# UniRide

Neel Patel, Danny Garcia, Patricia Saito

Team 11

# Table of Contents

# Functional Requirements:

The application provides functionality for different types of users which are passengers and drivers. Users will be able to register through the application and have their credentials authenticated through two-factor authentication. Users will be able to manage their profiles once they are signed in to the application. The behavior changes depending on whether the user chooses to drive or ride in another person's vehicle for the day which means that the two subtypes need to be further explained. Drivers will have the ability to offer rides to other students by providing the necessary information to the riders. Drivers will be able to fix or edit their vehicle description in order to provide an accurate depiction of the vehicle they are driving. Passengers will have less functionalities as they are not the one driving the vehicle but are the target audience nonetheless and the UI will still be designed with them in mind. Passengers will be able to search through the existing database in order to select a driver that best suits their schedule and route. Once the passenger finds a driver that best suits their needs and is available, the driver will have the ability to confirm or deny the ride request. Passengers will not have the ability to make any modifications to this application such as adding or updating a listing. Finally, admins will have the ability to edit, remove, or add any listing as well as block any users. This is dependent on user feedback and reports of safety concerns or inappropriate behavior

## Functions:

User Registration/Authentication:

- Users must be able to create an account with a unique username and password.
- User verification will be used to make the application safer and more secure for its users, by ensuring that everyone registered is a student and that their location data is accurate.
- User authentication should be implemented to ensure secure access to the platform.

Profile Management:

- Users should be able to edit and update their profile information, including contact details and student ID.
- Users should only be able to view public information from other users' profiles

Offering Rides:

- Users offering rides can create listings with essential details such as pickup and dropoff locations, date, and time.
- They can specify the number of available seats.

Requesting Rides:

- Users seeking rides can search for available listings based on pickup and dropoff locations and date.

Ride Matching and Confirmation:

- Basic ride matching functionality to connect ride offers with ride requests.
- Confirmation notifications sent to both the driver and passenger upon a successful match.

Ride Schedule:

- Once ride is confirmed, users will be able to see all of their upcoming rides consolidated on one page

Ride History:

- Users can view a simple list of past ride activities, showing rides they've offered and rides they've taken.

User Profile:

- Users can view and edit their basic profile information, including contact details.

Notifications:

- Basic notification system for essential updates, such as new ride requests or ride confirmations.
- Users will also be notified of upcoming rides

Search:

- Users will be able to search for other users using username and/or email address
- Users can refine their search results based on various parameters like location and other preferences

# Identification: entities, attributes, multiplicity

1. Universities entity set

The Universities entity set represents all of the universities in the system. It contains information about each university, such as its name, location, and creation and update dates.

- Relationship(s):
    - Users: A university can have many users
- Interaction(s):
    - When a user creates an account, they must select their university.
    - Users can view a list of all universities in the system
    - Users can view information about a specific university, such as its name, location, and website
    - The relationship has a one-to-many cardinality, meaning that one university can have many users, but each user can only belong to one university

2. Users entity set

The Users entity set represents all of the users in the system. It contains information about each user, such as their username, password, email address, student ID, university ID, and creation and update dates.

- Relationship(s):
    - Universities entity set has a one-to-many cardinality.
    - Vehicles entity set has a one-to-many cardinality.
    - ScheduledRides entity set has a one-to-many cardinality.
    - RideRequests entity set has a one-to-many cardinality.
    - RideReviews entity set has a one-to-many cardinality.
    - UserPreferences entity set has a one-to-many cardinality.
- Interaction(s):
    - Users can create and manage their account information, including their username, password, email address, and student ID.
    - Users can view a list of all universities in the system and select their university
    - Users can add, edit, and delete their vehicles
    - Users can schedule, request and review rides
    - Users can manage their user preferences, such as their preferred pickup and dropoff locations

3. Rides entity set

The Rides entity set represents all of the rides in the system. It contains information about each ride, such as the driver ID, pickup and dropoff locations, date and time, number of available seats, and creation and update dates.

