Software Design Specification

Restaurant Ordering System

Prepared by: Abbey Herter, JJ Rezaei, and Anthony Ngo

October 13th, 2023

Table Of Contents:

System Description:

The Restaurant Ordering System is a simple software system that makes the order-taking process easier and more efficient for restaurant employees and management. The primary goal of this system is to ensure customer satisfaction and user experience by maintaining straightforward and quick functionality of the system.

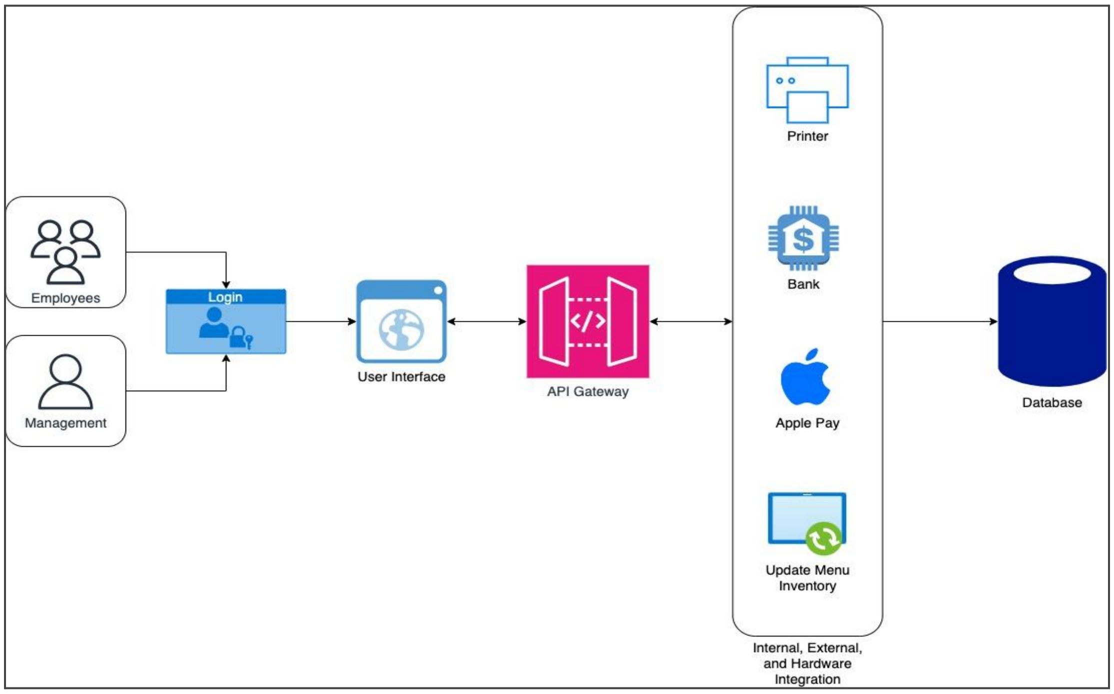
Users, the employees and management, will be able to login to the system with their own special account, access the user interface where they will place a customer's order, complete customer payment, and output a receipt for both the customer and the kitchen. Management will have additional control that will allow them to update the inventory and access database information as well.

Keeping a user-friendly system will not only benefit the restaurant with the ability to produce more orders but also guarantees a quicker runtime for the system as well.

This Software Design Specification includes Software Architecture Diagrams and UML Class Diagrams, both provided with visuals and descriptions to enhance the comprehension of the system's functions including its components, classes, attributes, and operations.

Software Architecture Overview:

Restaurant Ordering System Software Architecture Diagram



Software Architecture Diagram Descriptions:

Management: This is a user or users of the system that have administrative and authoritative functions over the system. This user will have to surpass a login page before accessing the UI. Users have the ability to oversee and control parts of the software system including updating the menu inventory along with backing up the inventory so that the information in the UI is up to date.

Employees: This is a user or users of the system whose role is to successfully navigate through the UI so that the order information is sent through the API Gateway and to the Internal, External, and Hardware Integration such as the printer, bank, or 3rd party systems. Users will have to surpass a login before accessing the UI as well.

Login: The login page acts as a security gateway to the UI. The page prompts the user to input their login information in order to authenticate their access.

