```
// Define a class to represent a railway crossing
class RailwayCrossing {
    // properties to represent the details of a railway crossing
    id;
    name;
    location;
    status; // open or closed

    // constructor to initialize a new railway crossing
    constructor(id, name, location, status) {
        this.id = id;
        this.name = name;
        this.location = location;
        this.status = status;
    }
}

// Define a class to represent a user
class User {
    // properties to represent the details of a user
    id;
    username;
    password;
    favorites; // list of favorite railway crossings

    // constructor to initialize a new user
    constructor(id, username, password) {
        this.id = id;
        this.username = username;
        this.password = password;
        this.favorites = [];
    }

    // method to mark a railway crossing as favorite
    markAsFavorite(railwayCrossing) {
        this.favorites.push(railwayCrossing);
    }

    // method to view the list of favorite railway crossings
    viewFavorites() {
        return this.favorites;
    }
}

// Define a class to represent the application
class Application {
    // properties to represent the state of the application
    users; // list of registered users
    railwayCrossings; // list of railway crossings

    // constructor to initialize the application
    constructor() {
        this.users = [];
        this.railwayCrossings = [];
    }

    // method to register a new user
    register(username, password) {
        let newUser = new User(this.users.length + 1, username,
password);
```

```
        this.users.push(newUser);
        return newUser;
    }

    // method to log in an existing user
    login(username, password) {
        for (let user of this.users) {
            if (user.username === username && user.password === password)
{
                return user;
            }
        }
        return null; // no matching user found
    }

    // method to fetch the details of the railway crossings and display
them on the frontend for the public
    fetchRailwayCrossings() {
        return this.railwayCrossings;
    }

    // method to search for railway crossings based on a query
    searchRailwayCrossings(query) {
        let results = [];
        for (let crossing of this.railwayCrossings) {
            if (crossing.name.includes(query) ||
crossing.location.includes(query)) {
                results.push(crossing);
            }
        }
        return results;
    }

    // method to add a new railway crossing to the database (admin only)
    addRailwayCrossing(name, location, status) {
        let newCrossing = new
RailwayCrossing(this.railwayCrossings.length + 1, name, location,
status);
        this.railwayCrossings.push(newCrossing);
        return newCrossing;
    }

    // method to delete a railway crossing from the database (admin only)
    deleteRailwayCrossing(id) {
        for (let i = 0; i < this.railwayCrossings.length; i++) {
            if (this.railwayCrossings[i].id === id) {
                this.railwayCrossings.splice(i, 1);
                return true; // deletion successful
            }
        }
        return false; // no matching railway crossing found
    }

     // method to update the status of a railway crossing in the database
(admin only)
     updateStatus(id, status) {
         for (let crossing of this.railwayCrossings) {
             if (crossing.id === id) {
                 crossing.status = status;
                 return true; // update successful
```

```
            }
        }
        return false; // no matching railway crossing found
    }
}
```