- Relationship(s):
    - The relationship with the Users entity set has a one-to-many cardinality
    - The relationship with the RideRequests entity set has a one-to-many cardinality
    - The relationship with the Notifications entity set has a one-to-many cardinality
- Interaction(s):
    - Users can view a list of all available rides
    - Users can request rides
    - Users can cancel rides that they have requested
    - Users can view information about a specific ride, such as the driver ID, pickup and dropoff locations, date and time, number of available seats, and status
    - Users can receive notifications about rides, such as notifications when a ride request has been accepted or when a ride is about to start

4. VehicleTypes entity set

The VehicleTypes entity set represents all of the vehicle types in the system. It contains information about each vehicle type, such as its name and creation and update dates.

- Relationship(s):
    - The relationship with the Vehicles entity set has a one-to-many cardinality
- Interaction(s):
    - Users can view a list of all vehicle types in the system
    - When adding a vehicle, users must select a vehicle type

5. Vehicles entity set

The Vehicles entity set represents all of the vehicles in the system. It contains information about each vehicle, such as the owner ID, vehicle type, plate number, color, and creation and update dates.

- Relationship(s):
    - The relationship with the Users entity set has a one-to-many cardinality

    - The relationship with the VehicleTypes entity set has a one-to-many cardinality
- Interaction(s):
    - Users can add, edit, and delete their vehicles
    - When adding a vehicle, users must select a vehicle type

6. RideRequests entity set

The RideRequests entity set represents all of the ride requests in the system. It contains information about each ride request, such as the ride ID, passenger ID, status, and creation and update dates.

- Relationships:
    - Rides: A ride request is for one ride
    - Users: A ride request is made by one user
- Interaction(s):
    - Users can request rides
    - Users can cancel rides that they have requested
    - Users can view a list of all of their ride requests

7. RideReviews entity set

The RideReviews entity set represents all of the ride reviews in the system. It contains information about each ride review, such as the ride request ID, rating, feedback, and creation date.

- Relationship(s):
    - RideRequests: A ride review is for one ride request
- Interaction(s):
    - Users can leave reviews for rides that they have requested

8. UserPreferences entity set

The UserPreferences entity set represents all of the user preferences in the system. It contains information about each user preference, such as the user ID, preference name, and preference value.

- Relationship(s):
    - Users: A user preference belongs to one user.
- Interaction(s):
    - Users can set their preferences in the app, such as their preferred pickup and dropoff locations and their preferred vehicle type.

9. ScheduledRides entity set

The ScheduledRides entity set represents all of the scheduled rides in the system. It contains information about each scheduled ride, such as the ride ID, user ID, scheduled date, and scheduled time.

- Relationship(s):
    - Rides: A scheduled ride is for one ride
    - Users: A scheduled ride is scheduled by one user
- Interaction(s):
    - Users can look at their scheduled rides in the application

10. Notifications entity set

The Notifications entity set represents all of the notifications in the system. It contains information about each notification, such as the ride ID, sender ID, receiver ID, and creation date.

- Relationship(s):
    - Rides: A notification is for one ride
    - Users: A notification is sent by one user and received by one user
- Interaction(s):
    - Users can receive notifications about rides in the app

11. RideRequestNotifications entity set

The RideRequestNotifications entity set represents all of the ride request notifications in the system. It inherits from the Notifications entity set and adds a notification_type attribute.

- Relationship(s):
    - Notifications: A ride request notification is a notification.
- Interaction(s):
    - Users can receive ride request notifications in the app.

12. RideAcceptanceNotifications entity set

The RideAcceptanceNotifications entity set represents all of the ride acceptance notifications in the system. It inherits from the Notifications entity set and adds a notification_type attribute.

- Relationships:
    - Notifications: A ride acceptance notification is a notifications

