Final Project

Food Waste Reduction Platform

Xihai Ren
Alves Martins Angela
Uyen Minh Trinh
Sam Doiron

Teamwork

- Xihai Ren: set up the project(environment, database), front end, user authentication
- Angela: built the Dao Pattern for food and store, built
- Uyen Minh:
- Sam:

Solution

- **Overview**: Our Food Waste Reduction Platform integrates multiple components seamlessly to achieve our goal of connecting retailers, consumers, and charitable organizations to minimize food waste.
- **Backend**: Utilized Java EE for developing Servlet web services that interact with the database using Data Access Objects (DAOs). Implemented business logic through service layers.
- **Frontend**: Developed responsive user interfaces using JSP, HTML, CSS (Bootstrap), and JavaScript (jQuery) to provide a smooth user experience.
- **Database**: Managed data persistence using a MySQL database. Used JDBC for database connectivity and operations.

External Libraries

- **Bootstrap**: Used for responsive and consistent styling across all web pages.
- **jQuery**: Facilitated DOM manipulation and AJAX calls for asynchronous data fetching.
- **Chart.js**: Employed for creating dynamic charts to display statistics on the dashboard.
- **JUnit**: Integrated for unit testing the backend services and DAOs to ensure code reliability and correctness.

Design Patterns

- **MVC (Model-View-Controller)**: Implemented to separate concerns and enhance code maintainability. The model handles data, the view manages the UI, and the controller processes user input and interactions.
- **Singleton Pattern**: Applied in the `JDBCClient` class to ensure a single instance of the database connection throughout the application, optimizing resource usage.
- **Observer Pattern:** Observer is a publish/subscribe pattern, which allows a number of observer objects to see an event. Applied in the SurplusFoodServlet for notify subscribe that new surplus food is available.
- **DAO Pattern** : For instance, the FoodDao, ClaimDao, and EmailNotificationDao classes handle database operations, isolating these concerns from the rest of the application.
- **Template Method Pattern** : deferring some steps to subclasses. Seen in the HttpServlet subclasses like FeedbackServlet, where doGet and doPost methods provide a template for request processing.
- **Front Controller Pattern** : The processRequest method in servlets like FeedbackServlet acts as a front controller, directing requests to appropriate handlers based on parameters.

JUnit and Code Coverage

- **Unit Testing**: Extensively tested DAOs and service layers using JUnit to validate the functionality of each component.
- **Code Coverage**: Achieved an overall code coverage of approximately 80%, ensuring that most of the codebase is tested and potential bugs are minimized.

Additional Information

- **Security Measures**: Implemented user authentication using the `AuthService` and `AuthServiceImpl` classes to manage login and registration securely.
- **User Roles**: Defined different user roles (retailer, consumer, organization) with specific functionalities tailored to each role.

Version Control System

GitHub Utilization:

- **Central Repository**: We used GitHub as the central repository for our project's source code. It facilitated collaboration and ensured all team members had access to the latest version of the code.
- **Version Control**: Enabled us to track changes, manage versions, and maintain a history of the project's development.
- **Issue Tracking**: Utilized GitHub Issues to track bugs, feature requests, and tasks. Each issue was assigned to the relevant team member, ensuring accountability and clear tracking of progress.

GitHub Repo Link

- **Repository URL**: https://github.com/RyanRen2023/fwrp.git
    - This link provides access to the repository where all project files, commit history, and issue tracking can be viewed.

Functionalities Used

Branching Strategy:

- **Main Branch**: Maintained the stable version of the code, reflecting the latest production-ready state.
- **Dev Branch**: Used for integrating new features and testing before merging into the main branch.
- **Dev Tomee Branch**: Specialized branch for development and testing with the Tomee server.
- **Dev Tomc9 Branch**: Primary branch for most of the development work.

Additional Information

Collaboration and Coordination:

- GitHub facilitated collaboration by providing a platform where all team members could contribute, review, and discuss code. Regular updates and clear commit messages ensured everyone stayed informed about the project's progress.

Documentation:

- Maintained comprehensive documentation within the repository, including README files, setup guides, and API documentation. This ensured that all team members and future developers had access to essential information about the project.

Challenges

Coding Challenges:

- **Asynchronous Data Handling**:
  - **Issue**: Managing asynchronous data loading and ensuring smooth interactions between the frontend and backend.
  - **Solution**: Implemented AJAX calls using jQuery to handle asynchronous requests, ensuring data consistency and user experience.
- **Database Optimization**:
  - **Issue**: Optimizing database queries to handle large datasets efficiently.
  - **Solution**: Indexed key columns and optimized SQL queries to improve performance.

Challenges

Deployment Challenges:

- **Server Configuration**:
  - **Issue**: Configuring the server environment for deployment.
  - **Solution**: Utilized containerization tools like Docker to standardize the deployment environment, ensuring consistency across different stages.
- **Environment Variables Management**:
  - **Issue**: Managing environment-specific configurations securely.
  - **Solution**: Used configuration files and environment variables to separate configuration from code, ensuring security and flexibility.

Challenges

Testing Challenges:

- **Achieving High Code Coverage**:
    - **Issue**: Ensuring extensive test coverage for the codebase.
    - **Solution**: Implemented unit tests using JUnit for DAOs and services, aiming for at least 80% code coverage.

Challenges

Overcoming Challenges:

- **Research and Learning**:
  - **Approach**: Spent time researching solutions and learning new technologies to address complex issues.
  - **Example**: Consulted documentation, online forums, and tutorials to overcome technical obstacles.
- **Team Collaboration**:
  - **Approach**: Leveraged team strengths and collaborative problem-solving to tackle challenges.
  - **Example**: Conducted regular team meetings to discuss issues and brainstorm solutions, leading to more effective problem resolution.

Demo

Required Functionalities:

- **User Authentication**:
  - Demonstrate the login and registration process, showing how users can create an account and log in to the platform.
  - Highlight different user roles (retailer, consumer, charitable organization) and their respective access.
- **Inventory Management**:
  - Show how retailers can add, edit, and delete food items in the inventory.
  - Display the inventory list and demonstrate the modal dialogs for adding and editing items.
- **Surplus Food Management**:
  - Illustrate the surplus food section where users can view and claim surplus food items.
  - Show the claim process for charitable organizations and consumers.
- **Notifications**:
  - Demonstrate the notifications system where users receive alerts for surplus food and discounts.
  - Show how users can view and manage their notifications.
- **Feedback System**:
  - Show how users can submit feedback and view submitted feedback.
  - Display the feedback form and the list of feedback entries.

Demo

Bonus Functionalities:

- **Advanced Search Filters**:
  - Demonstrate the advanced search functionality with filters for food type, price range, expiration date, and location.
  - Show how users can refine their search to find specific food items.
- **Subscription to Alerts**:
  - Illustrate the subscription system where users can subscribe to different types of alerts (surplus, discount, expiring soon).
  - Show the subscription form and the list of subscribed alerts.
- **Dashboard Statistics**:
  - Show the dashboard with dynamic charts displaying statistics based on user roles (retailer, consumer, charity).
  - Demonstrate the different statistics available for each role, such as inventory levels, purchase trends, and claim trends.

Demo

Future Enhancement

What is your plan for the future enhancement of the project