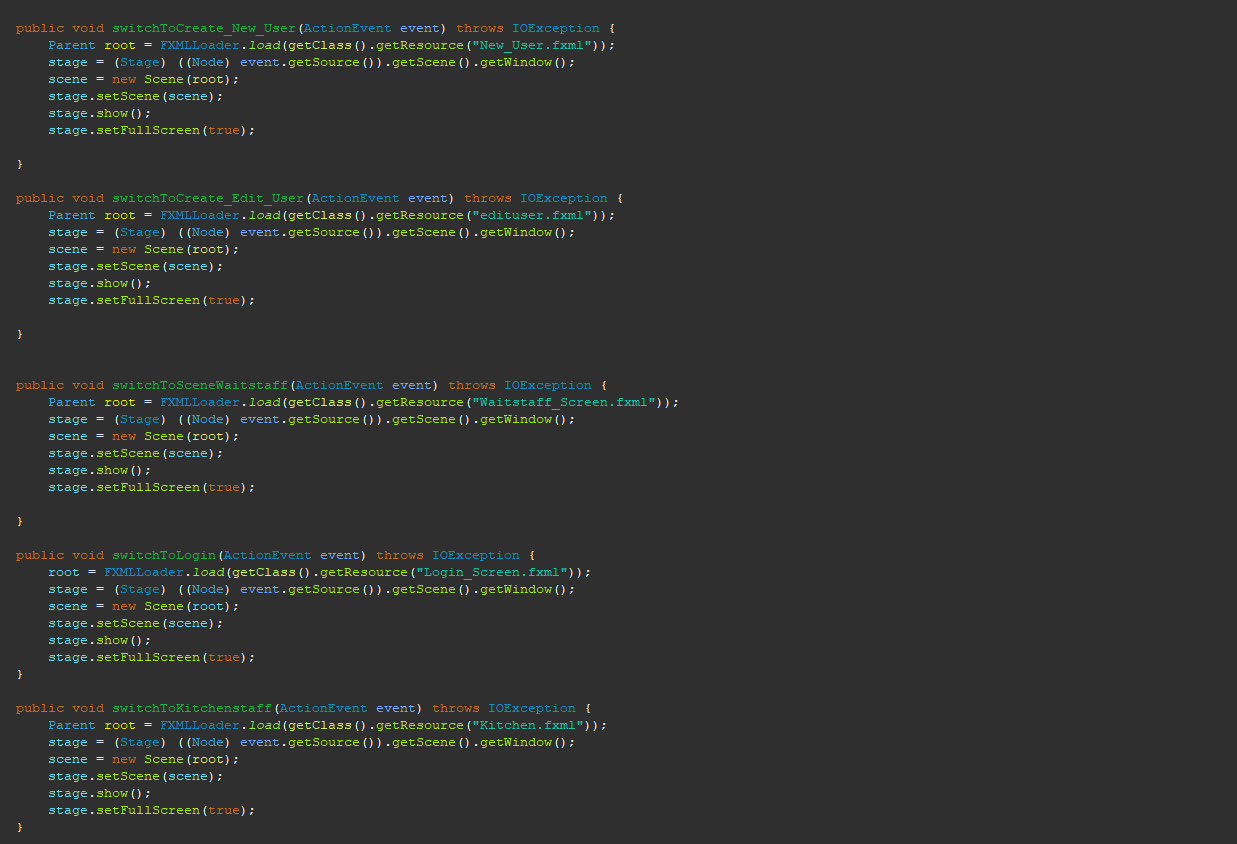
* Data from the text fields; table number, order number, user id, password,
* Queries from tables such as employees and orders
* Whenever objects are made it stores them in an array list

