**Hong Kong Institute of Vocational Education**

**Department of Information Technology**

**Higher Diploma in Software Engineering (IT114105)**

**Enterprise Systems Development (ITP4511)**

**Assignment Report**

**(2024/2025)**

**Fruit Management System**

|  |  |
| --- | --- |
| Student: | Sham Hoi Kin 230138439 |

We declare that this is a group project and that no part of this submission has been copied from any other student’s work or from any other source except where due acknowledgement is made explicitly in the text, nor has any part been written for us by another person.

|  |  |  |
| --- | --- | --- |
| **Student** | **Contribution to the project (%)**  **(Total 100%)** | **Signature** |
| Sham Hoi Kin | 100 | Ronald |
|  |  |  |
|  |  |  |
|  |  |  |

Content Page

[1 Assumption and The User and System Requirements 3](#_Toc196580122)

[1.1 Assumptions 3](#_Toc196580123)

[1.2 User Requirements 3](#_Toc196580124)

[1.3 System Requirements 4](#_Toc196580125)

[2 Site Map 5](#_Toc196580126)

[3 System Structure: MVC Model 6](#_Toc196580127)

[3.1 Model 6](#_Toc196580128)

[3.2 View 6](#_Toc196580129)

[3.3 Controller 6](#_Toc196580130)

[4 Database Structure 7](#_Toc196580131)

[4.1 ER-Diagram 7](#_Toc196580132)

[4.2 Data Description 8](#_Toc196580133)

[4.2.1 Users 8](#_Toc196580134)

[4.2.2 Shops 9](#_Toc196580135)

[4.2.3 Warehouses 10](#_Toc196580136)

[4.2.4 Borrowing 11](#_Toc196580137)

[4.2.5 Reservations 12](#_Toc196580138)

[4.2.6 Deliveries 13](#_Toc196580139)

[4.2.7 Fruits 14](#_Toc196580140)

[4.2.8 Stock 15](#_Toc196580141)

[4.2.9 Consumption 16](#_Toc196580142)

[4.2.10 Cities 17](#_Toc196580143)

[4.2.11 Countries 18](#_Toc196580144)

[4.3 Relationships 19](#_Toc196580145)

[5 Brief Description 20](#_Toc196580146)

[5.1 Major Characteristics 20](#_Toc196580147)

[5.2 Dsign Highlights 21](#_Toc196580148)

[6 Project Schedule 23](#_Toc196580149)

[7 Conclusions 24](#_Toc196580150)

[8 Skill Checklist 25](#_Toc196580151)

# Assumption and The User and System Requirements

# Assumptions

The system is designed for a bakery chain with multiple shops and warehouses.

Users are categorized into three roles: Bakery Shop Staff, Warehouse Staff, and Senior Management.

The system is accessed via a web interface, and all data is stored in a MySQL database.

The system supports CRUD operations for managing fruits, users, reservations, and stock levels.

# User Requirements

Bakery Shop Staff:

* Borrow fruits from other shops.
* Reserve fruits from warehouses.
* View stock levels and consumption records.

Warehouse Staff:

* Approve or reject reservations.
* Arrange deliveries to bakery shops or other warehouses.
* Update stock levels in the warehouse.

Senior Management:

* Manage users and fruit types.
* View reserve needs and consumption records.

# System Requirements

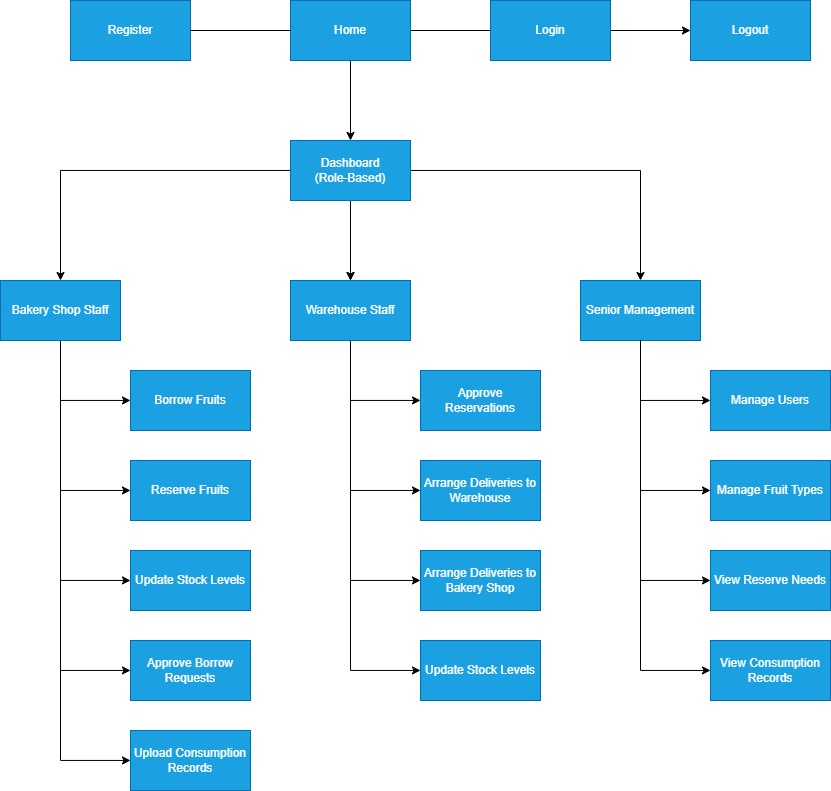
The system must support role-based access control.

