FreshSip Juice Bar

Table of Contents

Table of Figures

1. Introduction

“FreshSip” is an online platform designed for a juice bar that offers the online facility to order juices through a system. It has basic functions such as customer registration, ordering items, inventory, billing, and reviewing.

2. Goal

The goal of developing this system is to introduce a platform for the users that provides online ordering of juices from the juice bar, enhancing user experience.

3. Objectives

The objective of FreshSip system is to provide a user-friendly platform that allows customers to browse the menu, make juice selections, and place orders with ease. By ensuring reliability, scalability, and efficiency, the system will improve customer satisfaction and operational efficiency for the juice bar.

4. Functional and Non-Functional Requirements

4.1. User Registration

(K. G. G. N. D. Weerathunga – 11037)

This is one of the main functions in the system. It allows users to register and gain access to various features such as ordering juice, adding reviews, and more. Users can register by submitting a form with their personal details. After registering, they can log in to the system using their user ID and password to perform these tasks. After logging in, customers can access the services provided by the system and update certain details they used during registration, with a few restrictions.

4.1.1. Functional Requirements

- **Customer Registration**: Provides a registration form where customers can enter their personal details, such as name, email address, phone number, and password. Validate the input data (e.g.: checking for a valid email format and ensuring all required fields are filled). The customer’s details are securely stored in the database.

- **Login Functionality**: The system allows registered customers to log in using their user ID (e.g.: email) and password. JWT is used while logging in, deleting, and updating user details. A token is used for Profile management. This authorizes the website. This functionality also authenticates the user by verifying their credentials against the stored data.

- **Edit User Details**: The system allows customers to update certain details, such as their first name, last name, and mobile number, after logging in. Then validates and store the updated information securely.

- **Delete Account**: The system provides an option for customers to delete their account if they no longer wish to use the service. Removes all associated customer data from the database upon account deletion.

- **Session Management**: The system manages user sessions, ensuring that customers remain logged in during active use and are logged out after a period of inactivity.

4.1.2. Non-Functional Requirements

- **Security**: Encrypt sensitive customer information, such as passwords, to protect against unauthorized access.

- **Performance**: Handle multiple customer registrations, logins, and updates simultaneously without significant delays. Provides quick response times for registration, login, and account management actions.

- **Usability**: The registration form and related features (e.g., login, edit details, delete account) to be user-friendly and intuitive. Provides clear error messages for invalid inputs and confirmation messages during registration, login, and account deletion processes.

- **Availability**: Ensure that the registration function is available 24/7, allowing customers to register, log in, update, or delete their accounts anytime and from anywhere.

4.2. Order

(K. D. M. R. Amada – 14633)

Online ordering function mainly provides facility to place orders by all type of customers. To place an order customer must register to the system. when user logs in to the system there are two options. One is user can access online order form from main navigation bar but cannot put their order if they are not registered to the system. Also, every Juice item leads to online order form too. In online order form there are few fields to be filled.

Those fields will save in orders table in the "freshsip" database. After filling the form user can submit the form by clicking on submit button. Then user will be directed to another interface in that interface where the user can see main three buttons. Those are Edit Order, Delete Order, and Done.

- **Edit Order:** After putting an order customers can change their order within a specific period. Within one minute of time if customer has changed their mind, they can change their order by clicking the button. When they click on the button, they will be directed to update order form. It also the same as before. But only Item name, Date and Time, Date fields will be changed. Customer can reselect the Juice items and they can change the quantity too. Date, Date and time will be automatically set by the system. Then customer can submit the update form by clicking on submit button. then they will be directed to edit order, delete order and done page.

- **Delete Order:** Customers can cancel their order within a specific period. Within one minute of time if customer has changed their mind, they can delete their order by clicking the button. When they click the button, it confirms user wants to delete the order. Then customer can either confirm or cancel. If they cancel, order will not be deleted. They return to button page. if they confirm the deletion, order will be deleted from the system, and they return to button page.

- **Done:** After putting an order customers must click on done button. Then it navigates customer to email page. Email page holds some values like order number, customer name, ordered item name, ordered quantity, and order date. Those fields will be filled according to the already placed order details. Customer can go through it and then click on send email button. it will send an email to customer registered email address. It will confirm the order and provide order token to the customer. Order token email holds

some important details to fetch order from cafeteria and its evidence to prove their orders. Also, when they click on send email button it will navigate to home page.

- **Order Stock:** This facility is shown to the Administrator. In this page it shows all orders customers have put so far in descending order according to date. Also, it has a button called pending to every order. when customer fetch the order cashier will click on pending button and update the status to done. Doing this activity, they can identify the people receives the order or not and who paid or not. Also, in order stock function there is another feature called full count it will automatically shows categorically how many orders received according to current date. This order stock will be placed in administrator site.

- **Online Order:** There is another feature called online order. This facility can only be accessed by Administrator. This function has two buttons yes and no. Using these buttons administrator can disable and enable online order function. If administrator clicks “no” then it will block the online ordering form. If administrator enters “yes” button, it enables online order form to customers.

4.2.1. Functional Requirements

- Send emails when order has been placed by the user.
- Show alerts when user submit orders, cancel, and edit their orders.
- User can place their order by choosing items.
- System should provide update and delete options to the user.

4.2.2. Non-Functional Requirements

- **Performance:** Fast selection and fast items loading, and user data loading to the pages.
- **Availability:** System must be available on 24 x7, but the ordering process is only available during the shop opening hours.
- **Security:** For order, it can put by only registered customer, and it checks variety of validation before submitting a form.
- **Usability:** For order function it gives user less chance to manually enter data, which is very comfortable to the user. It is also a very friendly interface design.

4.3. Billing

(D. R. Ravindya – 11497)

Billing function allows the admin to handle billing operations, including adding new bills, updating existing ones, and viewing or deleting past billing records. The admin can generate a new bill by filling out the billing form with details such as the date, time, cart Id, cash, and automatically calculated full total and balance of payment. All past billing records can be accessed. But only the cash (user payment amount) can be edited. Additionally, if a bill is no longer valid or needs to be removed, it can be deleted. This functionality is only available from the admin side.

4.3.1. Functional Requirements

- **Add new bill:** A new bill can be added by filling the fields of the billing form. For validation, all required fields should be filled.
- **View bill records:** Allows the admin to view bill records in a table format. It consists of date, time, total of price, user payment amount and the balance.
- **Update billing details:** Changes can be only done to cash field, preventing unnecessary editing of records. After saving the changes, the database will be updated.
- **Delete bill records:** Allows the admin to delete a bill record if needed. To avoid accidental deletions, a confirmation message will be displayed. Once “ok” is clicked, the record will be deleted from the database.

4.3.2. Non-Functional Requirements

- **Performance**: Handles multiple billing operations simultaneously without lagging along with fast response time.
- **Usability:** Ensures the forms are easy to navigate. Provide clear instructions and error messages when needed.
- **Security:** Only authorized admins can have access to billing operations.
- **Availability:** Ensure that the function is available 24/7 to add, view, update and delete the billing records.

4.4. Inventory

(A. L. L. Wijesiri – 14634)

The Inventory function is crucial for ensuring the juice bar maintains an adequate supply of ingredients and items required for operations. It allows admins to add new stock, update

existing inventory details, view current stock levels, and delete expired items. This feature ensures efficient tracking and management of inventory, avoiding shortages or wastage. Admins can manage inventory through a dedicated interface, which includes functionality to monitor stock quantities, update supplier details, and track inventory movements.

4.4.1. Functional Requirements

- **Add New Stock:** Allows the admin to add new stock items by filling out a form with details like item name, quantity, supplier, purchase date, and expiry date. Validates all input fields (e.g.: ensure quantity is a positive number and expiry date is in the future). Stores the new stock details securely in the database.
- **View Stock Records:** Provides a comprehensive view of all stock items, including key details such as item name, quantity, supplier, and expiry date. Supports search and sorting options to simplify inventory tracking.
- **Update Stock Details:** Enables the admin to update editable fields like quantity, price, or supplier information. Validates updated data and reflect changes in the database immediately.
- **Delete Stock Records:** Allows deletion of stock items that are no longer needed or have expired. Displays a confirmation prompt to avoid accidental deletions.