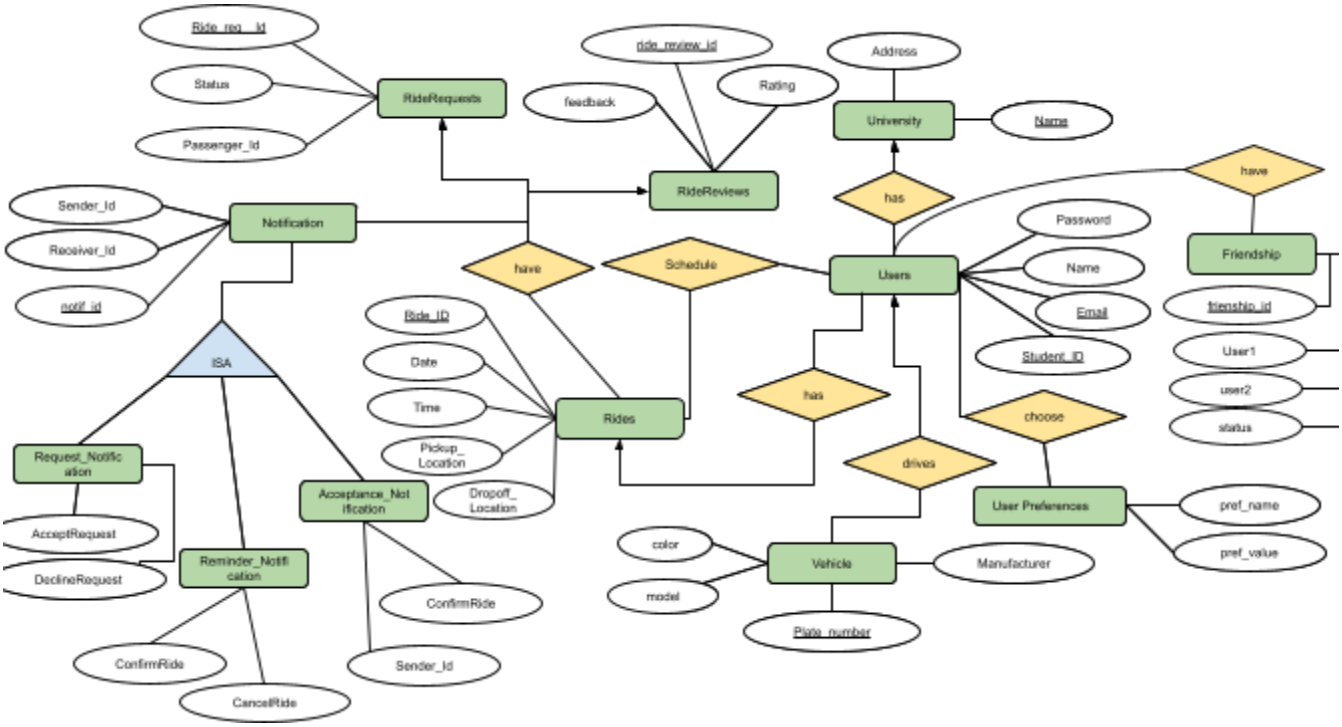
- Interaction(s): Users can receive ride acceptance notifications in the app

13. RideReminderNotifications entity set

The RideReminderNotifications entity set represents all of the ride reminder notifications in the system. It inherits from the Notifications entity set and adds a notification_type attribute.

- Relationship(s):
    - Notifications: A ride reminder notification is a notification
- Interaction(s):
    - Users can receive ride reminder notifications in the app