The system must provide real-time updates for stock levels, reservations, and borrow requests.

The system must generate reports for consumption and reserve needs.

The system must be responsive and accessible on desktop and mobile devices.

# Site Map



# System Structure: MVC Model

# Model

* Represents the database structure and business logic.
* Includes classes like User, Fruit, Borrow, Reservation, Delivery, and Stock.
* Handles database interactions using DAOs (Data Access Objects) such as BorrowDB, FruitDB, and ReservationDB.

# View

* JSP files render the user interface.
* Located in folders like bakeryShopStaff, warehouseStaff, and seniorManagement.
* Bootstrap is used for responsive design.
* Views are role-specific, displaying only relevant actions and data.

# Controller

* Servlets handle user requests and route them to the appropriate model and view.
* Examples:
  + BorrowFruitsHandler: Handles borrow requests.
  + ReserveFruitsHandler: Manages fruit reservations.
  + AdminHandler: Manages users and fruit types.

# Database Structure

# ER-Diagram

一張含有 文字, 螢幕擷取畫面, 圖表, 字型 的圖片

AI 產生的內容可能不正確。

# Data Description

# Users

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Name** | **Type** | **Description** | **Null?** |
| PK | User\_id | int | User ID number, generate from the program | Not null |
|  | Username | varChar(50) | User’s username | Not null |
|  | Email | varChar(50) | User’s email | Not null |
|  | Password | varChar(50) | User’s password | Not null |
|  | Role | enum | User’s role (‘BakeryShopStaff’,  ’WarehouseStaff’,  ’Seniormanagement’). | Not null |
|  | Shop\_id | int | User’s gender (‘Male’, ‘Female’, ‘Other’) | Null |
|  | Warehouse\_id | int | User’s birth of date | Null |

# Shops

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Name** | **Type** | **Description** | **Null?** |
| PK | Shop\_id | Int | Shop ID number, generate from the program | Not null |
|  | Shop\_name | varChar(255) | Shop’s name | Not null |
| FK | City\_id | Int | City ID number, generate from the program | Not null |

# Warehouses

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Name** | **Type** | **Description** | **Null?** |
| PK | Warehouse\_id | Int | Warehouse ID number, generate from the program | Not null |
|  | warehouse\_name | varChar(255) | Warehouse’s name | Not null |
| FK | Country\_id | Int | Country ID number, generate from the program | Not null |

# Borrowing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Name** | **Type** | **Description** | **Null?** |
| PK | borrow\_id | int | Borrow ID number, generate from the program | Not null |
|  | Form\_shop\_id | Int | shop ID number, generate from the program | Not null |
|  | To\_shop\_id | Int | shop ID number, generate from the program | Not null |
|  | Fruit\_id | Int | Fruit ID number, generate from the program | Not null |
|  | Quantity | int | Borrow’s fruit quantity | Not null |
|  | Borrow\_date | Date | Borrow’s date | Not null |
|  | status | Enum | Borrow’s status (‘Pending’, ‘Approved’, ‘Rejected’) | Not null |

# Reservations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Name** | **Type** | **Description** | **Null?** |
| PK | reservation\_id | int | Reservation ID number, generate from the program | Not null |
|  | shop\_id | Int | shop ID number, generate from the program | Not null |
|  | warehouse\_id | Int | Warehouse ID number, generate from the program | Not null |
|  | Fruit\_id | Int | Fruit ID number, generate from the program | Not null |
|  | Quantity | int | Reservation’s fruit quantity | Not null |
|  | reservation\_date | Date | Reservation’s date | Not null |
|  | status | Enum | Reservation’s status (‘Pending’, ‘Approved’, ‘Rejected’, ‘In Transit’, ‘Delivered’) | Not null |