4.4.2. Non-Functional Requirements

- **Performance:** Handles multiple inventory operations simultaneously without system slowdowns. Ensures quick updates to stock data and real-time notifications for low stock or expiry.
- **Usability:** Designs an intuitive interface for easy navigation and management of stock items. Provides clear feedback for operations like adding, updating, or deleting stock.
- **Security:** Restricts inventory management access to authorized admins. Encrypts sensitive data, such as supplier details, to protect against unauthorized access.
- **Availability:** Ensures the inventory function is operational 24/7 to support round-the-clock management.
- **Data Integrity:** Implements regular backups to prevent data loss and ensure quick recovery in case of system failures.

Overall, this function enables the juice bar to maintain optimal inventory levels, reduce waste, and ensure seamless operations.

4.5. Review

(K. K. N. T. Madushani – 11055)

The review function for the system enables registered users to submit, update, or delete their reviews, while non-registered users can view all submitted comments. The form for submitting has comments and email address fields that require the relevant user details. Submitted reviews are displayed on the same page for easy visibility. Users can manage only their own comments, with updates and deletions processed through the interface that ensures changes are reflected in real-time.

4.5.1. Functional Requirements

- **Add New Review:** Users can submit a new review by filling out the review form with email address and comment text. All fields must be filled correctly for successful adding. The email must follow the proper format (e.g.: user@example.com). Successfully submitted reviews are stored and displayed below the form.
- **View Reviews:** Enables all users to view submitted reviews, regardless of registration status. Displays a list of all reviews with the reviewer's email and comment.
- **Update Review:** Registered users can update their own reviews by clicking the "Update" button next to their comment. Displays an "Update Form" with the current comment pre-filled. Users can modify their comment and save the changes. Changes are applied only to the user's own comments. Updated reviews are stored in the database and reflected on the page.
- **Delete Review:** Registered users can delete their own reviews. A confirmation box ("Are you sure you want to delete this review?") ensures no accidental deletions. Deleted reviews are permanently removed from the database.
- **Confirmation Box:** Upon successful submission of a review, a confirmation box confirms the action. The message automatically after a short time.

4.5.2. Non-Functional Requirements

- **Performance:** Handles multiple review submissions and updates simultaneously without lag. Quickly loads and displays existing reviews, even with many comments.

- **Usability:** The review form should be easy to understand and use. Provides clear instructions and error messages for incomplete or invalid inputs.
- **Security**: Encrypts sensitive user information, such as email addresses. Ensures only authorized users can manage (update/delete) their own reviews.
- **Availability:** The review system should be accessible 24/7 for submitting, viewing, and managing reviews.
- **Data Integrity:** Ensures all reviews are accurately stored and retrieved.
- **Error Handling:** Displays user-friendly error messages if submission fails.
- **Scalability:** The system should efficiently handle an increasing number of users and reviews without performance degradation.