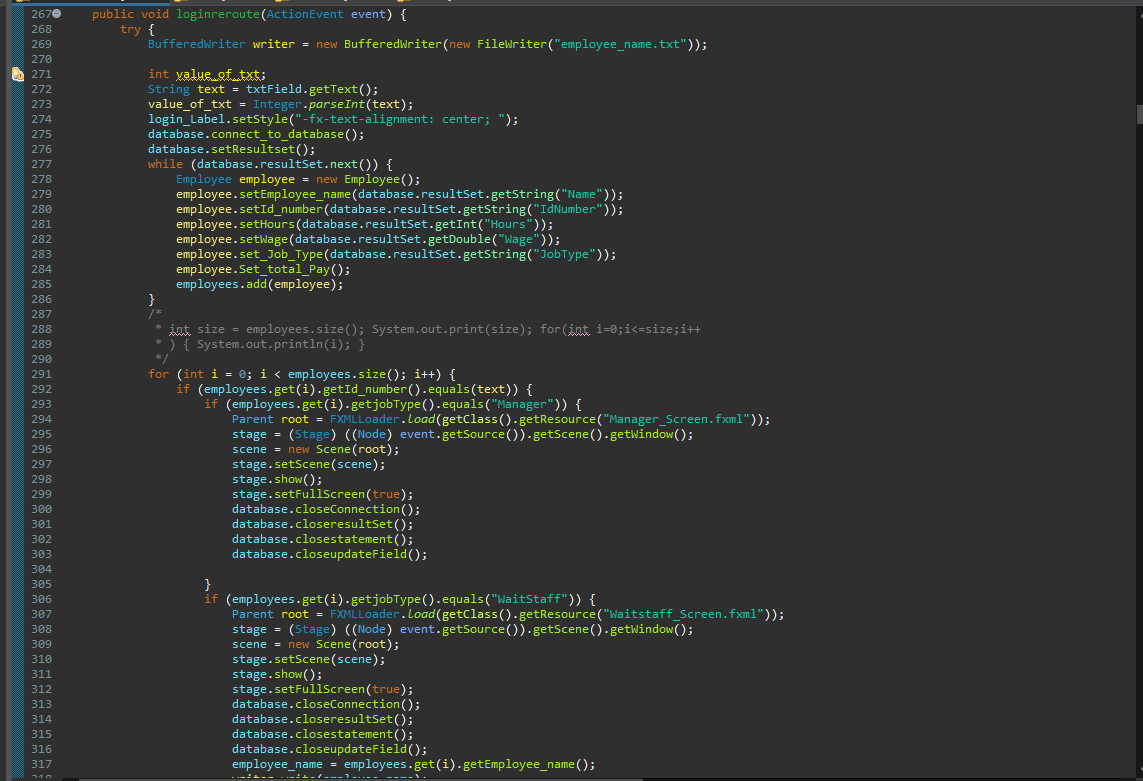
### Program Output:

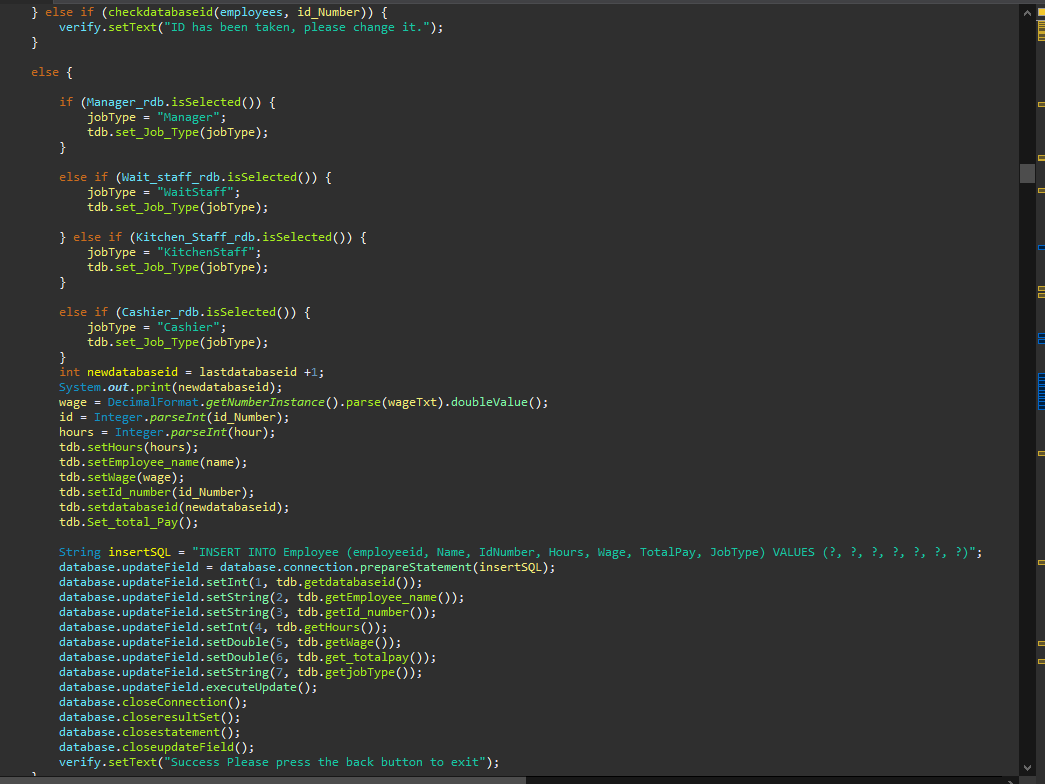
* Creates new users
* Creates new orders
* Recalls orders in certain UIs
* Updates Orders such as order made, and order paid

