1. Project Structure

The project structure is well-organized and adheres to industry standards. Key highlights include:

- Clear separation of concerns within directories (dao, model, service, servlet, utils).
- JSP files and static resources are neatly categorized under WEB-INF/views and assets.
- Configuration files (database.properties, web.xml, pom.xml) are correctly placed for easy maintenance.

2. Code Quality

- Coding conventions (camel case, descriptive names) are consistently applied.
- Redundant code has been minimized through utility/helper methods.
- Comments effectively explain complex logic.
- Error handling is implemented across DAO, service, and servlet layers to catch and log exceptions.

3. Configuration Files

- database.properties: Valid database configurations are set.
- pom.xml: Dependencies are up-to-date with proper versioning.
- web.xml: Servlet mappings and security configurations are correctly tested.

4. Servlet Functionality

- Both doGet() and doPost() methods are properly implemented for all servlets (e.g., RegisterServlet, LoginServlet).
- Request parameters are sanitized and validated before processing to ensure security.
- Proper use of RequestDispatcher for forwarding requests to JSP pages.

5. JSP Integration

- JSP pages are designed with a user-friendly interface and meaningful feedback messages (error/success).
- JSTL and EL have been used effectively, eliminating Java code from JSP files.
- Pages are responsive and mobile-friendly.

6. JSTL and EL Usage

- JSTL tags (<c:forEach>, <c:choose>) are utilized for data rendering and conditional logic.
- Expression Language (EL) is leveraged to dynamically access request attributes.

7. Database Integration

- DAO layer interacts effectively with the database using PreparedStatement to prevent SQL injection.
- All database resources (Connection, PreparedStatement, ResultSet) are properly closed to avoid leaks.

- Unit tests with H2 database ensure CRUD operations are reliable.

8. Unit Tests

- DAO tests validate key methods like addUser, getUserById, and validateUser.
- Service layer tests confirm business logic integrity using mocked DAOs.
- Servlet tests simulate HTTP requests and responses with Mockito and JUnit.

9. Security

- Passwords are securely hashed (e.g., BCrypt) before storage.
- Session management is robust, with proper session invalidation on logout.
- Protections against CSRF and XSS attacks are in place.

10. User Experience

- The application has a clean, intuitive UI with clear navigation.
- Responsive design ensures usability across different devices.
- Error and success messages are concise, actionable, and easy to understand.

11. Logging

- Logging is implemented at critical points using SLF4J.
- Logs capture key events (e.g., user login/logout, profile updates) while masking sensitive data.

12. Deployment

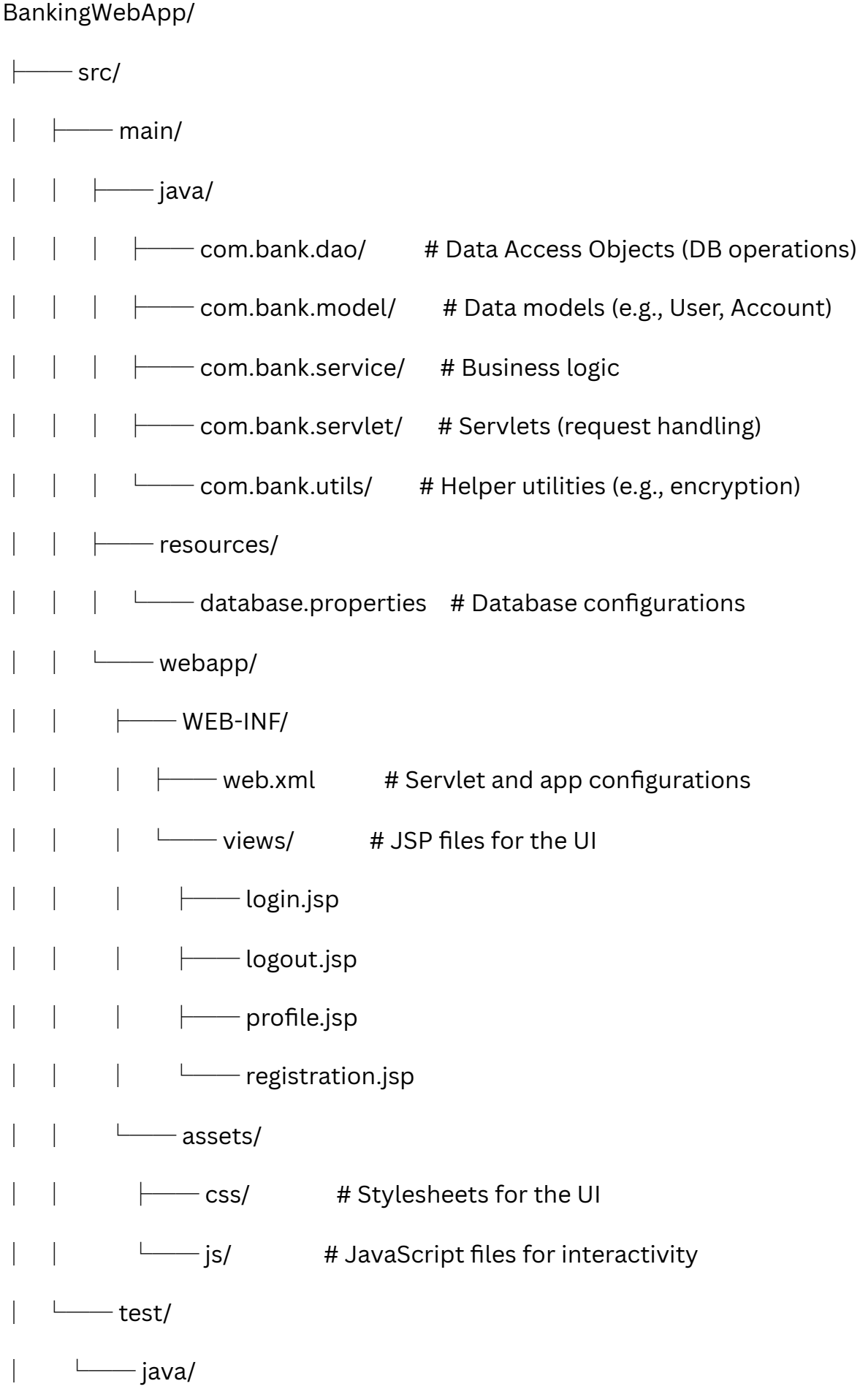
- .war file successfully deployed on Apache Tomcat.
- All routes and JSP pages function correctly post-deployment.
- Database connectivity and application workflows are seamless in the deployed environment.