User Interface (UI): The User Interface (UI) is a simple interface that allows the user to interact with different features and navigate through the system efficiently. These features include selecting if the order is dined in or if the order is to-go along with creating the order for the customer. Users will also be prompted to administer the customer's payment which will be sent from the UI through the API Gateway to the bank or Apple Pay. Furthermore, users will then be prompted to print a receipt which will be sent from the UI to go through the API Gateway to then reach the printer.

API Gateway: The API Gateway acts as the middleman in this Software Architecture Diagram and manages communication between the UI and the Internal, External, and Hardware Integration. The API Gateways functions include gathering, transforming, and requesting data along with routing components together.

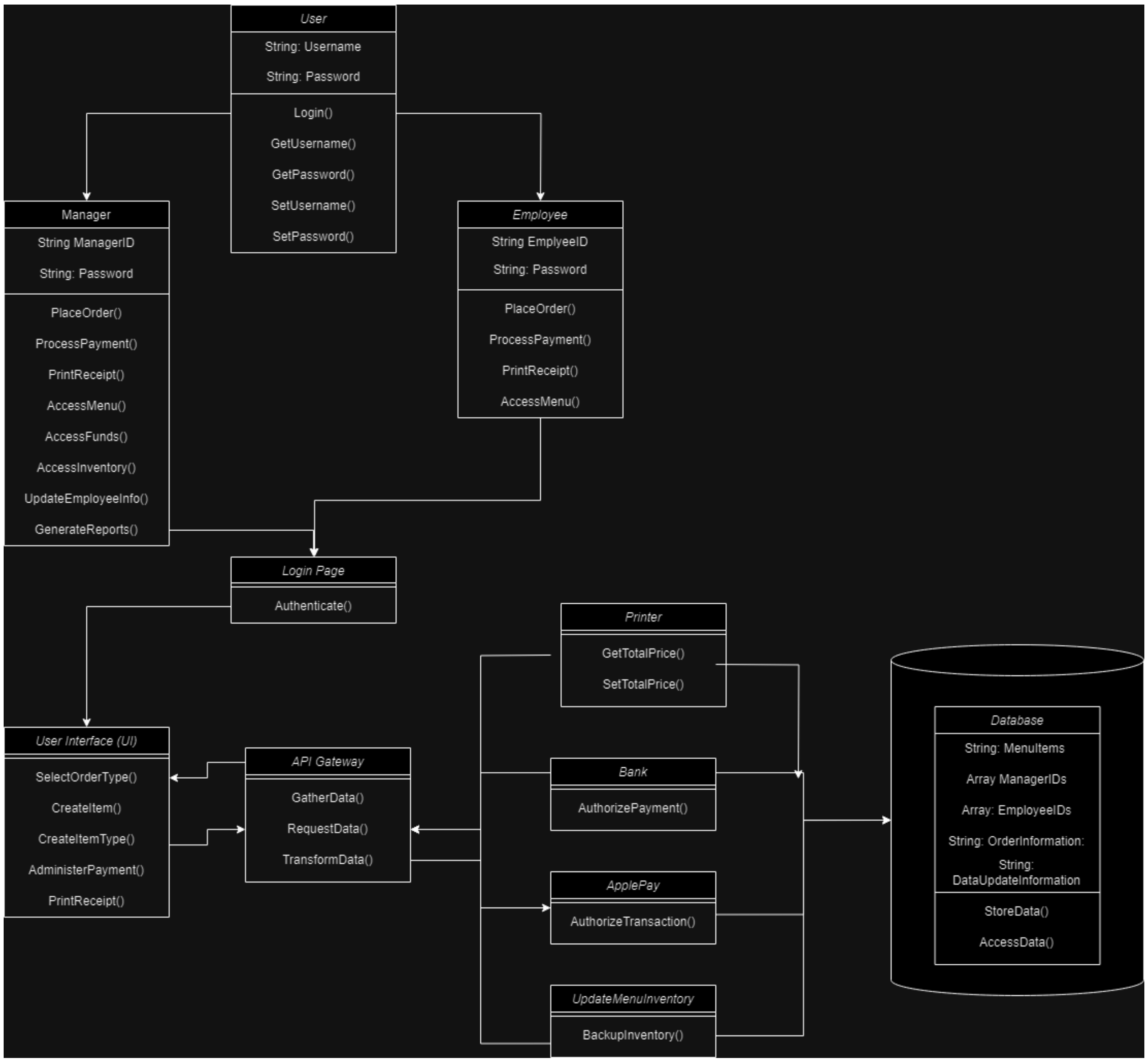
Internal, External, and Hardware Integration: These are the components that communicate through the API Gateway back to the UI to complete requests made by the user.