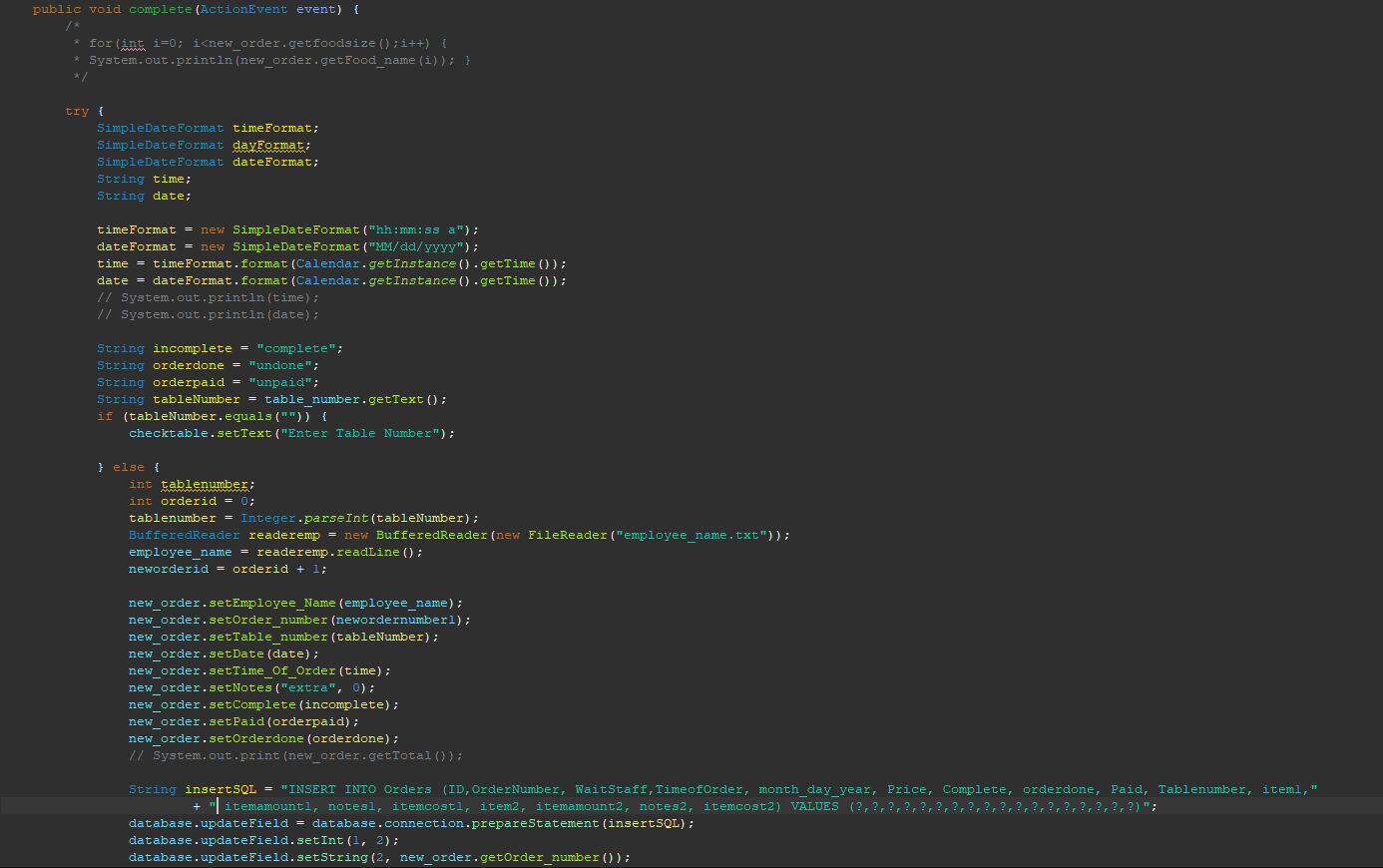
### Program Description:

The methods here switch from scene to scene. It is specifically meant for buttons.

This method is for the user log in screen. Routes to each scene.

This inserts the data entered from fields into the table ’employees’ in the database.

This sends orders to the database and clears the ticket on the interface.



# Data Structure or Algorithm Analysis

Data Structure or Algorithm Name:

### Description

Describe one of the project’s major data structures or algorithms, and, its general use.

### Analysis

1. What are the memory requirements, based off of the general number of elements used?

Our system primarily uses array lists that can be dynamically allocated and stored. Usually, we take data from the database and while the result set loops it gets stored in the array list of objects. Some objects have their own array, so it needs an additional loop for that. The size of the jar file is about 20 megabytes.

1. What are the pros and/or cons associated with the element?

The pros of the array list is that any number of elements can be stored there. However, the issue is that if there is an element that needs to be deleted arrays can’t delete them just make the index null.

1. What is an alternative program element that could have been used?

Nodes, not only can they store objects, but it can also delete them with ease as well. However looping thru them takes more time to do.

1. Why is the element appropriate for your project’s use?

Array lists are dynamic and in runtime we need to dynamically allocate objects depending on order items or how many employees there are.

# DB Documentation

## DB Description:

Give the name and description of the DB. The description should detail the type of data that the DB is used to store.

