**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: | Tam Tsz Kiu  230501558 |

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** |
| Tam Tsz Kiu | 100 | Vincent tam |

Content Page

[1 Assumption and The User and System Requirements 4](#_Toc196705488)

[1.1 Assumptions 4](#_Toc196705489)

[1.2 User Requirements 5](#_Toc196705490)

[1.3 System Requirements 6](#_Toc196705491)

[2 Site Map 7](#_Toc196705492)

[3 System Structure: MVC Model 8](#_Toc196705493)

[4 Database Structure 9](#_Toc196705494)

[4.1 ER-Diagram 9](#_Toc196705495)

[4.2 Data Description 10](#_Toc196705496)

[1. bakery 10](#_Toc196705497)

[2. centre\_warehouse 10](#_Toc196705498)

[3. cities 11](#_Toc196705499)

[4. fruit\_borrow\_record 11](#_Toc196705500)

[5. fruit\_reserve\_record 12](#_Toc196705501)

[6. fruit\_stock\_record 13](#_Toc196705502)

[7. fruits 13](#_Toc196705503)

[8. staff 14](#_Toc196705504)

[4.3 Relationships 14](#_Toc196705505)

[5 Brief Description 16](#_Toc196705506)

[Code Organization 16](#_Toc196705507)

[5.1 Technical Highlights in this Program and Design Highlights 16](#_Toc196705508)

[1. Frontend Architecture and Technologies 16](#_Toc196705509)

[2. Backend Architecture and Design Patterns 17](#_Toc196705510)

[3. Business Functionality Design 17](#_Toc196705511)

[4. Security and Data Validation 17](#_Toc196705512)

[5. Special Technical Highlights 18](#_Toc196705513)

[5.2 Major Characteristics 18](#_Toc196705514)

[System Architecture 18](#_Toc196705515)

[Multi-Role Access Control 18](#_Toc196705516)

[Data Management 19](#_Toc196705517)

[Business Process Automation 19](#_Toc196705518)

[Technical Implementation 19](#_Toc196705519)

[Reporting and Analysis 19](#_Toc196705520)

[6 Project Schedule 20](#_Toc196705521)

[7 Conclusions 21](#_Toc196705522)

[8 Skill Checklist 22](#_Toc196705523)

# Assumption and The User and System Requirements

# Assumptions

**1. Assumptions**

To develop the **Acer International Bakery (AIB) Enterprise System**, the following assumptions are made:

1. **User Roles & Access Control:**
   * Three distinct user roles exist: **Bakery Shop Staff, Warehouse Staff, and Senior Management**.
   * Each role has specific permissions (e.g., only Senior Management can modify fruit types).
2. **Fruit Inventory Management:**
   * Each bakery shop maintains its own stock of fruits.
   * Shops within the same city can borrow fruits from each other.
   * Reservations can be made for up to **14 days in advance**.
3. **Warehouse Operations:**
   * Central warehouses manage bulk inventory for each country.
   * Warehouse staff handle **check-in (receiving stock), checkout (shipping stock), and approval of reservations**.
4. **Delivery & Logistics:**
   * Fruits are first delivered from **source warehouses** to **central warehouses** in target countries.
   * Central warehouses then distribute fruits to local bakery shops based on reservation records.
5. **Reporting & Analytics:**
   * Senior Management can view **consumption trends, reservation summaries, and forecasts**.
   * Reports can be filtered by **shop, city, or country**.
6. **Technical Assumptions:**
   * The system is **web-based** and uses **Java EE (JSP/Servlets, JDBC, MVC)**.
   * A **relational database (MySQL/Oracle)** stores all transactional data.
   * **Session management** ensures secure login/logout functionality.