# ERD Diagram

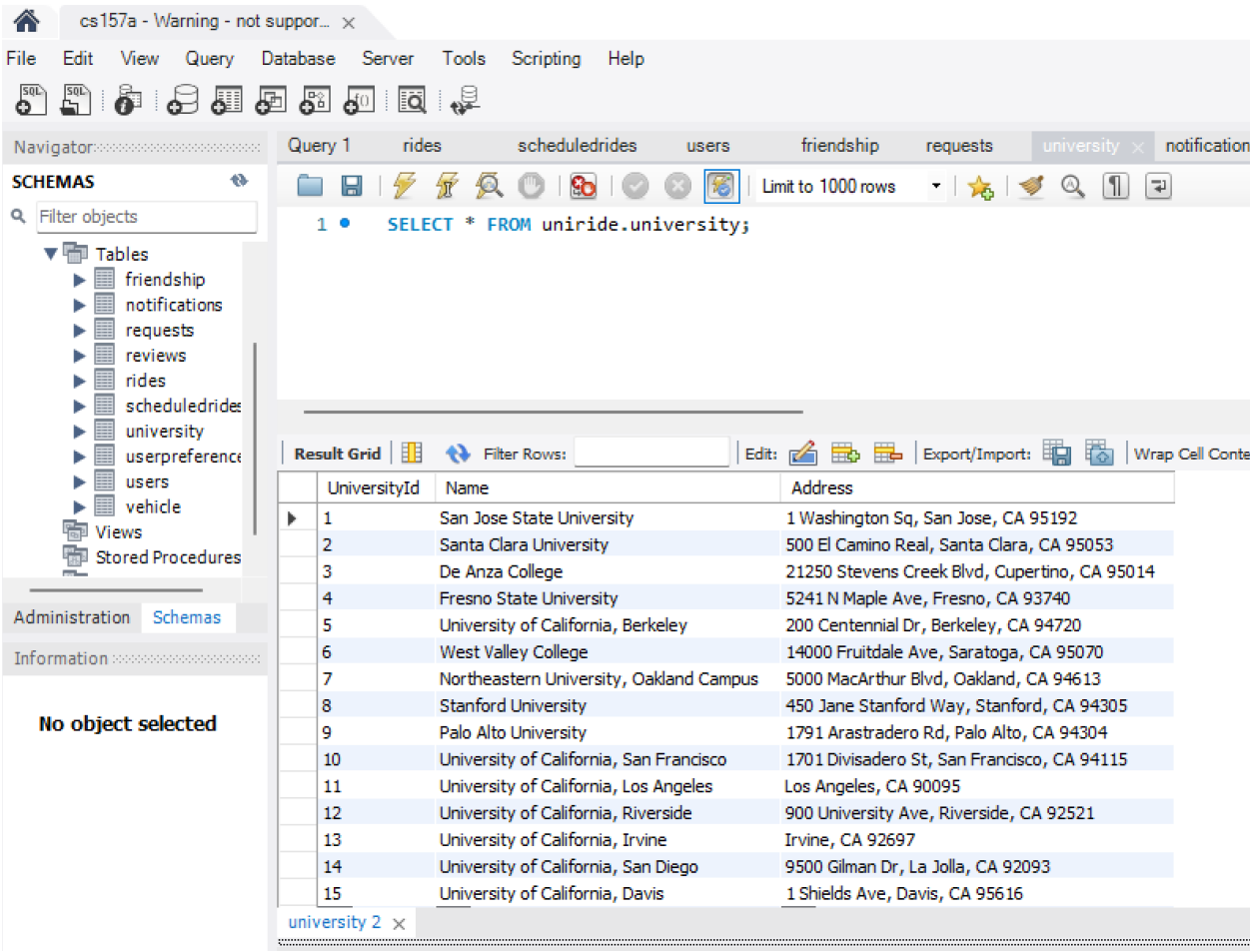

Ride_req__Id
Status
Passenger_Id
RideRequests
ride_review_Id
feedback
Rating
RideReviews
Address
University
Name
Sender_Id
Receiver_Id
notif_id
Notification
have
Schedule
has
Users
Password
Name
Email
Student_ID
have
Friendship
friendship_id
User1
user2
status
Ride_ID
Date
Time
Pickup_ Location
Dropoff_ Location
Rides
has
choose
drives
ISA
Request_Notific ation
AcceptRequest
DeclineRequest
Reminder_Notifi cation
Acceptance_Not ification
ConfirmRide
CancelRide
ConfirmRide
Sender_Id
color
model
Vehicle
Manufacturer
Plate_number
User Preferences
pref_name
pref_value

# ERD Schema

University(name, address)
Users(userId, username, password, email, is_driver, student_id, university_name)
Friendship(friendship_id, user1, user2, status)
Rides(ridesId, driver_id, pickup_location, dropoff_location, date, time)
Vehicle(user_id, plate_number, color, model, manufacturer)
RideRequests(ride_request_id, ride_id, passenger_id, status)
RideReviews(ride_request_id, rating, feedback)
UserPreferences(user_id, preference_name, preference_value)
ScheduledRides(id, ride_id, user_id, scheduled_date, scheduled_time)
Notifications(notif_id, ride_id, sender_id, receiver_id)