Microsoft access is an advanced database that gives the programmer many capabilities such as sending information to an excel sheet, repairing a database, and giving each column its own datatype. The systems database stores all types of information such as dates, numbers, and strings.

## DB Type:

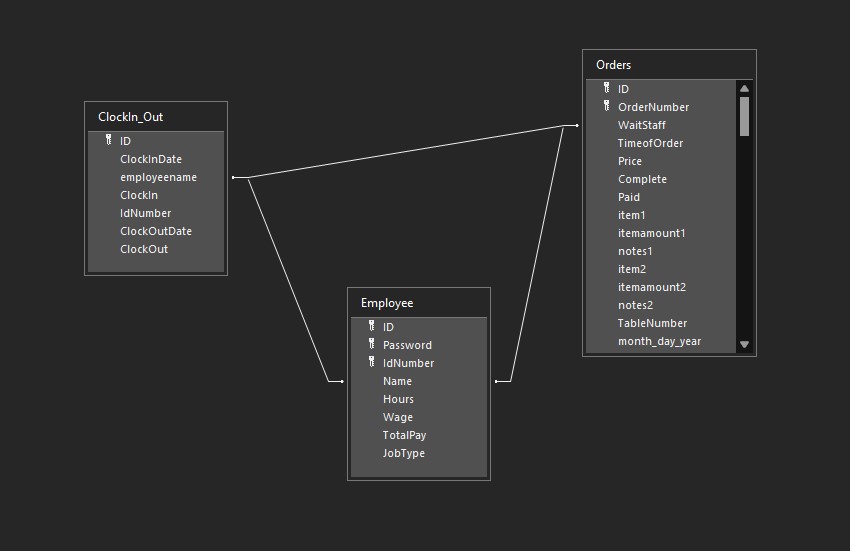
Describe the type of DB (e.g. MySQL Relational DB; MS SQL Server Relational DB)

Microsoft access which is a relational database

## DB Tables

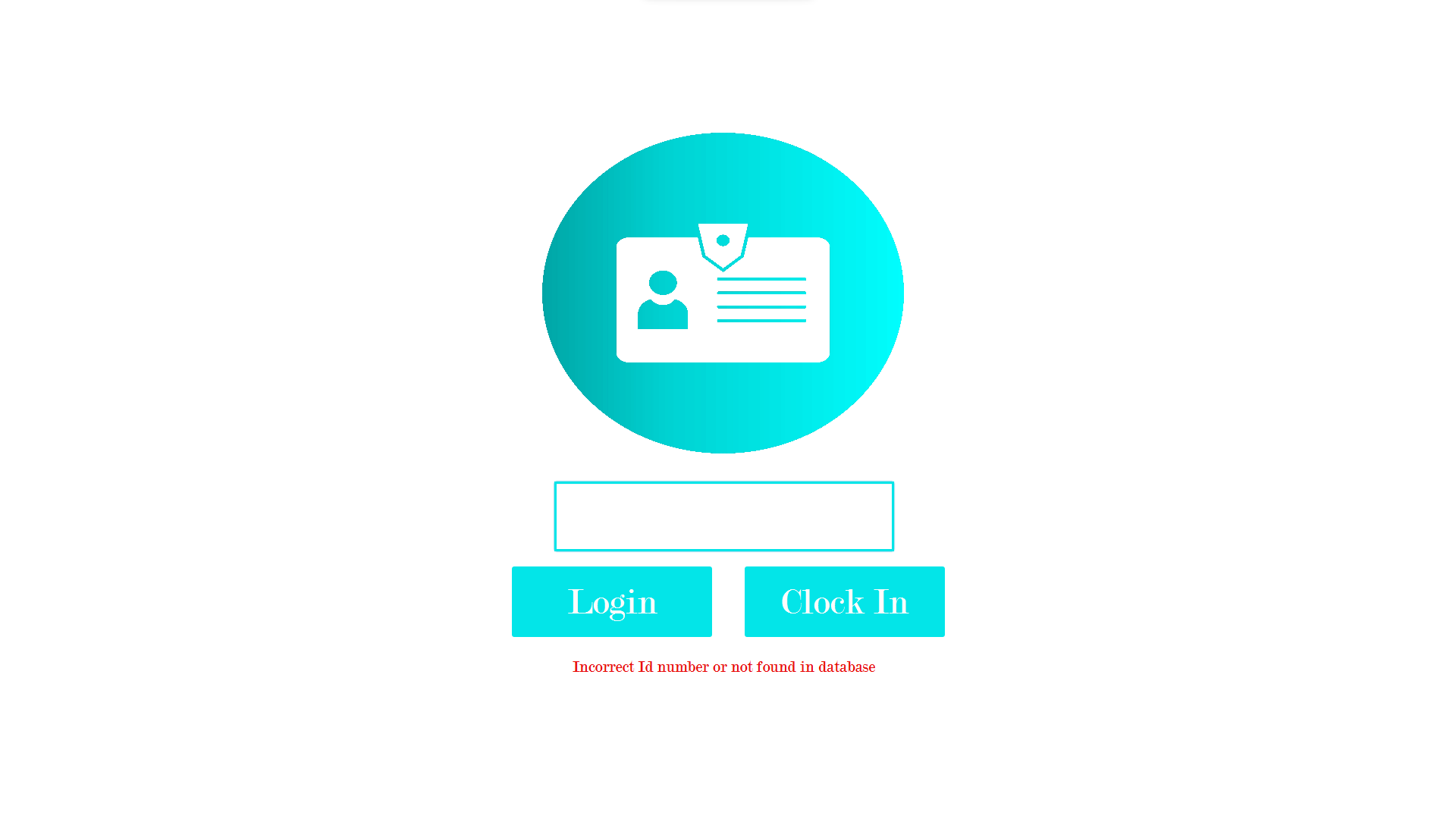
* Employee
  + EmployeeID: Primary Key
  + Name
  + Password: Primary Key
  + Hours
  + Wage
  + TotalPay
  + JobType
* ClockIn\_Out
  + ID: Primary Key
  + EmployeeName: FK Name
  + ClockInDate
  + ClockIn
  + ClockOutDate
  + ClockOut
* Orders
  + ID: Primary Key
  + OrderNumber
  + CustomerName
  + CustomerNumber
  + Waitstaff: FK Name
  + TimeofOrder
  + Month\_day\_year
  + Price
  + Complete
  + Paid
  + TableNumber
  + Item
  + Itemamount
  + Notes
  + Itemcost

