

RAILWAY CROSSING STATUS

ALGORITHM

Step1. Create a database named "railway" and switch to that database.

Step 2. Create a table named "user" with columns "uname", "email", and "password".

Step 3. Create a table named "admin" with columns "uname" and "password".

Step 4. Insert a record into the "admin" table with the values 'Admin' and '12345'.

Step 5. Create a table named "adminhome" with columns "id", "Name", "Address", "Landmark", "Trainschedule", "pname", and "status".

- "id" column is defined as an auto-increment integer with a unique index.

Step 6. Create a table named "favorites" with columns "Address", "id", "Name", "Landmark", "Trainschedule", "pname", and "status".

- "id" column is defined as the primary key.

Step7. Create an index.jsp file containing HTML code for the index page.

Step 8. Create a login.jsp file containing HTML code for the user login page.

Step Create a Registration.jsp file containing HTML code for the user registration page.

Step 10. Create an adminlogin.jsp file containing HTML code for the admin login page.

Step11. Create an adminhome.jsp file containing HTML code for the admin home page.

- This page also includes a database query to fetch data from the "adminhome" table and display it in a table format.

Step 12. Create an addrail.jsp file containing HTML code for adding a railway crossing.

Step13 Create an update.jsp file containing HTML code for updating a railway crossing.

- This page also includes a database query to fetch data of a specific crossing based on the provided ID and pre-fill the form fields with the fetched data.

Step 14. Create an updateProcess.jsp file containing HTML code for updating the database record based on the provided form data.

Step 15. Create a delete.jsp file containing HTML code for deleting a railway crossing.

- This page also includes a database query to fetch data of a specific crossing based on the provided ID and display the confirmation message for deletion.

Step16. Create a deleteProcess.jsp file containing HTML code for deleting the database record based on the provided ID.

- This page executes the database query to delete the record and displays a success message.

RAILWAY CROSSING STATUS

ALGORITHM

Step 17. Create a package named `adminlogin` and create a servlet named `AdminLogin`.

Step18 Inside the `doPost` method of the `AdminLogin` servlet:

- Get the `uname` and `password` parameters from the request.
- Establish a database connection using JDBC.
- Prepare a SQL query to select the `uname` from the `admin` table where `uname` and `password` match the provided values.
- Set the values of the parameters in the prepared statement.
- Execute the query and store the result in a `ResultSet`.
- If the result set has a next record:
 - Forward the request and response to the `adminhome.jsp` page.
- Else:
 - Print an error message indicating login failure.

Step19 Create a package named `favorite` and create a servlet named `AddToFavorite`.

Step20 Inside the `doPost` method of the `AddToFavorite` servlet:

- Get the `itemId` parameter from the request.
- Establish a database connection using JDBC.
- Prepare a SQL statement to insert the selected favorite crossing from the `adminhome` table into the `favorites` table based on the provided `itemId`.
- Set the value of the `itemId` parameter in the prepared statement.
- Execute the SQL statement and get the number of affected rows.
- Close the statement and the database connection.
- Redirect the user back to the `userhome.jsp` page.

Step 21. Create a package named `login` and create a servlet named `Login`.

step22. Inside the `doPost` method of the `Login` servlet:

- Get the `uname` and `password` parameters from the request.
- Establish a database connection using JDBC.
- Prepare a SQL query to select the `uname` from the `user` table where `uname` and `password` match the provided values.
- Set the values of the parameters in the prepared statement.

RAILWAY CROSSING STATUS

ALGORITHM

- Execute the query and store the result in a `ResultSet`.
- If the result set has a next record:
 - Forward the request and response to the `userhome.jsp` page.
- Else:
 - Print an error message indicating login failure.

step 23. Create a class named `RailCross` inside the `rail` package.

step 24. Add private instance variables for `name`, `address`, `landmark`, `trainSchedule`, `personInCharge`, and `status` to the `RailCross` class.

Step25 . Generate getters and setters for the instance variables.

step 26. Create a servlet named `RailCrossing` inside the `rail` package.

step 27. Inside the `doPost` method of the `RailCrossing` servlet:

- Get the parameters for `name`, `address`, `landmark`, `trainSchedule`, `personInCharge`, and `status` from the request.
- Create a new `RailCross` object with the provided values.
- Establish a database connection using JDBC.
- Prepare a SQL statement to insert the `RailCross` object into the `adminhome` table.
- Set the values of the parameters in the prepared statement.
- Execute the SQL statement and get the number of affected rows.
- Close the statement and the database connection.
- Return a success or failure message.

step 27. Create a class named `Member` inside the `register` package.

step 28. Add private instance variables for `uname`, `password`, and `email` to the `Member` class.

step 29. Generate getters and setters for the instance variables.

step 30. Create a servlet named `Register` inside the `register` package.