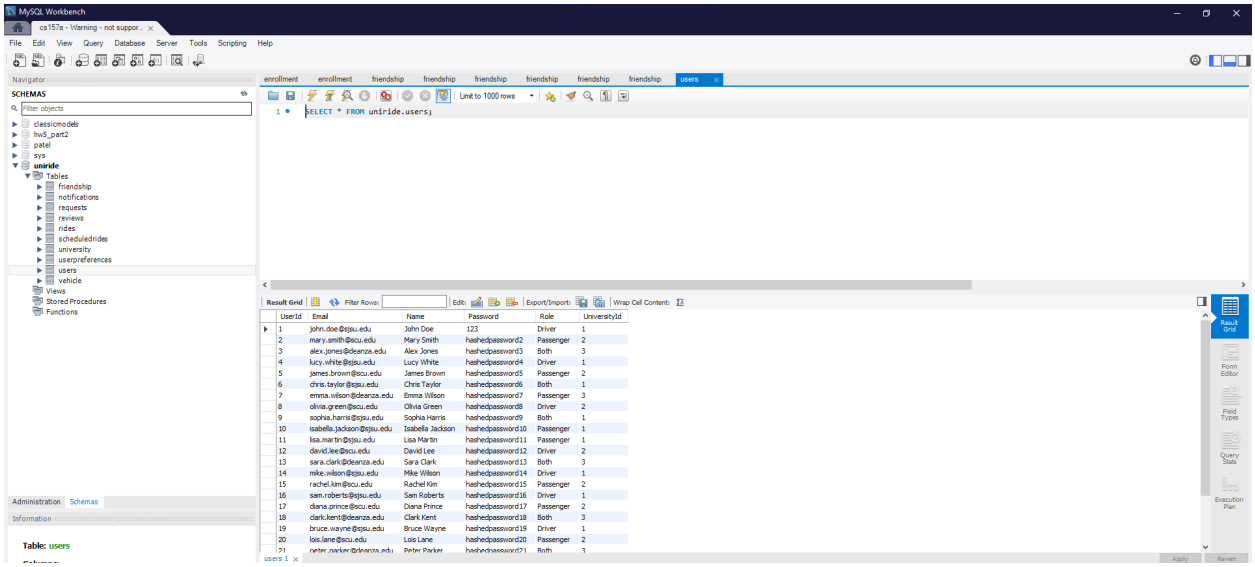
# MySQL Workbench Tables

## University

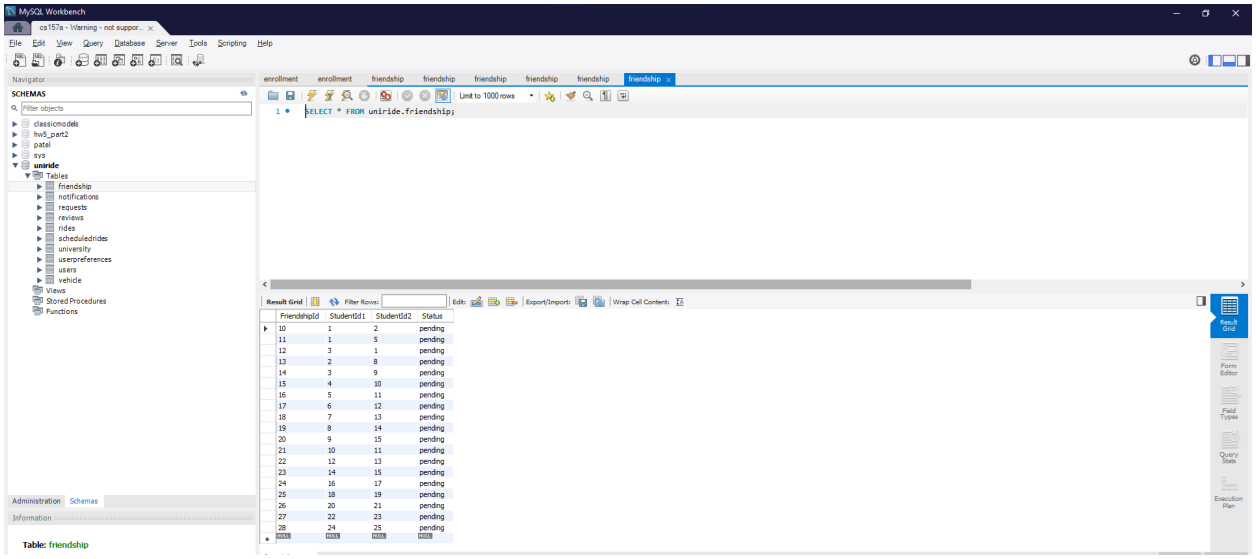



| UniversityId | Name | Address |
|---|---|---|
| 1 | San Jose State University | 1 Washington Sq, San Jose, CA 95192 |
| 2 | Santa Clara University | 500 El Camino Real, Santa Clara, CA 95053 |
| 3 | De Anza College | 21250 Stevens Creek Blvd, Cupertino, CA 95014 |
| 4 | Fresno State University | 5241 N Maple Ave, Fresno, CA 93740 |
| 5 | University of California, Berkeley | 200 Centennial Dr, Berkeley, CA 94720 |
| 6 | West Valley College | 14000 Fruitdale Ave, Saratoga, CA 95070 |
| 7 | Northeastern University, Oakland Campus | 5000 MacArthur Blvd, Oakland, CA 94613 |