5. Use Case Diagram

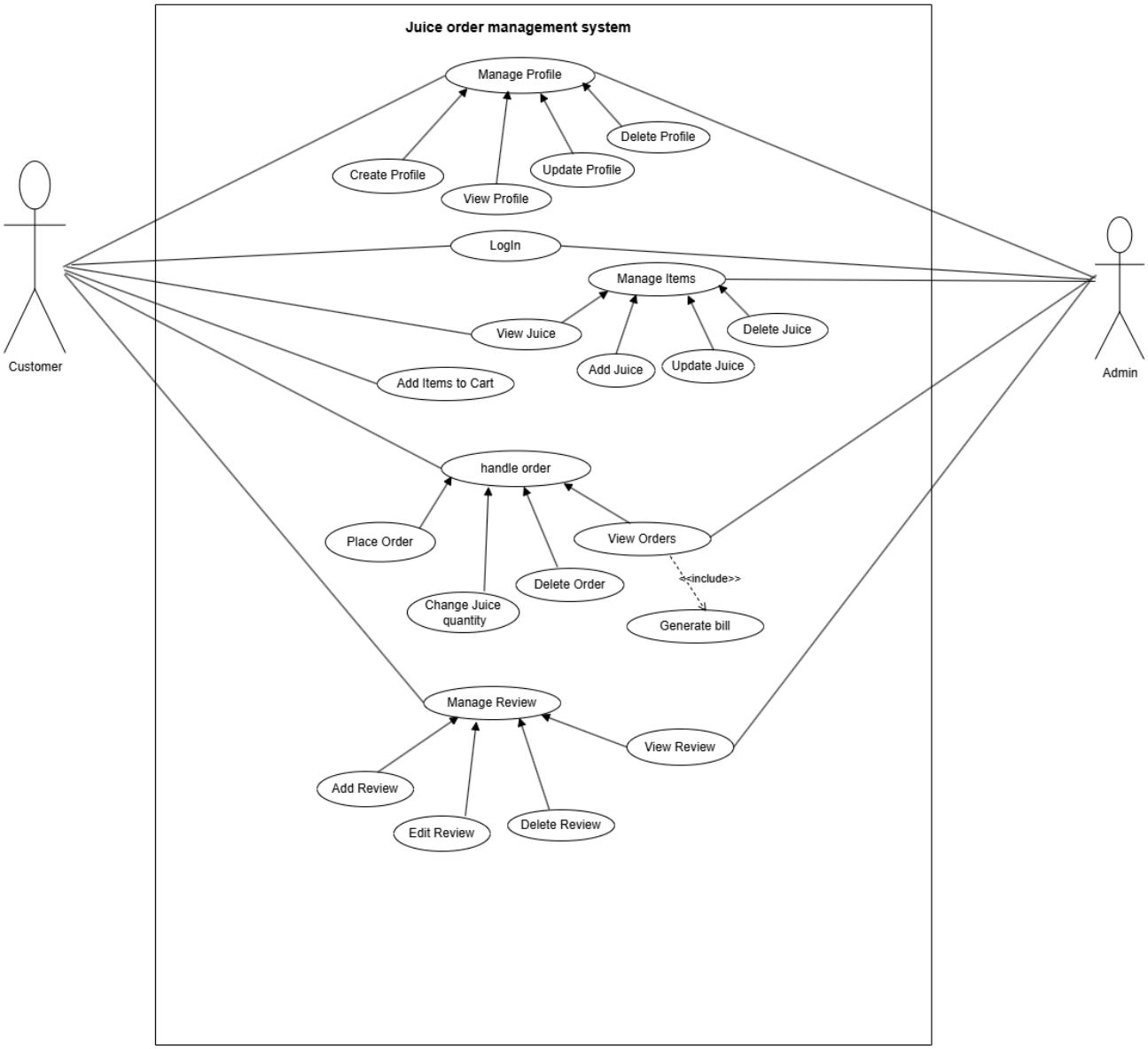

Figure 1. Use Case Diagram for the Entire System

6. Entity Diagram

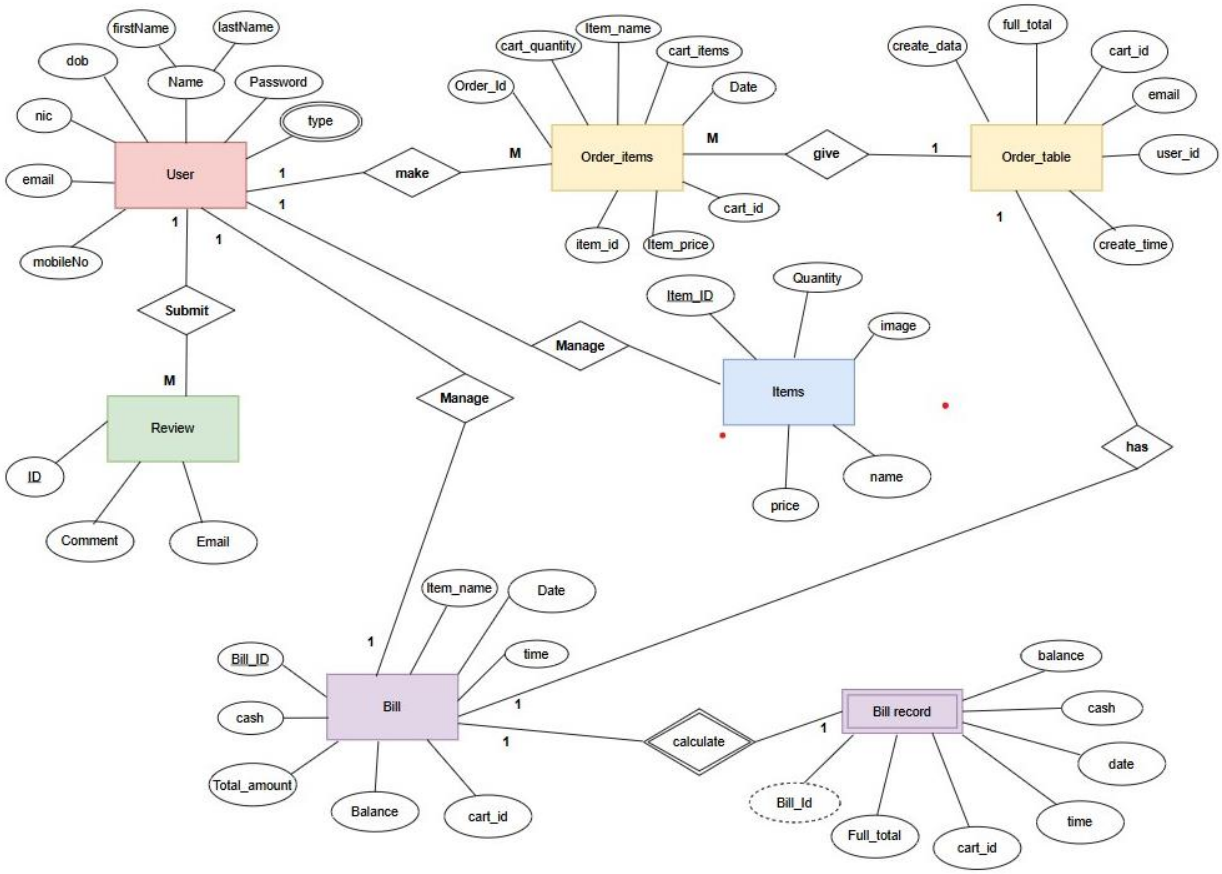

Figure 2. Entity Diagram for the Entire System

7. Architectural Pattern

Layered Architecture is the most suitable architectural pattern for FreshSip due to its clear separation of concerns and modular approach, which makes the system easier to manage and maintain. Since this architecture is divided into Presentation Layer, Application Layer, Data Access Layer, and Database Layer with each later being responsible for specific tasks. Each layer's interactions with the layers beneath it promotes better organization and clear responsibilities across the system.

- Presentation Layer: Responsible for handling user interactions, such as customer registration, ordering items, reviewing, and billing. It manages the interface that users interact wit, ensuring quality user experience.
- Application Layer: Contains core business logics, such as order processing, validating reviews, managing inventory, and handling payment transactions. This ensures that updates and deletions can happen without affecting the user interface or database.
- Data Access Layer: Supports communication between the application logic and the database. It ensures secure and consistent data management without complicating the interactions with the database.
- Database Layer: Stores all essential data such as customer details, order history, reviews, and inventory ensuring data integrity and scalability, that are important in smooth operation.

Why this architecture pattern is the most suitable for FreshSip is because it supports scalability and flexibility. Considering how the system might grow with new features, for example, adding promotions can be done in the Application Layer without having to modify user interface. Layered Architecture also enhances security and maintainability. Therefore, this architectural pattern fits FreshSip the most.

