Phase-End Project

Railway Crossing Status

- **Algorithm**

1. Set up the "railway" database with the following tables: "user," "admin," "adminhome," and "favorites."
2. AdminLogin Functionality:
   - Retrieve the username and password from the login form.
   - Connect to the "railway" database.
   - Execute a SELECT query to check if the provided credentials match an admin record in the "admin" table.
   - If a matching record is found, redirect to the admin home page (adminhome.jsp).
   - If not found, display an error message and provide an option to try again.
3. AddToFavorite Functionality:
   - Retrieve the ID of the crossing to be added to favorites.
   - Connect to the "railway" database.
   - Execute an INSERT INTO statement to add the selected crossing to the "favorites" table.
   - Close the database connection.
   - Redirect the user back to the user home page.
4. DeleteRecord Functionality:
   - Retrieve the ID of the crossing to be deleted.
   - Check if the ID is provided.
   - Connect to the "railway" database.
   - Execute a DELETE FROM statement to remove the crossing from the "adminhome" table.
   - Close the database connection.
   - Redirect back to the admin home page.
5. RemoveFromFavorite Functionality:
   - Retrieve the ID of the crossing to be removed from favorites.
   - Connect to the "railway" database.
   - Execute a DELETE FROM statement to remove the crossing from the "favorites" table.
   - Close the database connection.
   - Redirect the user back to the favorites page (favorite.jsp).
6. UserLogin Functionality:
   - Retrieve the email and password from the login form.
   - Connect to the "railway" database.
   - Execute a SELECT query to check if the provided credentials match a user record in the "user" table.
   - If a matching record is found, redirect to the user home page.
   - If not found, display an error message and provide an option to try again.
7. Registration Functionality:
   - Retrieve the username, email, and password from the registration form.
   - Create a Member object with the provided details.
   - Connect to the "railway" database.

    - Execute an INSERT INTO statement to add the new user to the "user" table.
    - Close the database connection.
    - Display a success message or an error message if the registration fails.
8. Create the necessary JSP files:
    - addrail.jsp: Contains a form for adding a new railway crossing.
    - adminhome.jsp: The admin home page that displays a table of railway crossings.
    - adminlogin.jsp: Provides a form for admin login.
    - delete.jsp: Used to delete a crossing record.
    - favorite.jsp: Displays a list of favorite railway crossings.
    - index.jsp: The main page of the application with links for user and admin login.
    - login.jsp: Provides a form for user login.
9. Divide the implementation into sprints:
    - Sprint 1: Set up the database and implement the AdminLogin functionality.
    - Sprint 2: Implement the AddToFavorite and DeleteRecord functionality.
    - Sprint 3: Implement the RemoveFromFavorite and UserLogin functionality.
    - Sprint 4: Implement the Registration functionality and perform testing for all features.
10. Handle edge cases and perform thorough testing for all implemented features to ensure the application functions correctly and securely.

- **Sprint Planning**

Sprint 1:

Set up the database with the required tables.

Implement the AdminLogin functionality.

Sprint 2:

Implement the AddToFavorite functionality.

Implement the DeleteRecord functionality.

Sprint 3:

Implement the RemoveFromFavorite functionality.

Implement the UserLogin functionality.

Sprint 4:

Implement the Registration functionality.

Handle edge cases and perform testing for all implemented features.