# User Requirements

Bakery Shop Staff:

|  |  |
| --- | --- |
| User Registration | Create an account with shop details. |
| Fruit Reservation | Reserve fruits from source cities for future use (up to 14 days). |
| Borrow Fruits | Request fruits from other shops in the same city |
| Check Reservations | View past and upcoming fruit reservations. |
| Update Stock | Adjust fruit stock levels after receiving deliveries or borrowing. |

Warehouse Staff:

|  |  |
| --- | --- |
| User Registration | Create an account with warehouse details. |
| Stock Management | Update fruit stock levels (check-in/check-out). |
| Reservation Approval | Approve or reject bakery shop reservation requests. |
| Delivery Coordination | Arrange shipments from source to central warehouses and then to shops.Senior |

Management:

|  |  |
| --- | --- |
| **Analytics Dashboard** | View consumption reports by shop/city/country. |
| **User Management** | Create, edit, or delete user accounts (bakery Shop/Warehouse/manager) |
| **Fruit Management** | Add, modify, or remove fruit types. |

# System Requirements

Functional Requirements :

|  |  |
| --- | --- |
| Fruit Management | CRUD operations for fruit types, stock tracking, reservations. |
| Borrowing System | Allow same-city shops to borrow fruits with approval. |
| Reservation System | Aggregate reservations to determine country-wide demand. |
| Delivery Tracking | Monitor shipments from source to central warehouse to shops. |
| Reporting | Generate consumption analytics and reservation summaries. |
| User Management | Role-based access control (RBAC) for different users. |

# Site Map

A diagram of a company

AI-generated content may be incorrect.

# System Structure: MVC Model

**4.2 MVC Component Breakdown**

**(A) Model Layer (Business Logic & Data)**

| **Component** | **Technology** | **Responsibility** |
| --- | --- | --- |
| **JavaBeans** | POJOs | Represent entities (Users, Fruits, Reservations) |
| **DAO (Data Access Objects)** | JDBC | Database operations (CRUD for Users, Fruits, etc.) |
| **Service Classes** | Java Classes | Business rules (e.g., reservation validation) |

▶ **Key Models:**

* **staff.java** (Stores user details, role)
* **Fruit.java** (Fruit type, source location)
* **FruitReserveRecord.java** (Shop requests, dates, quantities)
* **FruitStockRecord.java** (Inventory levels per location)

**(B) View Layer (Presentation)**

| **Component** | **Technology** | **Responsibility** |
| --- | --- | --- |
| **JSP Pages** | JSP + EL | Dynamic HTML rendering |
| **Custom Taglibs** | JSTL + Custom Tags | Reusable UI components |
| **Bootstrap/CSS** | Frontend | Responsive design |

# Database Structure

# ER-Diagram

A computer screen shot of a computer program

AI-generated content may be incorrect.

# Data Description

### **1. bakery**

| **Attribute Name** | **Attribute Description** |
| --- | --- |
| id | Unique identifier for each bakery |
| name | Name of the bakery |
| city\_id | Foreign key referencing the city where the bakery is located |
| location | Physical address of the bakery |
| usa\_warehouse\_distance | Distance (in km) from the New York Central Warehouse |
| japan\_warehouse\_distance | Distance (in km) from the Tokyo Main Warehouse |
| hk\_warehouse\_distance | Distance (in km) from the Hong Kong Central Depot |

### 

### **2. centre\_warehouse**

| **Attribute Name** | **Attribute Description** |
| --- | --- |
| id | Unique identifier for each warehouse |
| name | Name of the warehouse |
| location | Physical address of the warehouse |

### 

### **3. cities**

| **Attribute Name** | **Attribute Description** |
| --- | --- |
| id | Unique identifier for each city |
| city | Name of the city |
| country | Country where the city is located (USA, Japan, Hong Kong) |

### 

### **4. fruit\_borrow\_record**

| **Attribute Name** | **Attribute Description** |
| --- | --- |
| id | Unique identifier for each borrow record |
| stock\_id | Foreign key referencing the fruit stock being borrowed |
| bakery\_id | Foreign key referencing the bakery requesting the borrow |
| borrow\_bakery\_id | Foreign key referencing the bakery lending the stock |
| state | state 0: Pending - Initial state when a reservation is created but not yet approved  state 1: Approved - Reservation has been approved by warehouse management  state 2: Sending to centre warehouse - Fruit is in transit from origin to the central warehouse  state 3: Arrived at centre warehouse - Fruit has arrived at the central warehouse facility  state 4: Sending to bakery store - Fruit is in transit from central warehouse to destination bakery  state 5: Arrived at bakery store - Fruit has arrived at the destination bakery location  state 6: Completed - Reservation process has been successfully completed |