8. High Level Component Diagram

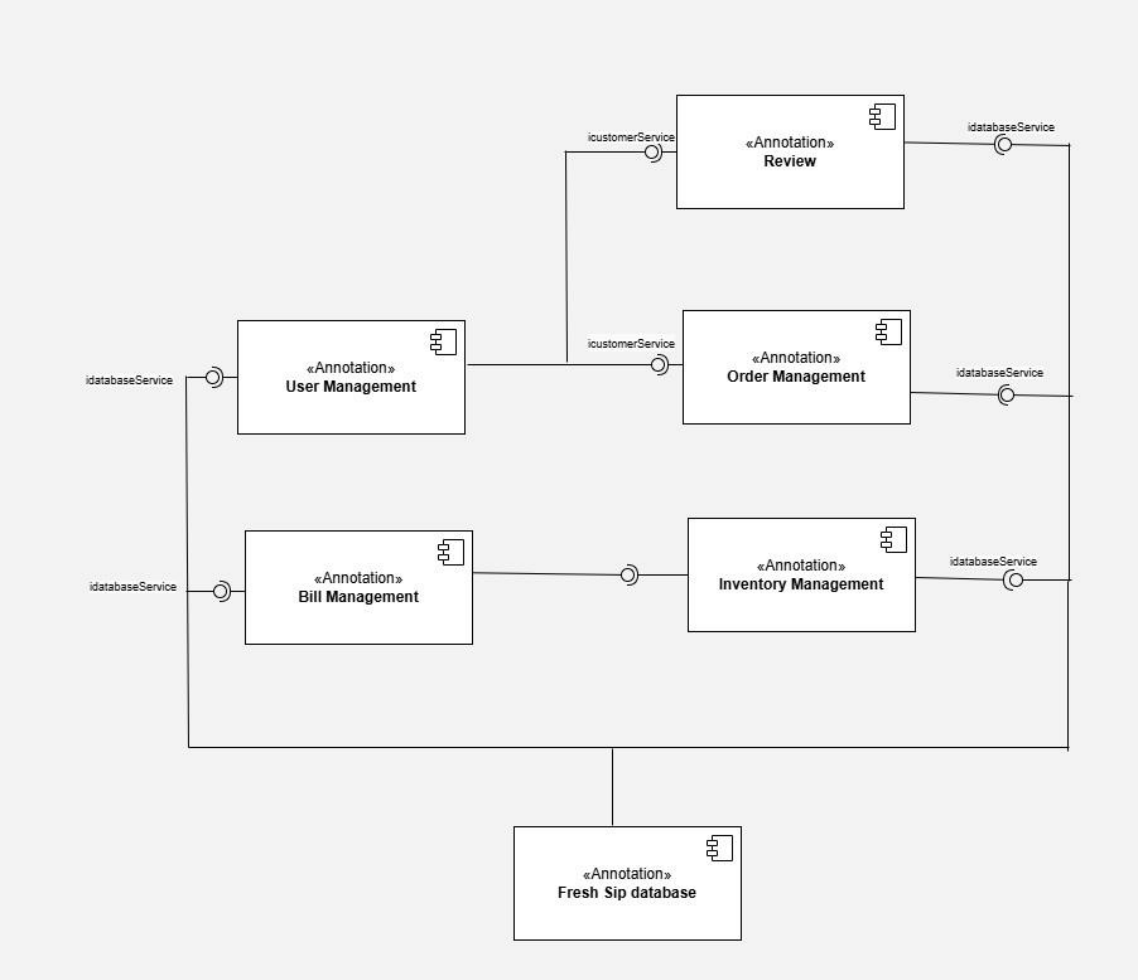

Figure 3. High Level Component Diagram for Entire System

9. Component Diagrams

9.1. User Registration

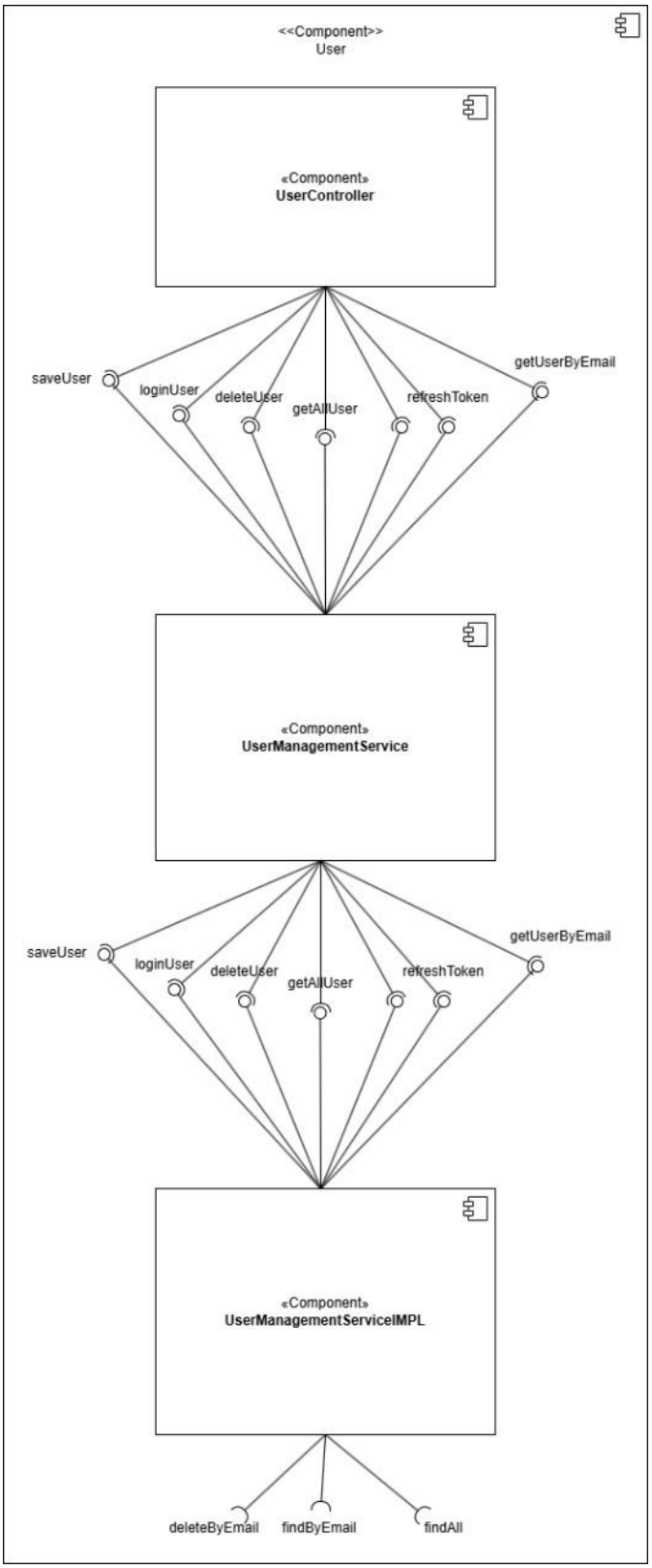

Figure 4. Component Diagram for User Registratio

9.2. Order

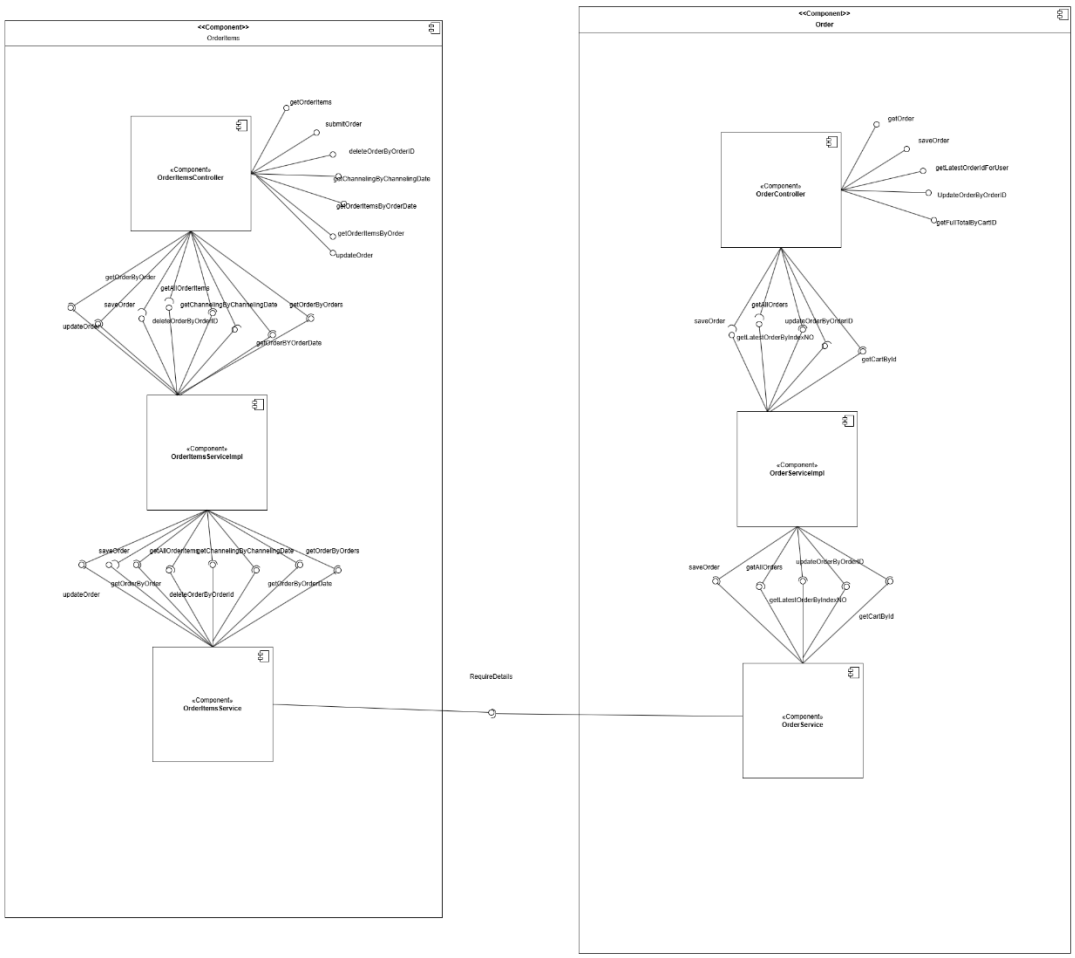