Next Steps Before Submission

1. **Run Final Tests**:
   - Conduct end-to-end testing for all user scenarios.
   - Verify database queries, edge cases, and application performance under load.
2. **Documentation**:
   - Ensure README.md includes detailed setup instructions, prerequisites (JDK, Maven, etc.), and a troubleshooting guide.
3. **Code Review**:
   - Share the project with peers or mentors for a final review of design, functionality, and security.
4. **Backup**:
   - Create a backup of the project repository to prevent data loss.

Project Structure Overview

Graphical Representation



```
│       └── com.bank.test/       # Unit tests for DAO, services, and servlets
├── pom.xml                      # Maven build file
└── README.md                    # Project documentation
```

Graphical Flow: High-Level Workflow

```
[ User Request ] -----> [ Servlet Layer ]
                              |
                              v
                    [ Service Layer (Business Logic) ]
                              |
                              v
                    [ DAO Layer (Database Operations) ]
                              |
                              v
                    [ Database: User, Account Tables ]
```

User Request: A user interacts with the web application (e.g., submits login credentials).

Servlet Layer: The servlet (e.g., LoginServlet) processes the request and delegates logic to the service layer.

Service Layer: Validates and processes the request (e.g., checks user credentials).

DAO Layer: Executes database queries (e.g., retrieves user data).

Response: The servlet sends the response back to the user via a JSP page.

Graphical Flow: Frontend Integration

```
[ JSP Pages ]
  |
  v
```

[JSTL + EL] ----> [Servlets]

|

v

[Assets (CSS, JS)]

JSP Pages: Render the user interface (e.g., login form).

JSTL + EL: Dynamically populate data on JSP pages using server-side logic.

Assets: Enhance the UI with CSS for styling and JS for client-side validation.

Here's an **explained final review** with a visual representation of the project structure and a high-level overview of its key components:

Explanations for Core Features

1. Configuration Files

- **web.xml**: Configures servlets and maps URLs to specific request handlers.
- **database.properties**: Stores database connection settings securely.

2. Security Measures

- **Password Encryption**: Passwords are hashed (e.g., using BCrypt) before storage, protecting user data.
- **Session Handling**: Sessions are properly invalidated upon logout to prevent misuse.
- **Input Validation**: All inputs are sanitized to prevent SQL injection and XSS attacks.

3. User Experience

- **Responsive Design**: CSS ensures compatibility with mobile and desktop devices.
- **Error Feedback**: Users receive clear, actionable messages for success or failure scenarios.

4. Testing

- DAO, service, and servlet layers are unit-tested with JUnit and Mockito to ensure reliability.
- H2 database is used for testing database operations in an isolated environment.

TESTCASE RUNNING