## DB Relationship Diagrams



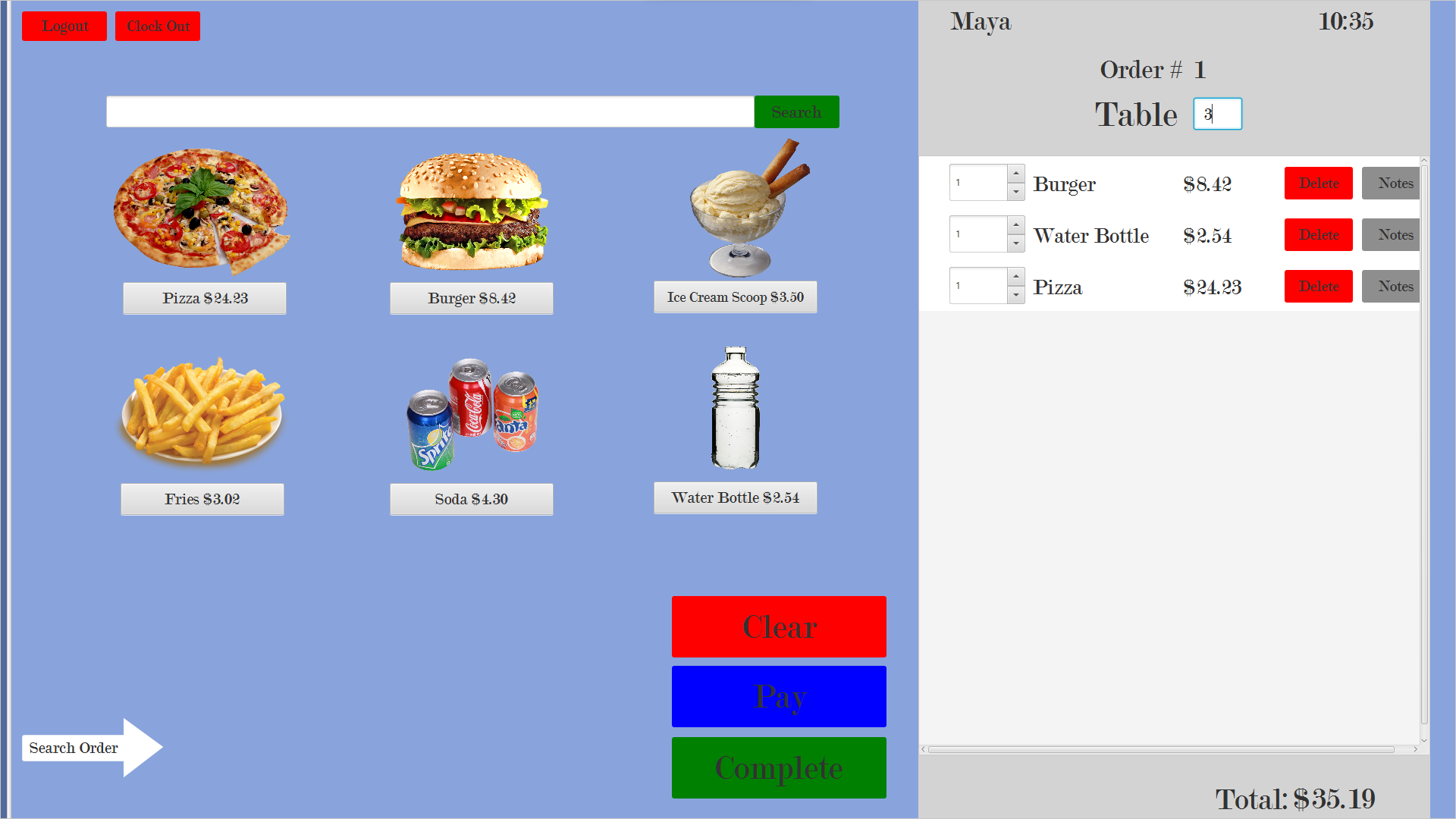
# Forms and Screens

Login Interface:

The login is the first screen that shows when the program is ran. It requires the employee’s password only. Once the password is entered, login or clock in can be pressed. 

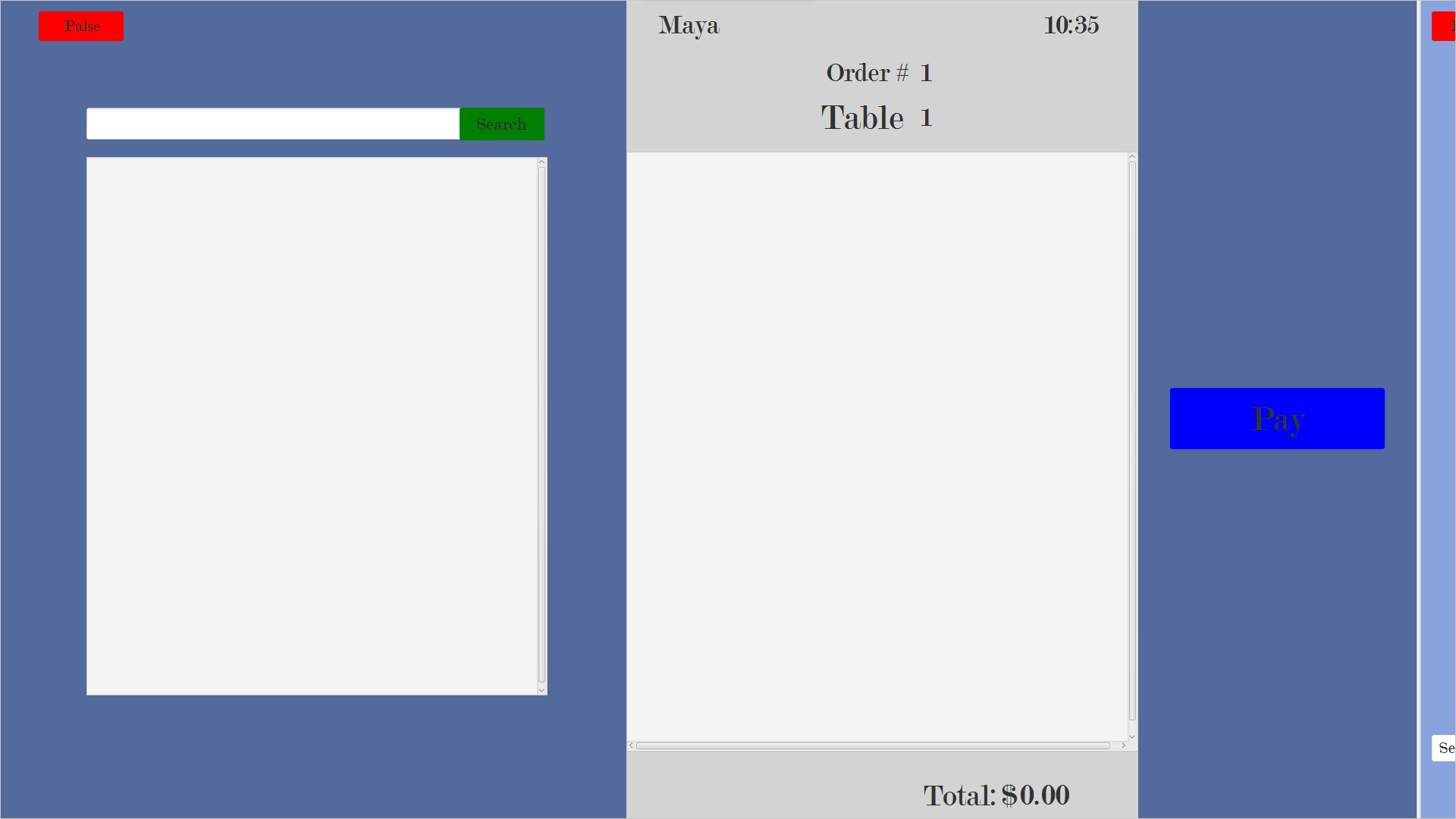
Waitstaff Interface (Taking Orders):

The waitstaff screen displays after a waitstaff member logs in. They are able to press the menu item buttons which will display the proper information on the ticket on the right. They can then delete items form the ticket, add notes, clear the whole order, take payment, and complete orders.