Figure 5. Component Diagram for Order

9.3. Billing

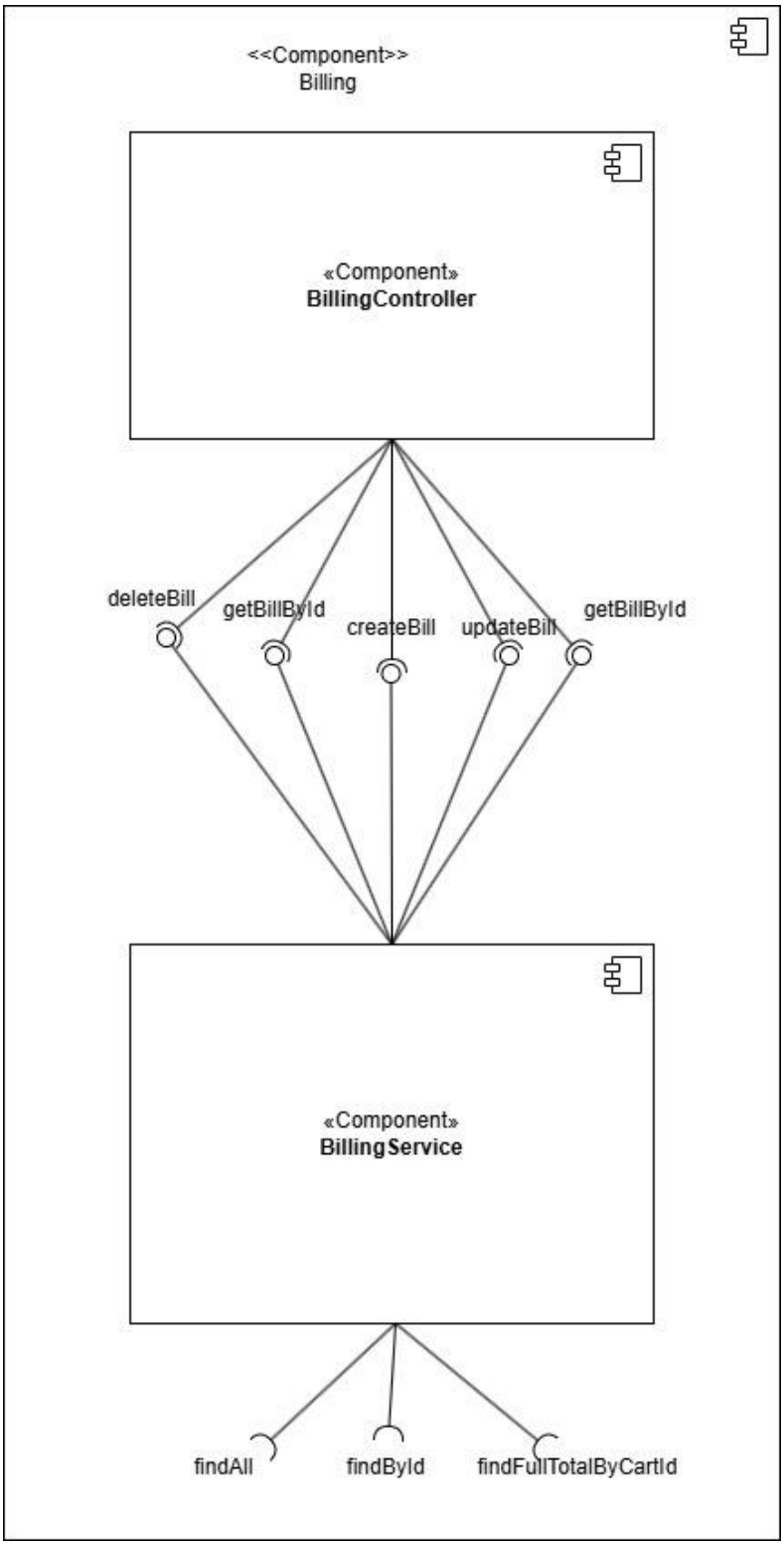

Figure 6. Component Diagram for Billing

9.3. Inventory

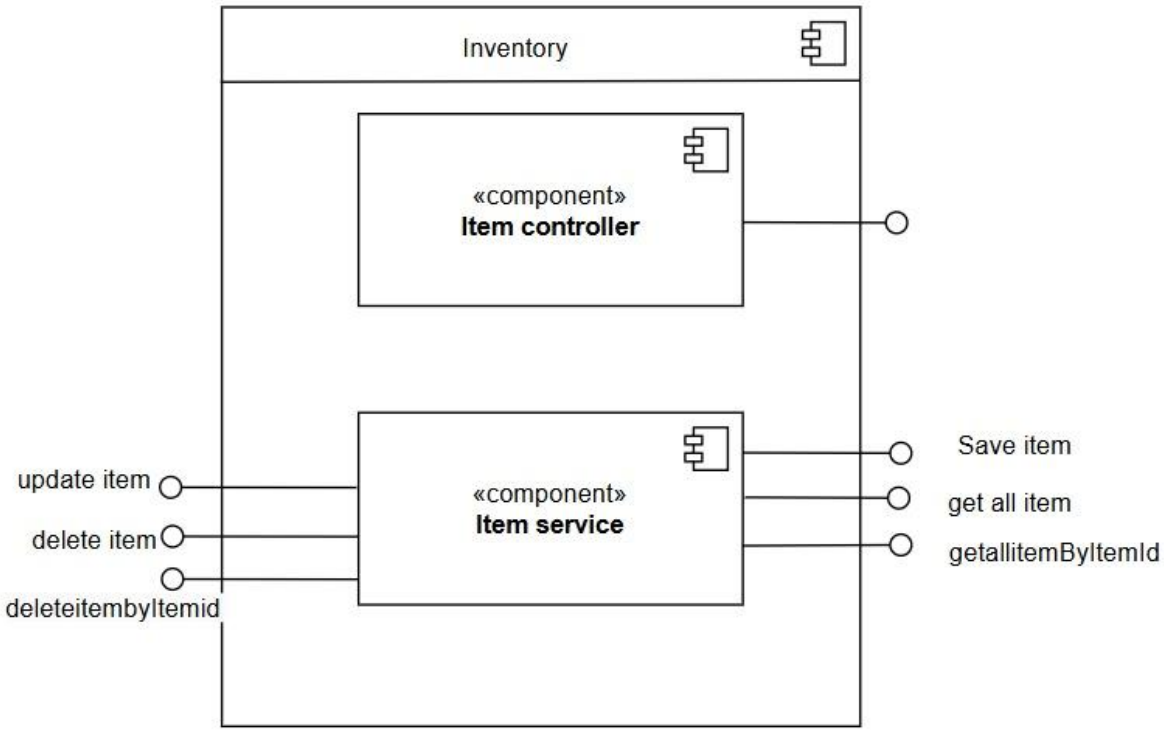

Figure 7. Component Diagram for Inventory

9.4. Review

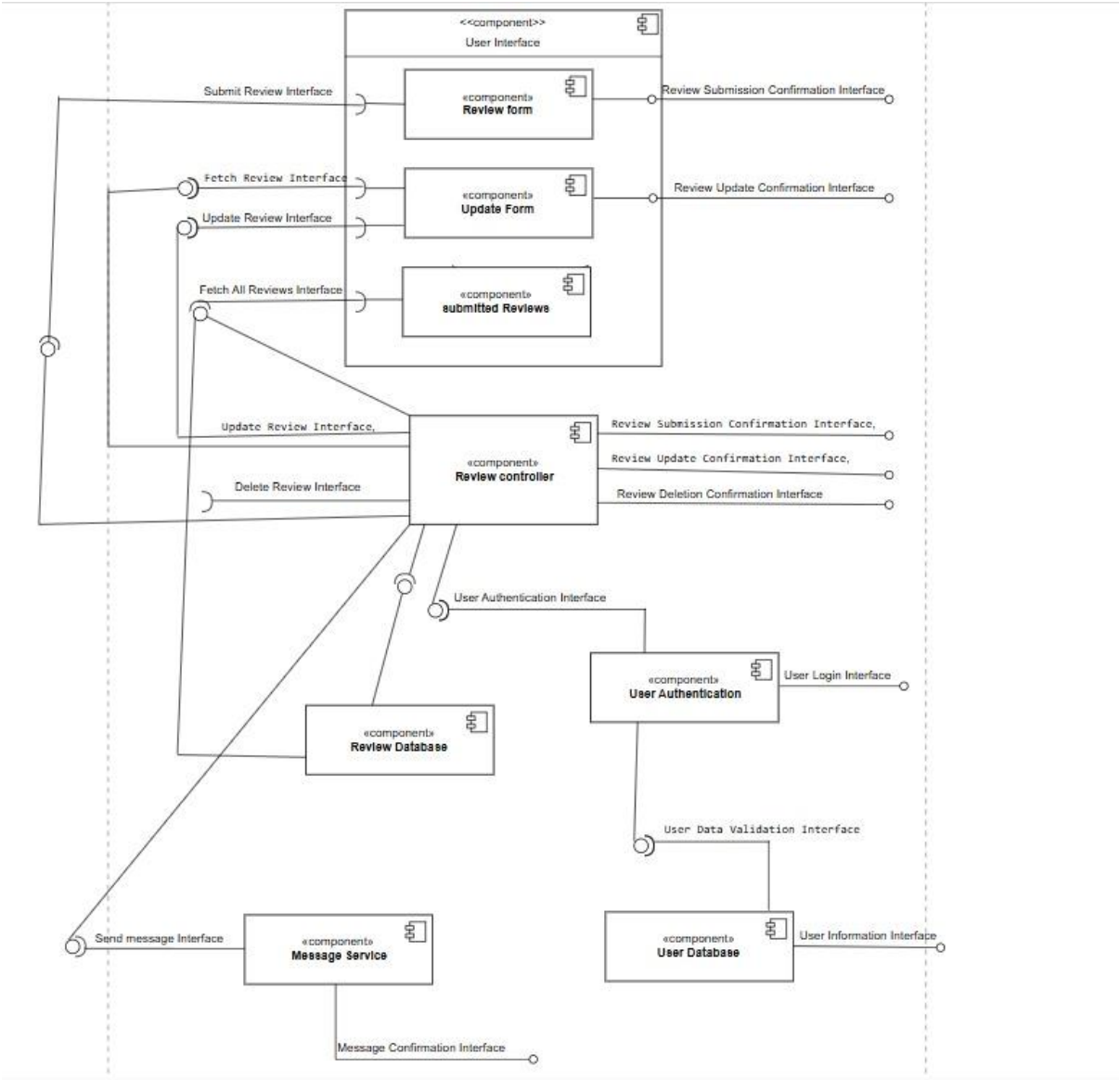

Figure 8. Component Diagram for Review

