RAILWAY CROSSING STATUS

ALOGARITHM :

1. **index.jsp**: This is the home page of the application. It provides links for user login and admin login.

2. **login.jsp**: This JSP file contains the login form for users. It collects the username and password from the user.

3. **Login.java** (servlet): This servlet is responsible for handling the user login process. It retrieves the username and password from the request parameters, performs a database query to validate the credentials, and redirects the user to the "userhome.jsp" page if the login is successful. Otherwise, it displays an error message.

4. **Registration.jsp**: This JSP file contains the registration form for users. It collects the username, password, and email from the user.

5. **Register.java** (servlet): This servlet handles the user registration process. It retrieves the username, password, and email from the request parameters, creates a **Member** object, and uses a **RegisterDao** to insert the user into the database.

6. **RegisterDao.java**: This class is responsible for database operations related to user registration. It has a method **insert** that takes a **Member** object as input and inserts the user into the database.

7. **Member.java**: This class represents a user and contains their username, password, and email.

8. **Userhome.jsp**: This file represents the user's home page. It displays a list of railway crossing information retrieved from a MySQL database. It also provides links to navigate to other pages like search and favorite crossings.

9. **Search.jsp**: This file displays a simple form where users can enter a keyword to search for railway crossings. It submits the form data to the **searchResult.jsp** page.

10. **SearchResult.jsp**: This file processes the search query entered by the user and performs a database query to retrieve matching railway crossings. It displays the search results in a table format.

11. **Favorite.jsp**: This file is likely intended to display the user's favorite railway crossings. However, the code provided is similar to the **SearchResult.jsp** file, so it appears to be a duplication or a mistake.

12. AddToFavorite.java (Servlet): This servlet is responsible for adding an item to the user's favorites. It establishes a database connection, prepares an SQL statement to insert the favorite item, executes the statement, and then redirects the user back to the user home page.

13. RemoveFromFavorite.java (Servlet): It seems that there was an error in copying the code, as this file is identical to AddToFavorite.java. However, based on the name, it is likely intended to remove an item from the user's favorites. You would need to provide the correct code for this functionality.

14. Adminlogin.jsp: This JSP (JavaServer Pages) file is a view that displays search results from the adminhome table. It sets up a data source and performs a query to search for records that match a given keyword in the Name or Address columns. The search results are then displayed in an HTML table.
15. AdminLogin.java (Servlet): This servlet handles the login functionality for the admin user. It receives the username and password parameters, establishes a database connection, and executes a query to check if the provided credentials are valid. If the login is successful, it forwards the request to the adminhome.jsp page. Otherwise, it displays an error message indicating a failed login attempt.

16. **adminhome.jsp**: This JSP file displays search results for railway crossings. It uses SQL queries to retrieve data from a MySQL database and displays it in a tabular format.

17. **Addrail.jsp**: This JSP file contains a form for adding a new railway crossing. It includes input fields for the name, address, landmark, train schedule, person in charge, and status of the crossing.

18. **RailCross.java**: This Java class defines a **RailCross** object that represents a railway crossing. It has private fields for the crossing's attributes and corresponding getters and setters.

19. **RailCrossing.java**: This servlet receives form data from **Addrail.jsp** and inserts a new railway crossing into the MySQL database using the **RailCross** object.

20. **Update.jsp**: This JSP file displays a form for updating an existing railway crossing. It is similar to **Addrail.jsp** but pre-fills the form with the existing crossing's data.

21. **UpdateProcess.jsp**: This JSP file handles the update process by updating the corresponding record in the MySQL database based on the form data received from **Update.jsp**.

22. **Delete.jsp**: This JSP file handles the deletion of a railway crossing record from the MySQL database.