# Deliveries

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Name** | **Type** | **Description** | **Null?** |
| PK | delivery\_id | int | Delivery ID number, generate from the program | Not null |
|  | reservation\_id | Int | Reservation ID number, generate from the program | Not null |
|  | fruit\_id | Int | Fruit ID number, generate from the program | Not null |
|  | From\_warehouse\_id | Int | warehouse ID number, generate from the program | Not null |
|  | To\_warehouse\_id | int | warehouse ID number, generate from the program | Null |
|  | To\_shop\_id | int | Shop ID number, generate from the program | Null |
|  | quantity | int | Delivery’s fruit quantity | Not null |
|  | Delivery\_date | Date | Delivery’s date | Not null |
|  | status | Enum | Delivery’s status (‘In Transit’, ‘Delivered’) | Not null |

# Fruits

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Name** | **Type** | **Description** | **Null?** |
| PK | Fruit\_id | Int | Fruit ID number, generate from the program | Not null |
|  | Fruit\_name | varChar(255) | Fruit’s name | Not null |
|  | Source\_location | Int | Fruit’s source location | Not null |

# Stock

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Name** | **Type** | **Description** | **Null?** |
| PK | Stock\_id | Int | Stock ID number, generate from the program | Not null |
| FK | Fruit\_id | Int | Fruit ID number, generate from the program | Not null |
| FK | Shop\_id | Int | Shop ID number, generate from the program | Not null |
| FK | Warehouse\_id | Int | Warehouse ID number, generate from the program | Not null |
|  | Stock\_level | Int |  | Not null |

# Consumption

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Name** | **Type** | **Description** | **Null?** |
| PK | Consumption\_id | Int | Consumption ID number, generate from the program | Not null |
| FK | Shop\_id | Int | Shop ID number, generate from the program | Not null |
| FK | Fruit\_id | Int | Fruit ID number, generate from the program | Not null |
|  | Quantity | Int | Consumption’s fruit quantity | Not null |
|  | season | enum | Consumption’s season (‘Spring’,’Summer’,Autumn’,’Winter’) | Not null |

# Cities

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Name** | **Type** | **Description** | **Null?** |
| PK | city\_id | Int | City ID number, generate from the program | Not null |
|  | City\_name | Varchar(50) | City’s name | Not null |
| FK | country\_id | Int | Country ID number, generate from the program | Not null |

# Countries

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Name** | **Type** | **Description** | **Null?** |
| PK | country \_id | Int | Country ID number, generate from the program | Not null |
|  | Country\_name | Varchar(50) | Country’s name | Not null |

# Relationships

1. users → shops: A user can belong to a shop.
2. users → warehouses: A user can belong to a warehouse.
3. borrowing → shops: Tracks borrow requests between shops.
4. reservations → shops and warehouses: Tracks reservations made by shops or warehouses.
5. deliveries → reservations: Tracks deliveries linked to reservations.
6. stock → shops and warehouses: Tracks stock levels in shops and warehouses.
7. consumption → shops and fruits: Tracks fruit consumption by shops.

# Brief Description

The Fruit Management System is a web-based application tailored for a bakery chain with multiple shops and warehouses. It streamlines the management of fruit stock, reservations, borrow requests, and deliveries, ensuring operational efficiency and data accuracy. The system is designed to cater to three distinct user roles: Bakery Shop Staff, Warehouse Staff, and Senior Management, each with specific functionalities and permissions.

# Major Characteristics

1. Role-Based Access Control

The system enforces strict role-based access control (RBAC) to ensure that users can only access features relevant to their roles:

Bakery Shop Staff:

* Borrow fruits from other shops.
* Reserve fruits from warehouses.
* View stock levels and consumption records.

Warehouse Staff:

* Approve or reject reservations.
* Arrange deliveries to bakery shops or other warehouses.
* Update stock levels in the warehouse.

Senior Management:

* Manage users and fruit types.
* View reserve needs and consumption records.

This approach enhances security and simplifies the user experience by hiding irrelevant features.

1. Real-Time Updates

The system dynamically updates stock levels, reservations, and borrow requests to reflect the latest changes:

* Borrow requests update the stock levels of both the borrowing and lending shops in real time.
* Delivery statuses (e.g., "In Transit", "Delivered") are updated dynamically, ensuring accurate tracking.

1. Responsive Design

The system is built using Bootstrap, ensuring a responsive and mobile-friendly interface:

* Users can access the system seamlessly on desktops, tablets, and smartphones.
* The layout adapts to different screen sizes, providing a consistent user experience across devices.

1. Data Integrity

The system enforces data integrity through the use of foreign key constraints in the database:

* Ensures that borrow requests, reservations, and deliveries are linked to valid shops, warehouses, and fruits.
* Prevents orphaned records by cascading updates and deletions where necessary.
* Validation checks in the application layer further ensure that only valid data is entered into the system.

1. Scalability

The system is designed to handle the growing needs of the bakery chain:

* Supports multiple shops and warehouses, each with its own stock levels and operations.
* The modular MVC architecture allows for easy addition of new features, such as automated notifications or integration with external systems.
* The normalized database structure ensures efficient data storage and retrieval, even as the dataset grows.