10. Sequence Diagrams

10.1. User Registration

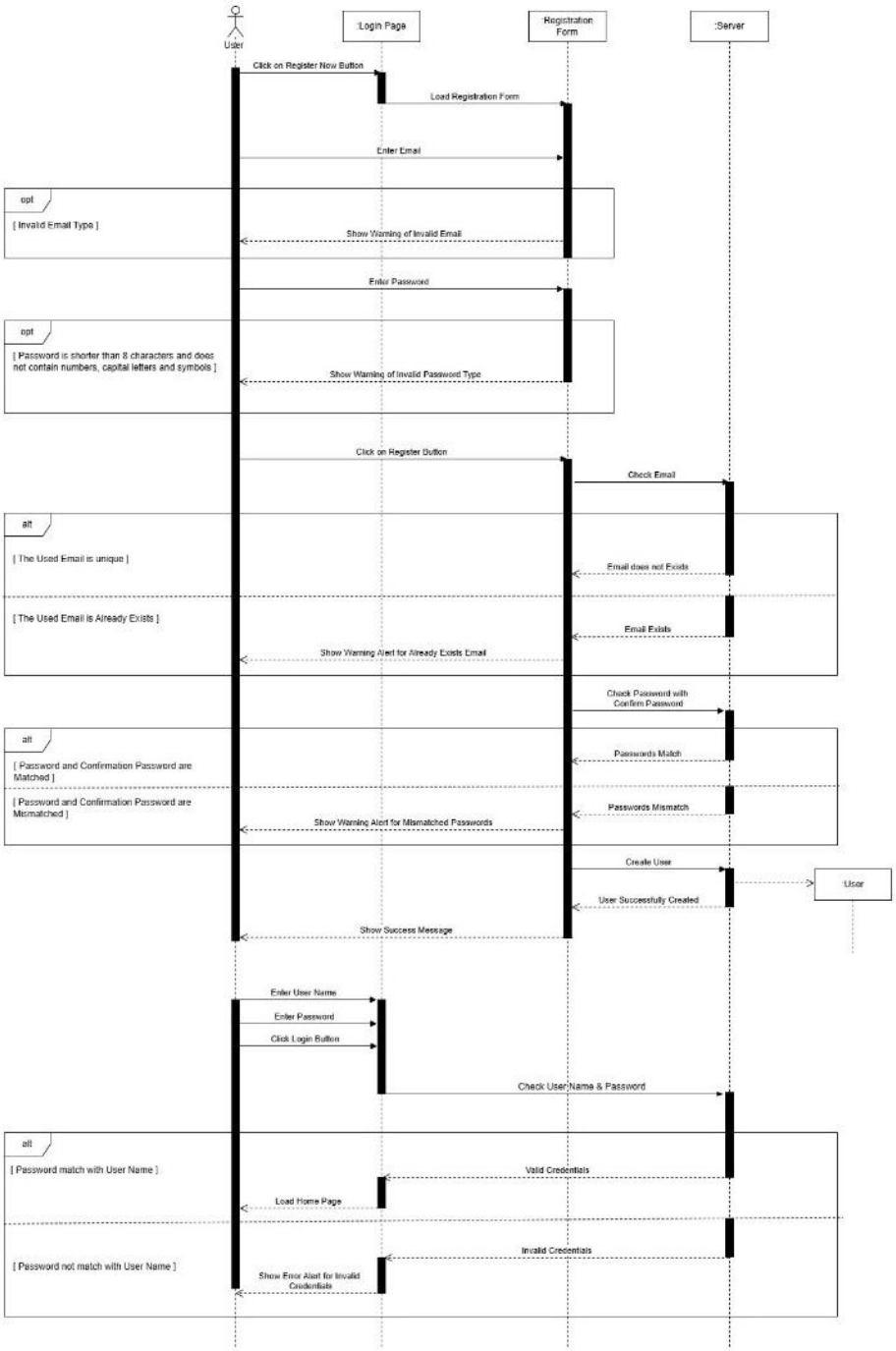

Figure 9. Sequence Diagram for User Registration

10.2. Order

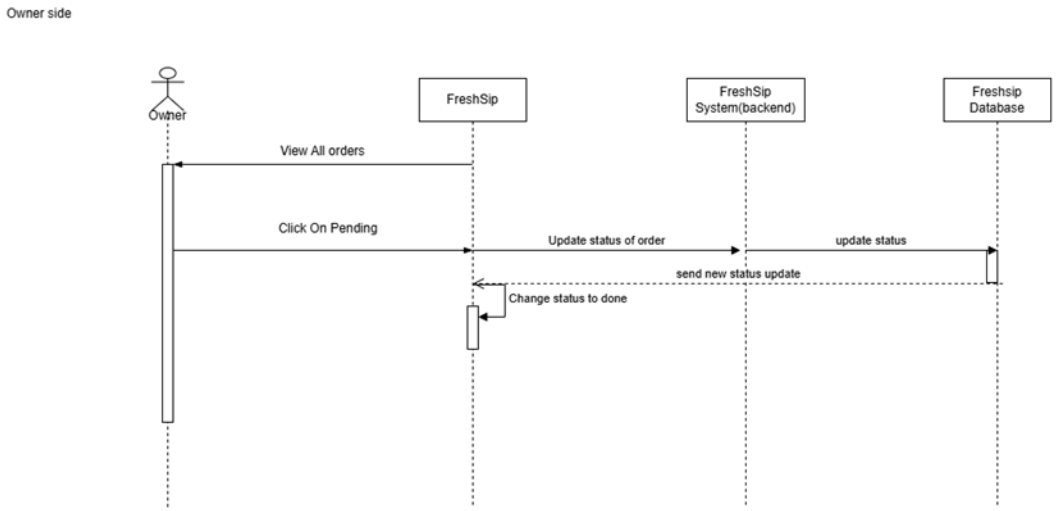

Figure 10.Sequence Diagram for Order(1)

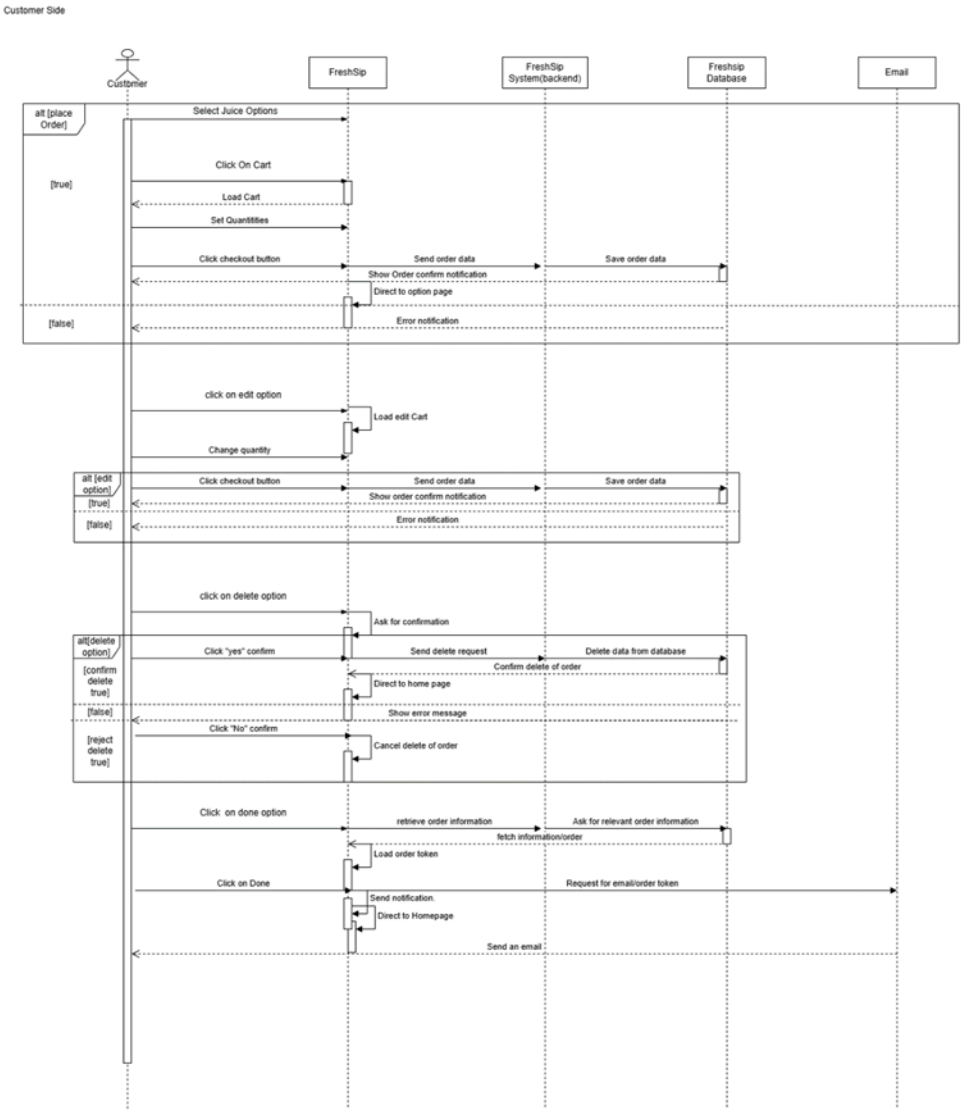

Figure 11. Sequence Diagram for Order(2)