Waitstaff Interface (Finding Orders):

When the white bar is dragged to the right, a second screen reveals itself. This screen is used to find orders in the system. Orders can be searched to complete payments or searched to check for its completion statues from the kitchen.

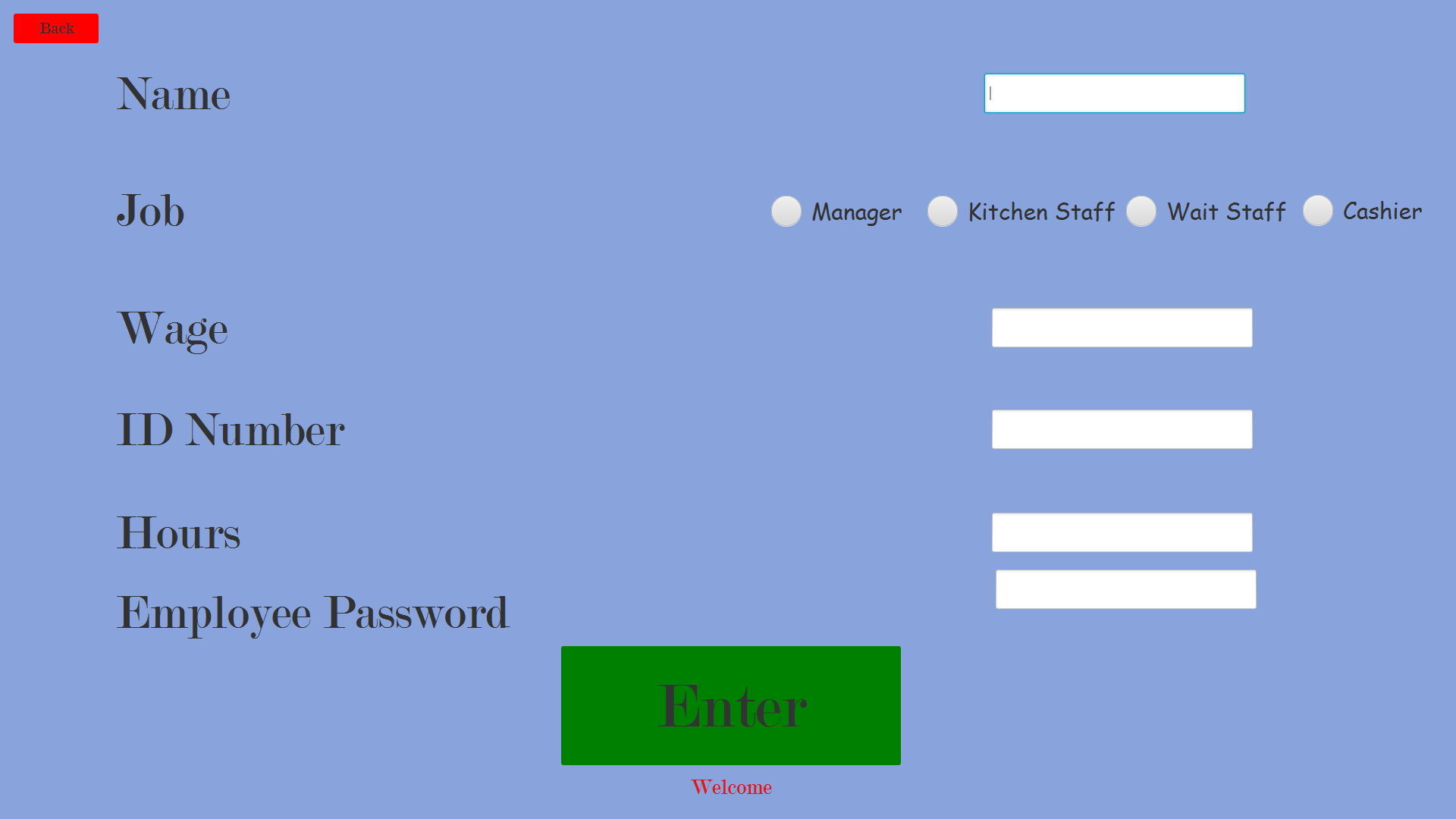


Manager Interface:

This screen gives the manager options to do. The manager can go into different screens in case he needs to assist either the kitchen or the waitstaff. He also has unique capabilities such as editing and creating users. 