# Dsign Highlights

1. MVC Architecture

The system follows the Model-View-Controller (MVC) design pattern:

* Model: Represents the business logic and database interactions (e.g., BorrowDB, ReservationDB).
* View: JSP files render the user interface, organized by user roles (e.g., bakeryShopStaff, warehouseStaff, seniorManagement).
* Controller: Servlets handle user requests and route them to the appropriate model and view (e.g., BorrowFruitsHandler, AdminHandler).

This separation of concerns makes the system modular, maintainable, and testable.

1. Database Normalization

The database is normalized to reduce redundancy and improve performance:

* Separate tables for users, shops, warehouses, fruits, reservations, and deliveries.
* Relationships between tables are established using foreign keys, ensuring data consistency.
* Indexing is applied to frequently queried columns (e.g., user\_id, fruit\_id) to optimize performance.

1. Enhanced User Experience

The system leverages Bootstrap and jQuery to provide a clean and interactive interface:

* Bootstrap components (e.g., modals, alerts, and responsive grids) enhance usability.
* jQuery is used for dynamic interactions, such as populating modals with data and validating forms.
* Alerts provide immediate feedback to users on actions like successful reservations or errors.

1. Security Features

The system incorporates several security measures:

* Input validation prevents SQL injection and cross-site scripting (XSS) attacks.
* Session management ensures that users cannot access restricted areas without logging in.

1. Reporting and Analytics

The system provides reporting features for Senior Management:

* View consumption records to analyze trends and plan inventory.
* View reserve needs to ensure adequate stock levels in warehouses.

# Project Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Start Date | End Date | Duration |
| Requirements Gathering | 2025-04-01 | 2025-04-03 | 3 days |
| Database Design | 2025-04-04 | 2025-04-06 | 3 days |
| Backend Development (Model) | 2025-04-07 | 2025-04-12 | 6 days |
| Frontend Development (View) | 2025-04-13 | 2025-04-17 | 5 days |
| Controller Development | 2025-04-18 | 2025-04-20 | 3 days |
| Testing and Debugging | 2025-04-21 | 2025-04-23 | 3 days |
| Deployment | 2025-04-24 | 2025-04-25 | 2 days |

# Conclusions

The Fruit Management System successfully addresses the needs of the bakery chain by providing a centralized platform for managing fruits, stock levels, reservations, and deliveries. The system's modular design ensures scalability and maintainability, while its responsive interface enhances usability. Future enhancements could include automated notifications and integration with external systems for delivery tracking.

# Skill Checklist

|  |  |  |
| --- | --- | --- |
| Skills / Technology | Description | Applied |
| **JSP/Servlets for Dynamic HTML** | Used JSP and Servlets to dynamically generate HTML pages based on user roles and data. | ✔ |
| **JSP/Servlets for User Input** | Accepted user inputs from browser forms and processed them using Servlets. | ✔ |
| **JSP Action** | Utilized JSP actions like <jsp:useBean>, <jsp:setProperty>, and <jsp:getProperty>. | ✔ |
| **Custom Tag (Taglib)** | Implemented custom tags using Tag Libraries (taglib) for reusable components. | ✔ |
| **JavaBean** | Created JavaBeans for entities like User, Fruit, Reservation, and Borrow. | ✔ |
| **JDBC for Database Connection** | Used JDBC to connect to MySQL database and perform CRUD operations. | ✔ |
| **Session Checking** | Implemented session management to restrict unauthorized access to pages. | ✔ |
| **Login Control** | Developed a login system with role-based access control. | ✔ |
| **MVC Model** | Applied the MVC architecture to separate concerns (Model, View, Controller). | ✔ |
| **Bootstrap for Responsive Design** | Used Bootstrap to create a responsive and mobile-friendly user interface. | ✔ |
| **jQuery for Dynamic Interactions** | Used jQuery for client-side interactions like form validation and modal population. | ✔ |
| **Role-Based Access Control** | Restricted access to features based on user roles (e.g., Bakery Shop Staff, Warehouse Staff, Senior Management). | ✔ |
| **Database Normalization** | Designed a normalized database to reduce redundancy and improve performance. | ✔ |
| **Foreign Key Constraints** | Ensured data integrity using foreign key constraints in the database. | ✔ |
| **Session Management** | Used HTTP sessions to track logged-in users and their roles. | ✔ |
| **Password Hashing** | Secured user passwords by hashing them before storing in the database. | ✔ |
| **Testing and Debugging** | Conducted thorough testing and debugging to ensure system reliability. | ✔ |
| **Error Handling** | Implemented error handling for invalid inputs and database errors. | ✔ |
| **Reporting and Analytics** | Provided reporting features for consumption records and reserve needs. | ✔ |
| **Version Control (Git)** | Used Git for version control and collaboration. | ✔ |