10.3. Billing

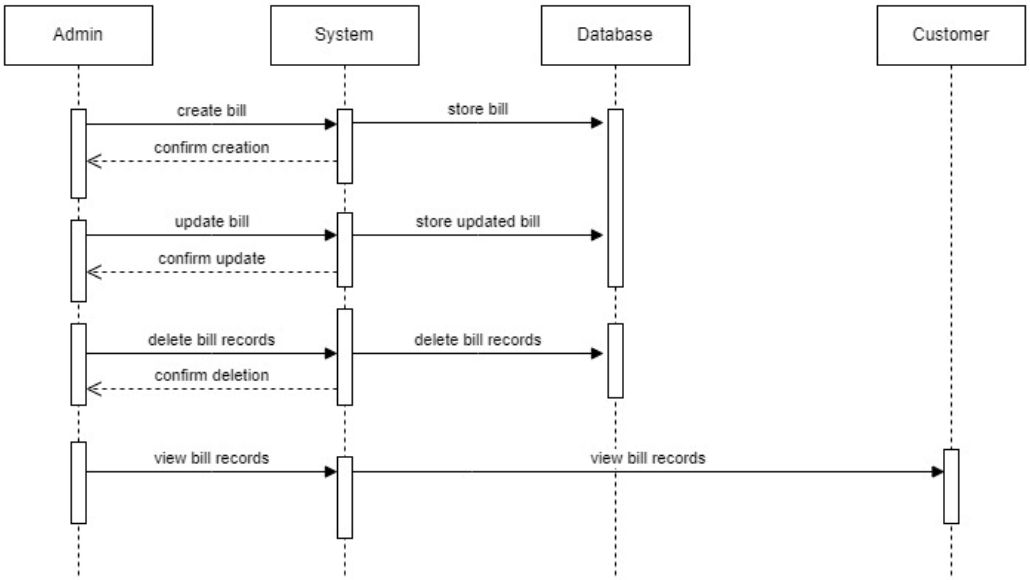

Figure 12. Sequence Diagram for Billing

10.4. Inventory

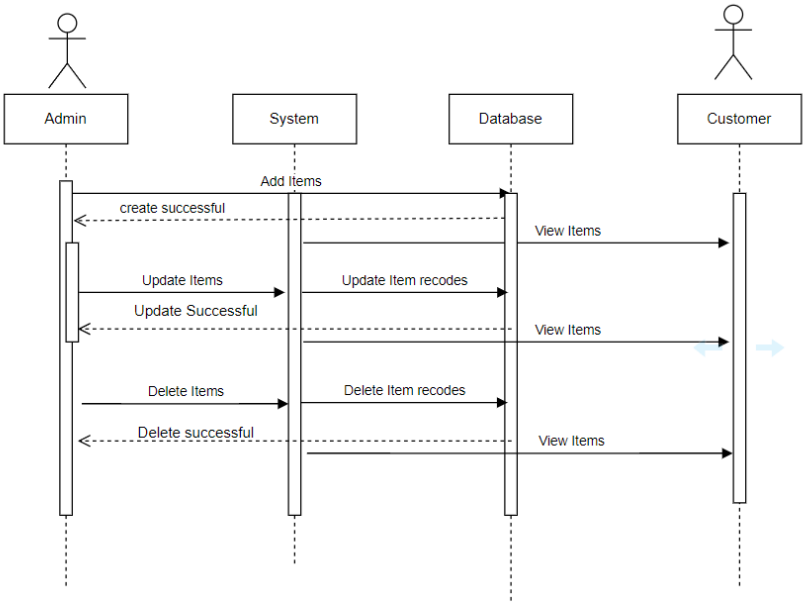

Figure 13. Sequence Diagram for Inventory

10.5. Review

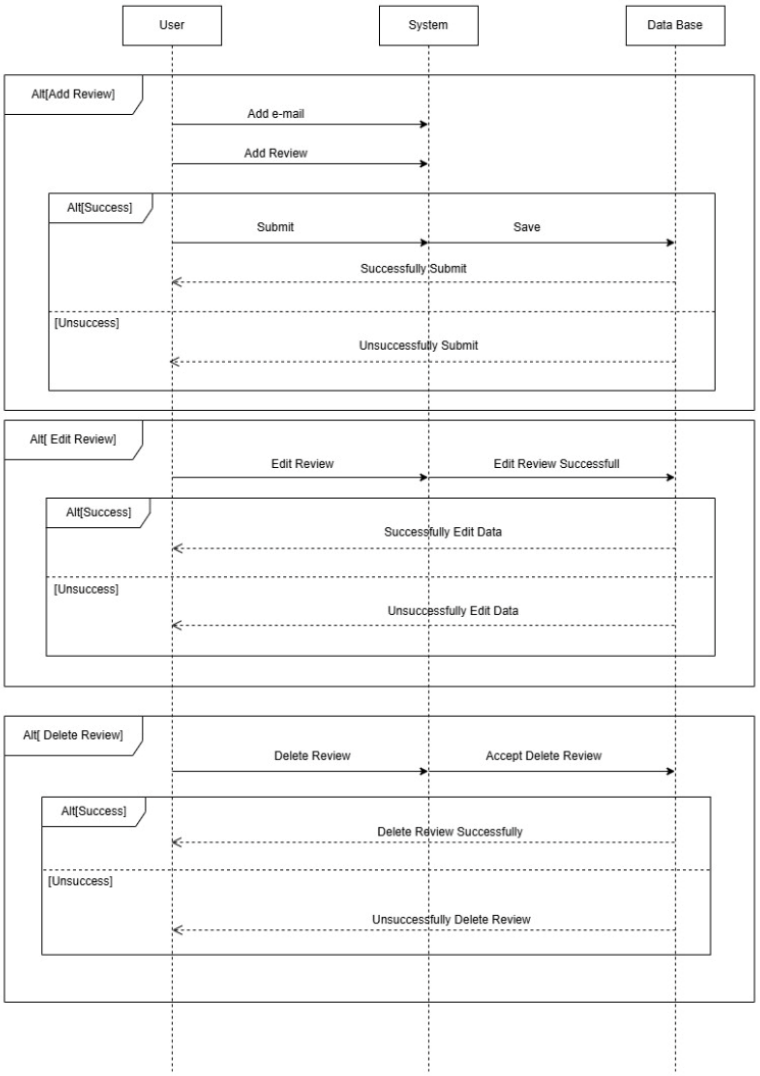

Figure 14. Sequence Diagram for Review

11. Design Pattern Usage

11.1. User Registration

- Singleton Pattern: The JWTUtils class appears to follow the Singleton design pattern. This is because it initializes the SecretKey object (key) once, and the same instance of JWTUtils is likely used throughout the application (as it's annotated with @Component for Spring's dependency injection).

- Factory Method Pattern: The SecurityConfig class configures the authentication provider using authenticationProvider() and sets the UserDetailsService (userServiceImpl) and PasswordEncoder (passwordEncoder()) in the DaoAuthenticationProvider. This can be seen as a Factory Method Pattern where the configuration of security beans is abstracted to provide different security mechanisms based on the requirements.

- Template Method Pattern: The UserServiceImpl class implements the UserDetailsService interface. In a way, this can be seen as implementing a Template Method Pattern, where the core logic of loading user details is provided by the UserDetailsService interface, but UserServiceImpl is expected to implement the method loadUserByUsername. The parent interface provides the skeleton of the algorithm, while the subclass (UserServiceImpl) defines the implementation.

11.2. Order

- MVC Pattern: The code effectively uses several design patterns to create a clean, maintainable, and scalable architecture. It follows the MVC pattern, with OrderController managing HTTP requests and delegating business logic to the service layer. The Service pattern is implemented in OrderService to encapsulate core logic, while the Repository pattern in OrderRepository abstracts database interactions, ensuring separation of concerns. The DTO pattern is used for secure data transfer between layers, and Entity mapping defines the database structure with JPA annotations. Additionally, Dependency Injection ensures loosely coupled components, and Spring's Singleton pattern optimizes resource usage by managing beans' lifecycle. These patterns collectively enable efficient data handling, testability, and adherence to modern software design principles.

- Controller Pattern: This Component utilizes several design patterns to achieve a clean, modular architecture. The Controller pattern in OrderController handles HTTP requests, delegating business logic to the Service layer in OrderService, which centralizes processes and interacts with the Repository layer in OrderRepository, abstracting database operations using Spring Data JPA. The DTO pattern in OrderDTO ensures secure and simplified data transfer, while the Entity pattern in Order models database tables with JPA annotations. Dependency Injection and Factory patterns promote loose coupling and efficient bean management via Spring's IoC container.

- Singleton Pattern: ensures single-instance beans like OrderService, while the Builder pattern via ModelMapper facilitates object mapping. Additionally, Template Method in JPA repositories provides reusable database operations, and Spring AOP's Proxy pattern manages cross-cutting concerns like transactions. Together, these patterns enhance maintainability, scalability, and testability of the application.