### 

### **5. fruit\_reserve\_record**

| **Attribute Name** | **Attribute Description** |
| --- | --- |
| id | Unique identifier for each reservation record |
| fruit\_id | Foreign key referencing the fruit being reserved |
| bakery\_id | Foreign key referencing the bakery making the reservation |
| state | Status of the reservation (0=Pending, 1=Approved, 2=Rejected) |
| quantity | Quantity of fruit reserved (in kg) |
| create\_date | Timestamp when the reservation was created |
| arrival\_date | Expected arrival date of the reserved fruit |
| origin\_to\_warehouse | Distance (in km) from the fruit origin to the warehouse |
| warehouse\_to\_store | Distance (in km) from the warehouse to the bakery |

### 

### **6. fruit\_stock\_record**

| **Attribute Name** | **Attribute Description** |
| --- | --- |
| id | Unique identifier for each stock record |
| fruit\_id | Foreign key referencing the fruit in stock |
| bakery\_id | Foreign key referencing the bakery where the stock is stored |
| quantity\_kg | Quantity of fruit in stock (in kg) |
| expired\_date | Expiration date of the stock |
| borrow\_record | Foreign key referencing the bakery that borrowed the stock (if applicable) |
| created\_at | Timestamp when the stock record was created |
| updated\_at | Timestamp when the stock record was last updated |

### 

### **7. fruits**

| **Attribute Name** | **Attribute Description** |
| --- | --- |
| id | Unique identifier for each fruit |
| fruit\_name | Name of the fruit |
| shelf\_life | Shelf life of the fruit (in days) |
| city\_name | Origin city of the fruit |
| usa\_warehouse\_distance | Distance (in km) from the New York Central Warehouse |
| japan\_warehouse\_distance | Distance (in km) from the Tokyo Main Warehouse |
| hk\_warehouse\_distance | Distance (in km) from the Hong Kong Central Depot |

### 

### **8. staff**

| **Attribute Name** | **Attribute Description** |
| --- | --- |
| id | Unique identifier for each staff member |
| user\_id | Unique username for login |
| name | Full name of the staff member |
| password | SHA-256 hashed password |
| role | Role of the staff (management, warehouse, bakery, none( no active)) |
| store\_id | Foreign key referencing the bakery where the staff works (only for bakery staff) |

# Relationships

**Key Relationship :**

1. **Borrowing System**:
   * A bakery can request fruit from another bakery through **fruit\_borrow\_record**
   * The request must reference both bakeries and the specific stock item
2. **Inventory Flow**:
   * Fruits are reserved (**fruit\_reserve\_record**) → stocked (**fruit\_stock\_record**) → potentially borrowed (**fruit\_borrow\_record**)
3. **Geographic Hierarchy**:
   * Cities contain bakeries
   * Bakeries have staff
   * Both bakeries and fruits have warehouse distance metrics
4. **Staff Access Control**:
   * Bakery staff are tied to specific locations via **store\_id**
   * Warehouse and management staff aren't location-specific

# Brief Description

**6.1 System Architecture Overview**

The AIB Enterprise System is implemented using a **three-tier Java EE architecture** following MVC pattern, with clear separation between:

* **Presentation Layer (View):** JSP pages with Bootstrap for responsive UI
* **Business Logic Layer (Controller/Model):** Servlets + Service classes
* **Data Access Layer (Model):** DAO pattern with JDBC

**6.2 Key Implementation Features**

**Core Functional Modules**

1. **User Management System**
   * Role-based access control (3 user types)
   * Secure authentication with session management
   * Servlet filters for authorization
2. **Fruit Inventory System**
   * Real-time stock tracking across shops/warehouses
   * Reservation system with 14-day advance booking
   * Same-city fruit borrowing with approval workflow
3. **Logistics Management**
   * Multi-level delivery tracking (Source→Central→Local)
   * Check-in/check-out processes for warehouses
4. **Reporting & Analytics**
   * Aggregate reservation data visualization
   * Seasonal consumption trend analysis

