WRITE UP FOR RAILWAY CROSSING STATUS WEB APP

1.Set up the eclipse and configure hibernate and required jars and runtime servers.

2.Create a database named "RailWayCross" and switch to that database.

3. A bean Class named "user" with columns "name", "email", and "password".

4. create a bean class named "admin" with columns "Name" and "password".

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

6. create a bean class named "adminhome" with columns "id", "Name", "Address", "Landmark", "Train-schedule", "person-incharge", and "status".
- "id" column is defined as an auto-increment integer with a unique index.

7. create a bean class named "favorites" with columns "Address", "id", "Name", "Landmark", "Train-time", "person-incharge", and "status".
- "id" column is defined as the primary key.

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

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

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

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

12. 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.

13. Create an addrail.jsp file containing HTML code for adding a RailWayCross crossing.

14. Create an update.jsp file containing HTML code for updating a RailWayCross 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.

15. Create an update.jsp file containing HTML code for updating the database record based on the provided form data.

16. Create a delete.jsp file containing HTML code for deleting a RailWayCross 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.

17. Create a delete.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.

18. Create a package named `adminlogin` and create a servlet named `AdminLogin`.

19. Inside the `doPost` method of the `AdminLogin` servlet:
- Get the `Name` and `password` parameters from the request.
- Establish a database connection using JDBC.
- Prepare a SQL query to select the `Name` from the `admin` table where `Name` 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.

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

22. Inside the `doPost` method of the `AddToFavorite` servlet:
- Get the `item-id` 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 `item-id` 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.

23. Create a package named `login` and create a servlet named `Login`.

24. Inside the `doPost` method of the `Login` servlet:
- Get the `Name` and `password` parameters from the request.
- Establish a database connection using JDBC.
- Prepare a SQL query to select the `Name` from the `user` table where `Name` 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 `userhome.jsp` page.
- Else:
- Print an error message indicating login failure.

25. Create a class named `RailCross` inside the `rail` package.

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

27. Generate getters and setters for the instance variables.

28. Create a servlet named `Rail-crossing` inside the `rail` package.

29. Inside the `do-Post` method of the `RailCrossing` servlet:
- Get the parameters for `name`, `address`, `landmark`, `train-schedule`, `person-incharge`, and `status` from the request.
- Create a new `Rail-Cross` object with the provided values.
- Establish a database connection using JDBC.
- Prepare a SQL statement to insert the `Rail-Cross` 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.

30. Create a class named `Member` inside the `register` package.

31. Add private instance variables for `Name`, `password`, and `email` to the `Member` class.

32. Generate getters and setters for the instance variables.

33. Create a servlet named `Register` inside the `register` package.

34.STOP