Writeup:-

Github Link:- https://github.com/akshansha9/JAVA-FSD.git

ADMIN EMAIL: Kumar@gmail.com
ADMIN PASSWORD: Kumar@123

Product Perspective :

The application will be a standalone system designed to communicate railway crossing status information to the public and enable government personnel to manage and update the crossing status. It will interact with a database to store and retrieve data.

User Classes and Characteristics :

Public Users: They will use the application to view railway crossing details, check the status, search for specific crossings, and mark crossings as favorites.

- Government Personnel: They will have access to the admin dashboard to manage railway crossings, including adding, deleting, and updating their status.

Operating Environment :

The application will be developed using Java EE and will be deployed on a suitable web server. The system will support modern web browsers.

Design and Implementation Constraints :

- The application will be implemented using Java EE technologies.
- The frontend will be developed using Java Server Pages (JSP) and HTML.
- The backend will be implemented using Servlets.
- The application will use either JDBC or Hibernate for database interaction.
- The database management system used will be MySQL.

User Documentation :

The application will be accompanied by user documentation that provides instructions on how to use the features and functionalities of the application.

System Features :

Public Features :

1. Create an Account:
   - Allow users to register and create an account with basic details like name, email, and password.Validate user input and ensure the uniqueness of email addresses.
2. User Login:
   - Provide an option for registered users to log in to the application.
3. Fetch Railway Crossings:
   - Retrieve and display the details of railway crossings from the database.
4. Display Railway Crossings with Status:
   - Display the list of railway crossings along with their current status (open or closed).
5. Search Crossings:
   - Implement a search functionality to allow users to search for specific railway crossings by name.
6. Mark Crossing as Favorite:
   - Allow users to mark a railway crossing as a favorite.
7. View Favorite Crossings:
   - Display a list of favorite railway crossings separately.

Government Features :

1. Admin Login:
   - Provide a secure login mechanism for government personnel to access the admin dashboard.
   - Authenticate users using pre-created email and password stored in the database.
2. Access Government Dashboard:
   - Allow authorized government personnel to access the admin dashboard.
3. Add Railway Crossing:
   - Provide a form for government personnel to add new railway crossings to the application.
   - Capture details such as name, address, landmark, train schedules, person in charge, and initial status.

4. Delete Railway Crossing:
    - Display a list of railway crossings and allow the deletion of selected crossings.
5. Update Crossing Status:

- Enable government personnel to update the status of a railway crossing (open or closed).

6. Navigation Options:

- Implement a navigation bar for easy access to all features and operations.

System Requirements :

Functional Requirements :

- The application should provide user authentication for both public and government users.
- The application should fetch and display the details of railway crossings, including their status.
- The application should allow users to search for specific railway crossings by name.
- Users should be able to mark railway crossings as favorites.
    - The application should have a separate view to display favorite railway crossings.
- Government personnel should be able to authenticate and access the admin dashboard.
- The admin dashboard should allow adding and deleting railway crossings.
- Government users should be able to update the status of railway crossings.
- The application should handle exceptions and validate user inputs.

Non-Functional Requirements :

- The application should have an intuitive and user-friendly interface.
    - The application should be responsive and compatible with modern web browsers.
- The system should handle concurrent user requests efficiently.
- The application should provide proper error handling and informative error messages.
- The code should follow industry best practices and be well-documented.

Appendices :

Algorithm:

1. Start the application.
2. Display the welcome screen with application name and developer details.
3. Display options for user interaction: Public or Government.

4. If the user selects "Public": 4.1. Authenticate the user by providing options to create an account or login. 4.2. Fetch details of railway crossings from the database. 4.3.
   Display the list of railway crossings along with their status. 4.4. Provide a search option to find a railway crossing by name. 4.5. Allow the user to mark railway crossings as favorites. 4.6. Display the list of favorite railway crossings separately.
5. If the user selects "Government": 5.1. Authenticate the user as an administrator using pre-created email and password. 5.2. Provide access to the government dashboard. 5.3. Implement CRUD operations for managing railway crossings: 5.3.1. Add a new railway crossing to the application. 5.3.2. Delete a railway crossing from the application. 5.3.3. Update the status of a railway crossing. 5.4. Implement search functionality to find specific railway crossings.
6. Handle exceptions and validate user input.
7. End the application.
8.

ER DIAGRAM :

RailwayCrossing

- crossingId
- name
- address
- landmark
- trainSchedules
- personInCharge
- status

Data Flow Diagram (DFD):

```
        User Interface
              |
          Controller
              |
           Service
              |
      Data Access Object
              |
          Database
================================
```