# Code Organization

The project follows strict package organization:

src/

├── controller/ # All servlets

├── entity/ # JavaBean classes

├── service/ # Business logic

├── db/ # DAO implementations

webapp/

├── WEB-INF/

├── bakery/ # Shop staff views

├── warehouse/ # Warehouse staff views

├── management/ # Admin views

# Technical Highlights in this Program and Design Highlights

## 1. Frontend Architecture and Technologies

* **Single Page Application Style**: Using jQuery's dynamic content loading ($("#content").load(url)) to update content without reloading the entire page
* **Parameter-driven Component Loading**: Using URL parameters (?loadComponent=xyz) to control which component to load, implementing a simple frontend routing
* **Responsive Layout**: Using Bootstrap framework for responsive design, adapting to different device sizes
* **DataTables Integration**: Integration of DataTables plugin in fruitManagement.jsp, providing powerful data table functionality including sorting, filtering, and exporting

## 2. Backend Architecture and Design Patterns

* **Standard MVC Architecture**: Clear separation of Model-View-Controller
* **Multi-tier Architecture**:
* Controller layer handles request routing
* Service layer encapsulates business logic
* DAO/DB layer handles data access
* Entity layer defines data models
* **Infrastructure Abstraction**: Using BaseDB as a base class for data access, unifying connection management and basic operations, optimum the school workshop to make a baseDB for other Date access easy to build up.

## 3. Business Functionality Design

* **Multi-role System**: Distinguishing functionality for different roles - Bakery, Warehouse, and Management
* **Logistics Management**: Implementation of complex fruit inventory and transport logistics management
* **Reservation and Borrowing System**: Implementation of fruit reservation (Reserve) and borrowing (Borrow) functions
* **Geographic Distance Calculation**: Application of geographic distance model to calculate distances between different warehouses

## 4. Security and Data Validation

* **Client-side Validation**: Frontend form validation using Bootstrap's validation features
* **Server-side Validation**: Backend input checking and exception handling
* **Session Management**: Implementation of login/logout functionality and user session control

## 5. Special Technical Highlights

* **Hybrid Dynamic Content Loading**: Combining server-side forwarding and client-side dynamic loading, maintaining clear URL structure while providing smooth user experience
* **Modular Design**: Building interfaces in a component-based way, with each functional block as an independent JSP component
* **Data Analysis Capabilities**: FruitAnalysisService and FruitAnalysisDB provide data analysis capabilities
* **Exception Handling Mechanism**: Good exception catching and user feedback mechanisms

# Major Characteristics

## System Architecture

* **Three-Tier MVC Architecture**: Clear separation of Model (data and business logic), View (UI components), and Controller (request handling)
* **Modular Component Design**: Functionality organized into independent, reusable components
* **Domain-Driven Design**: System structure reflects the business domain (fruits, bakeries, warehouses, logistics)

## Multi-Role Access Control

* **Role-Based Interface Segregation**: Separate portals for Bakery Staff, Warehouse Personnel, and Management
* **Specialized Workflows**: Each role has access to specific functions relevant to their responsibilities
* **Centralized Authentication**: Unified login system with role-specific redirections

## Data Management

* **Comprehensive Fruit Lifecycle Tracking**: From sourcing to distribution
* **Real-Time Inventory Visibility**: Current stock levels across all locations
* **Geographic Distance Optimization**: Warehouse selection based on distance calculations
* **Shelf-Life Management**: Tracking and managing perishable items based on shelf-life parameters

## Business Process Automation

* **Reservation System**: Automated fruit reservation process for bakeries
* **Borrowing Mechanism**: Streamlined process for bakeries to borrow fruits from other locations
* **Approval Workflows**: Digital approval process for requests between business units
* **Supply Chain Visibility**: End-to-end tracking of fruit movement across the supply network

## Technical Implementation

* **AJAX-Based Dynamic Content Loading**: Enhanced user experience without full page reloads
* **Responsive Design**: Bootstrap-based interface adapting to different device sizes
* **Advanced Data Presentation**: Sortable, filterable data tables with export capabilities
* **State Management**: Effective handling of application state during user interactions

## Reporting and Analysis

* **Real-Time Dashboards**: Visual representation of system status
* **Analytical Tools**: Built-in capabilities for data analysis and decision support
* **Export Functionality**: Data extraction in multiple formats (CSV, Excel, PDF)
* **Audit Trails**: Tracking of system activities for compliance and troubleshooting