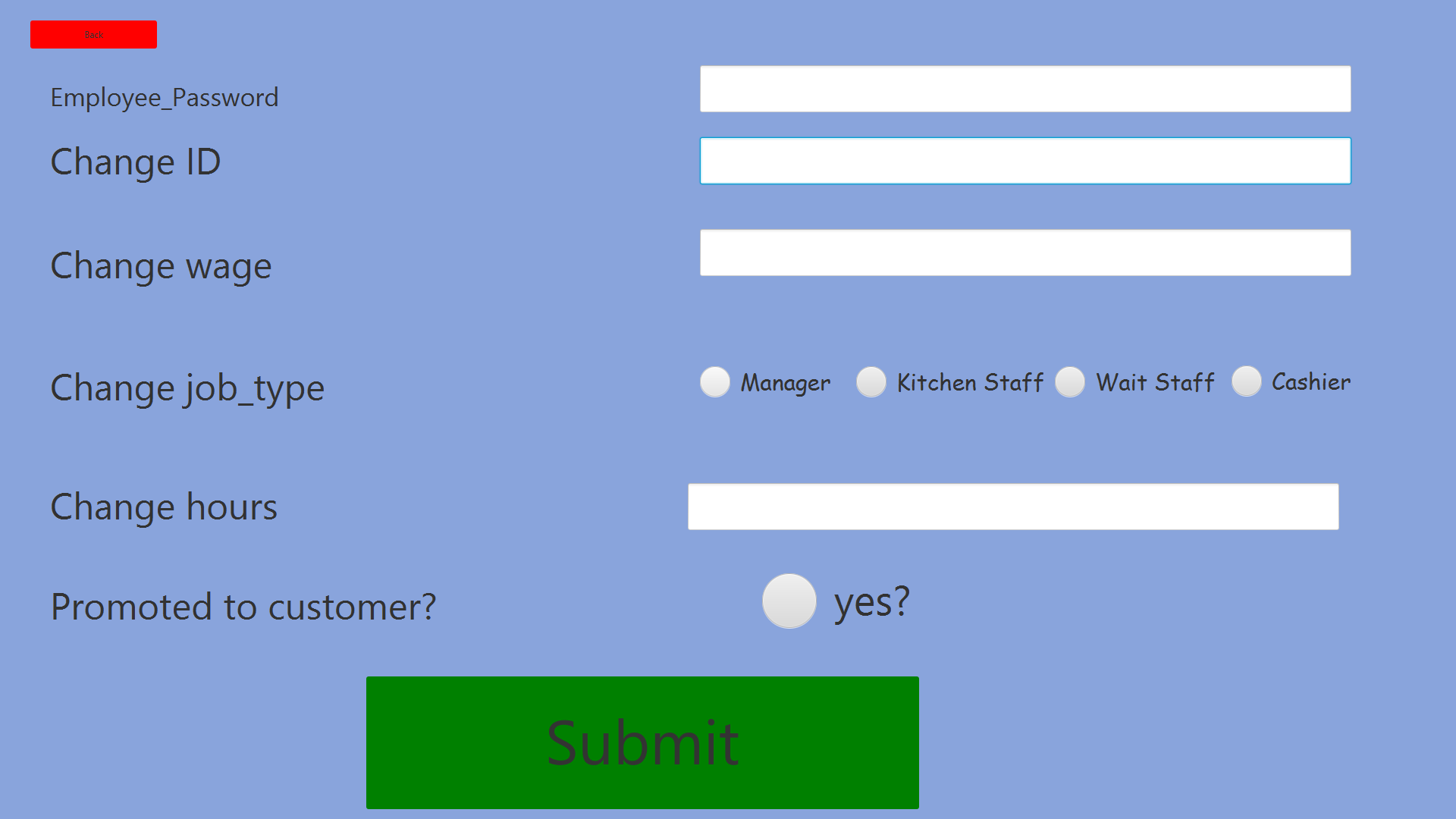
Create User Interface:

This is where the manager makes users. The system checks to see if the ID number is a duplicate key and presents an error to the screen There are multiple checks to make sure every information in the table is correct. If it passes all checks then it will insert in to the database with the new employee and his given title.



Edit User Interface:

This interface gives the manager access to change the number of hours worked, wage, and most importantly job type. Employees can be promoted or fired. Their password is a unique key only the manager can get. This is to make sure anyone who steals the user id can be changed by using the password as a unique key to the database.



Kitchen Interface:

This interface is one of the key elements of the system. It allows the kitchen staff to view all the orders it needs to prepare. Also, it allows us to view the orders in chronological order. There are also buttons for clearing and refreshing orders. The last button is for completion. This updates the database and makes it available for the cashier/ waitress to do.



# Known Issues

1. Time- Showing the time was difficult to do. Threading with javaFx is extremely difficult due to other threads not being able to access the javaFx thread. Clocking in and clocking out did work because it was done in the moment of execution.
2. Order complete notification- having a pop-up order completion notification to the waitstaff that took the order did not work. However, the UI does do its job of recalling data needed.

# Future Enhancements

1. Adding a pop-up order completion notification to the waitstaff that took the order.

# Other Resources

1. GitHub
2. UcanAccess
3. JavaFX
4. Scene builder
5. Professor Katos video on databases