> Printer: The printer is a hardware component that is prompted by the user to output the customer's receipt and the receipt for the kitchen.
>
> Bank: The bank is an external service that will authorize the payment given by the customer.
>
> Apple Pay: This is a third-party payment service that will authorize the transaction given by the customer through insertion or tap.
>
> Update Menu Inventory: This is an internal integration that allows management access to frequently backup and update the menu inventory including changing pricing, stock amounts, menu additions, and menu

removals. This integration will check those features and communicate the feedback through the API Gateway to the UI.

Database: The database is an important internal component of the system that contains all of the data storage. Within the database is the menu item names, prices, and descriptions along with order information and data update information. All of the information that has traveled from the UI through the API Gateway to the Internal, External, and Hardware Interfaces is then stored in the database and is able to be accessed by the management.

UML Class Diagram:



UML Class Diagram Descriptions:

User

EmployeeID string - used to log into the employee account

Password string - used to log into employee account

Login() void - logs employee to either manager or employee account depending on employee ID or manager ID

getUsername() string - retrieves employee username from database

getPassword() string - retrieves password of login from database

setUsername(string) string - sets username information in database to variable

setPassword(string) string - sets password information in database to variable

Manager

ManagerID string - used to log into the manager account

Password string - used to log into manager account

PlaceOrder() void - places an order for a customer

ProcessPayment() void - processes the payment for the customer

PrintReceipt() void - prints the receipt for the customer

AccessMenu() void - returns the menu contents (stored in the database)

AccessFunds() void - returns the funds available for the restaurant (stored in the database)

UpdateEmployeeInfo() void - used to modify or delete employee information

Employee

PlaceOrder() void - places an order for a customer

ProcessPayment() void - processes the payment for the customer

PrintReceipt() void - prints the receipt for the customer

AccessMenu() string - returns the menuItems

Login Page

Authenticate() bool - ensures that the previous committed action attempted is valid

User Interface (UI)

SelectOrderType() void - sets a certain order type to item

CreateItem() void - creates a menuItem

createItemType() void - creates a certain item type

AdministerPayment() void - administers a type of payment from the customer

PrintReceipt() void - sends a print request to the printer

API Gateway

GatherData() void - aggregates the data from database into array of strings

RequestData() string[] - returns data aggregation from gatherData

TransformData() void - allows user to change data from database

Printer

GetTotalPrice() int - gets and calculates price of items selected

SetTotalPrice(int) int - sets the total price of the items selected to a variable

Bank

AuthorizePayment() void - ensures payment is valid

ApplePay

AuthorizeTransaction() void - ensures payment is valid

UpdateMenuInventory

BackupInventory() void - creates a backup inventory log with all database info

Database

StoreData() void - stores data into the database

AccessData() string - returns data from the database

Development Plan and Timeline

Partitioning of Tasks:

We decided to partition the tasks as equally as possible. Abbey handled the software architecture overview and its description. Anthony handled the UML diagram with some input from JJ and JJ handled the UML class diagram descriptions. We thought this would all take approximately the same amount of time for each individual, as each section would take the individuals an estimated 1-2 hours to complete.

Team Member Responsibilities:

As mentioned previously, Abbey handled the software architecture overview and description. This means ensuring that the software architecture is accurate to the group's vision and understanding of the program. Anthony drew out the UML diagram, meaning he needed to focus on what the implementation of the software architecture diagram and its overview would look like. Finally, JJ handled the UML diagram descriptions. This meant specifying the UML diagram to be as clear as possible how the program will finally be implemented.

Restaurant food system

- Payment method
  - Card
    - Debit/credit
  - Apple pay
- Admin and user interface
  - Manager
    - Ability to add/remove menu
    - Modify payment methods accepted
    - Everything employee can do
  - Employee
    - Manually update inventory
    - Everything the customer can do (except pickup/delivery, replace with dine in/takeout)

  - Customer
    - Pickup or delivery
      - Delivery: send it to delivery employee as well. Prompt for location of delivery
    - Order food
      - Send receipt electronically to kitchen
      - Prompt if want to print receipt for customer
      - Customize food order (add/subtract)
      - Update inventory

UML Portion

Sign In

- EmployeeID