11.3. Billing

- Singleton Pattern: In a Spring Boot application, beans managed by the Spring container are singleton by default. This ensures that only one instance of each bean exists in the application context. The BillingService, BillingRepo, OrderItemsRepo, and ModelMapper are singleton beans because they are annotated with @Service or @Repository. Spring ensures that only one instance of these classes is created and reused throughout the application. Spring Context: Handles the singleton behavior automatically.

- Factory Pattern: The conversion between Billing entity and BillingDTO is abstracted out using ModelMapper. The ModelMapper object acts like a Factory, providing the logic to create DTOs from entities without exposing the creation details.

- Strategy Pattern: for handling different strategies for calculating the fullTotal or managing different types of billing.

11.4. Inventory

- Model-View-Controller (MVC): Controller (ItemController) handles HTTP requests. Service (ItemServices) manages business logic. Repository (ItemRepo) handles database operations.

- Data Transfer Object (DTO): ItemDTO transfers data between layers, keeping the domain model (ItemStock) secure.
- Repository Pattern: ItemRepo abstracts data access, making it easy to interact with the database.
- Singleton Pattern: Spring's dependency management ensures single instances of services and repositories.
- Dependency Injection (DI): @Autowired injects dependencies, promoting loose coupling.
- Adapter Pattern: ModelMapper converts between ItemDTO and ItemStock.
- Observer Pattern (Logging): Logger in ItemController observes and logs method activity.
- These patterns enhance maintainability, scalability, and security in this function.

11.5. Review

1. Service Layer Pattern:

- **Purpose**: Separates business logic from controllers and repositories.
- **Benefits**:
    - **Separation of concerns**: Keeps controllers focused on HTTP handling, while the service layer manages the business logic.
    - **Reusability**: Centralizes business logic, allowing it to be reused across different parts of the application.
    - **Maintainability**: Makes it easier to manage and test the logic.
    - **Transactional control**: Ensures data consistency, especially for operations like adding, updating, and deleting reviews.

2. DTO (Data Transfer Object) Pattern:

- **Purpose**: Transports data between layers (e.g., between the controller and service) without exposing internal entity details.
- **Benefits**:
    - **Decouples layers**: Prevents direct exposure of entity structure to the client.
    - **Optimized data transfer**: Only the necessary data is transferred, reducing overhead.

    - **Flexibility**: API response data can be structured differently from the internal database schema.
    - **Validation**: Allows input data validation before saving to the database.

Why These Patterns Together:

- **Service Layer + DTOs**: They work together to provide a clear separation of concerns and flexible data handling.
    - The **controller** handles HTTP requests.
    - The **service layer** manages business logic.
    - The **DTOs** ensure data is transferred efficiently and safely between layers.
- As a summary,
- The **Service Layer** and **DTO** patterns make my system **modular**, **maintainable**, **testable**, and **scalable**. They help in organizing my code while separating different responsibilities, leading to a cleaner and more efficient design.

12. Achieving Quality Attributes

12.1. User Registration

- Availability: The user registration function is designed to be operational 24/7, ensuring that users can register, log in, update, delete their accounts at any time. Session management ensures smooth transitions and logs out inactive users, contributing to system's security.
- Interoperability: System cleanly integrates with other features, for example, linking user profiles across orders or reviews modules through secure APIs. This keeps the data consistent and accessible across all interconnected functions.
- Modifiability: It is easier to update or improve features in the future, such as adding new fields to registration form, without affecting other parts of the system.
- Performance: This is ensured by optimizing database queries and implementing data validation methods. User registration function is capable of handling concurrent user registrations and logins without major delays. This enhances responsive user experience.
- Security: Sensitive user information, such as passwords, is encrypted using encryption protocols. JWT tokens are used for secure authentication during login, deletion, and updates. Input validations prevent vulnerabilities.

12.2. Order

- Availability: The order function ensures the system is operational 24/7, allowing customers to browse items and place orders. Admin features such as order stock management online ordering availability provide flexibility while maintaining the system availability.
- Interoperability: Order function smoothly integrates with email notifications and database. Using order submission, details are automatically stored in the database and an email confirmation with an order token is sent to the customer's registered email.
- Modifiability: This is achieved by enabling updated or improvements without affecting other components. For example, the ability to enable or disable online orders via admin shows the flexibility to adapt to changes.
- Performance: Uses optimized database queries to ensure fast loading of items and user data. Alerts for submissions, cancellation, and updates are done instantly, providing a smooth and responsive experience.
- Security: Security is ensured by restricting order placement to registered customers and implementing input validations to prevent harmful acts. Email confirmation with an order token adds an extra layer of verification, ensuring order authenticity.

12.3. Billing

- Availability: The billing function is designed to be available 24/7, ensuring that admins can add, view, update, and delete billing records at any time they are logged in. The function is also developed to remain operational even during the periods of high traffic or minor system failures.
- Interoperability: The billing function works in harmony with other related components. Billing process interacts with ordering process to ensure that billing details are consistent with the customer's order. For example, when a customer places an order, cart Id and total price are generated and stored within the bill_records table.
- Modifiability: Admins can easily modify the billing records when necessary. For example, the ability to update only the cash field ensures that only authorized changes

are made to the bill while maintaining the integrity of the other details. Since billing data are critical, every field shouldn't be updated. If any changes are to be done to the format or features, the system can adjust without significant disruption.

- Performance: Multiple billing operations can happen concurrently without lagging. The system ensures multiple billing updates can happen without compromising response time. This also ensures fast processing of billing operations, such as updating a bill without a delay.
- Security: The billing function provides security by limiting access to only authorized users (admins). Through the authentication mechanisms done at the login, only the admins with correct permission can modify billing records.

12.4. Inventory

- Availability: The system is reliable on cloud infrastructure, ensuring that inventory function is accessible to admins 24/7. Redundant servers and failover mechanisms are in place to make the system run smoothly even if a hardware or software issue might occur.
- Interoperability: RESTful APIs are used for smooth communication between the inventory component and other components like billing. It adapts data from JASON for easy compatibility with third-party systems. Due to its extensibility future changes to the modules can be done without requiring major architectural changes.
- Modifiability: Gives opportunity to update specific fields such as item names, price or quantity, without impacting other parts of the system. Hence, future updates can be efficiently managed.
- Performance: Inventory is designed for optimal performance with efficient database queries to reduce the time needed for operations such as adding records, viewing, updating or deleting records.
- Security: Ensures security through RBAC (Role-Based Access Control) that restricts inventory operations to unauthorized users. Sensitive data is encrypted using protocols like AES for data at rest and HTTPS for data in transit to prevent unauthorized access. User inputs are validated to prevent security vulnerabilities. Admin sessions are monitored as well.

12.5. Review

- Availability: The review function is accessible 24/7, allowing users to submit, view, and manage their reviews anytime. The feedback can be captured without interruptions. The function ensures user access for both registered and non-registered users.
- Interoperability: The review function smoothly integrates with other components, such as user, to verify the ownership of reviews. It also supports standardize data formats that allows reviews to be displayed clearly.
- Modifiability: Enables quick and efficient updates. For example, adding new fields like rating to the form can be implemented without disrupting existing functionalities.
- Performance: Improved database queries ensure that reviews load and display quickly even with large number of comments. Updates and deletions proceed without any lag. The system can handle multiple concurrent review submissions without slowing down.
- Security: Sensitive information such as email address is encrypted to protect user data. Role-based access control ensure only the registered users can edit or delete their own reviews. Input validations protect the function from malicious activities.

13.Youtube Link

https://youtu.be/yNxXQYdKvPI

14. Conclusion

The "FreshSip Juice Bar" system provides an innovative and user-friendly platform designed to enhance the efficiency and customer experience of ordering juices online. By integrating core functionalities such as user registration, online ordering, billing, inventory management, and reviews, it ensures a reliable service to the users. Each module is carefully designed to meet functional requirements and non-functional attributes like performance, security, and usability. The system’s architecture and design patterns ensure scalability, making it adaptable to future growth or feature additions. FreshSip offers a dependable solution for both customers and administrators, addressing critical needs such as order management, billing accuracy, and inventory tracking. This comprehensive approach to service delivery simplifies operations and improves customer satisfaction.

Leader Details:

14634

A.L.L.Wijesiri

0716181138

lihinilochanana643@gmail.com

Group Member Details:

1. K. D. M. R. Amada - 14633

2. K. G. G. N. D. Weerathunga - 11037

3. D. R. Ravindya - 11497

4. K. K. N. T. Madhushani – 11055

GitHub Link:

https://github.com/lihinilochana/FreshSip.git

https://github.com/lihinilochana/FreshSip/tree/ordering