# Project Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Start Date | End Date | Duration |
| Read Requirement | 2025-04-24 | 2025-04-24 | 1days |
| Database Design | 2025-04-25 | 2025-04-25 | 0.5 days |
| Service Development (Model & Controller) | 2025-04-25 | 2025-04-28 | 3 days |
| Frontend Development (View) | 2025-04-25 | 2025-04-28 | 3 days |
| Testing and Debugging | 2025-04-28 | 2025-04-28 | 1 days |
| Deployment | 2025-04-28 | 2025-04-28 | 1 days |

# Conclusions

The AIBFMS represents a significant advancement in the management of bakery fruit inventory, successfully addressing the business challenges while establishing a technical foundation capable of evolving with future business needs. The modular architecture ensures that the system can adapt to changing requirements without requiring a complete redesign.

Through this implementation, the organization has not only gained an efficient tool for daily operations but has also established a digital platform that can drive strategic improvements in its supply chain management processes.

# Skill Checklist

|  |  |  |
| --- | --- | --- |
| Skills / Technology | Description | Applied |
| **JSP/Servlets for Dynamic HTML** | Multiple JSP pages across bakery, warehouse, and management modules dynamically generate HTML content  Examples: fruitManagement.jsp (generates dynamic tables), reserve.jsp (shows customized reservation interfaces)  Evidence: Use of scriptlet tags <% %> for dynamic content generation and conditional rendering | ✔ |
| **JSP/Servlets for User Input** | Form submissions handled by various servlets  Examples: FruitManagementServlet.java processes form submissions for adding, updating, and deleting fruits  Evidence: Parameter extraction like request.getParameter("fruitName"), form processing in doPost methods | ✔ |
| **JSP Action** | Implementation: Standard JSP actions used for component inclusion and bean manipulation  Examples: <%@ include file="./component/header.jsp" %> in bakeryHome.jsp  Evidence: Page compositions using include actions | ✔ |
| **Custom Tag (Taglib)** | Custom tags implemented in the tags directory  Examples: Custom tags in the src/java/tags/ directory  Evidence: Tag usage in various JSP pages for specialized functionality | ✔ |
| **JavaBean** | Extensive use of JavaBeans for data encapsulation  Examples: Fruit.java, Staff.java, FruitReserveRecord.java and other entity classes  Evidence: Proper encapsulation with private variables and public getter/setter methods | ✔ |
| **JDBC for Database Connection** | Database operations using JDBC throughout the application  BaseDB.java providing database connection functionality  Evidence: Implementation in various DB classes like FruitDB.java, StaffDB.java | ✔ |
| **Session Checking** | Implementation: Session management for user authentication and state maintenance  Examples: Session validation in controller servlets and JSP pages  Evidence: Session checks and user role verification | ✔ |
| **Login Control** | Implementation: Comprehensive login system with role-based access control  Examples: loginServlet.java handling authentication, logoutServlet.java for session termination  Evidence: Role-specific redirections and protected resources | ✔ |
| **MVC Model** | Implementation: Clear separation of Model, View, and Controller components  Examples: Entity classes and DB classes (Model), JSP pages (View), and Servlets (Controller)  Evidence: Organized directory structure separating concerns (entity, service, db, controller packages) | ✔ |
| **AJAX-based** | AJAX-based dynamic content loading: Using jQuery to load content without full page reloads | ✔ |
| **Responsive design:** | Responsive design: Bootstrap framework implementation for cross-device compatibility | ✔ |
| **DataTables integration** | : Advanced data visualization and manipulation  Multi-tier architecture: Separation of data access, business logic, and presentation layers | ✔ |
| **Parameter-based component routing:** | Simple but effective front-end routing mechanism | ✔ |
| **Exception handling:** | Comprehensive error handling and user feedback | ✔ |
| **Export functionality:** | Data exports in multiple formats (CSV, Excel, PDF)  State tracking system: Sophisticated state management for business processes | ✔ |
| **Reporting and Analytics** | Provided reporting features for consumption records and reserve needs. | ✔ |
| **Version Control (Git)** | Used Git for version control and collaboration. | ✔ |