| 8 | Stanford University | 450 Jane Stanford Way, Stanford, CA 94305 |
| 9 | Palo Alto University | 1791 Arastradero Rd, Palo Alto, CA 94304 |
| 10 | University of California, San Francisco | 1701 Divisadero St, San Francisco, CA 94115 |
| 11 | University of California, Los Angeles | Los Angeles, CA 90095 |
| 12 | University of California, Riverside | 900 University Ave, Riverside, CA 92521 |
| 13 | University of California, Irvine | Irvine, CA 92697 |
| 14 | University of California, San Diego | 9500 Gilman Dr, La Jolla, CA 92093 |
| 15 | University of California, Davis | 1 Shields Ave, Davis, CA 95616 |

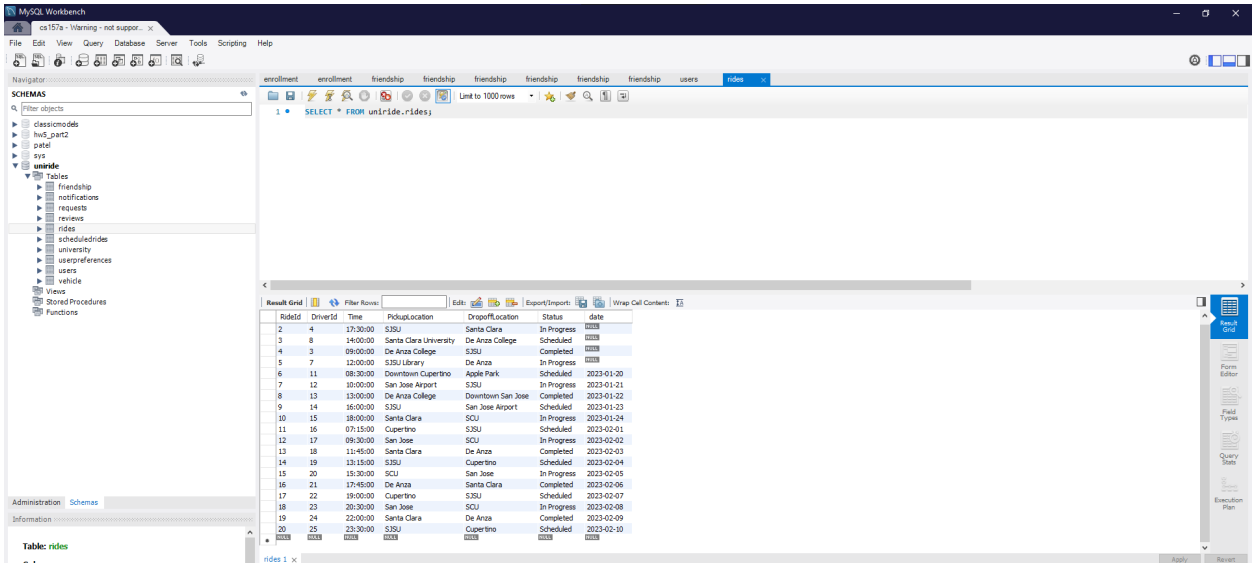
## Users



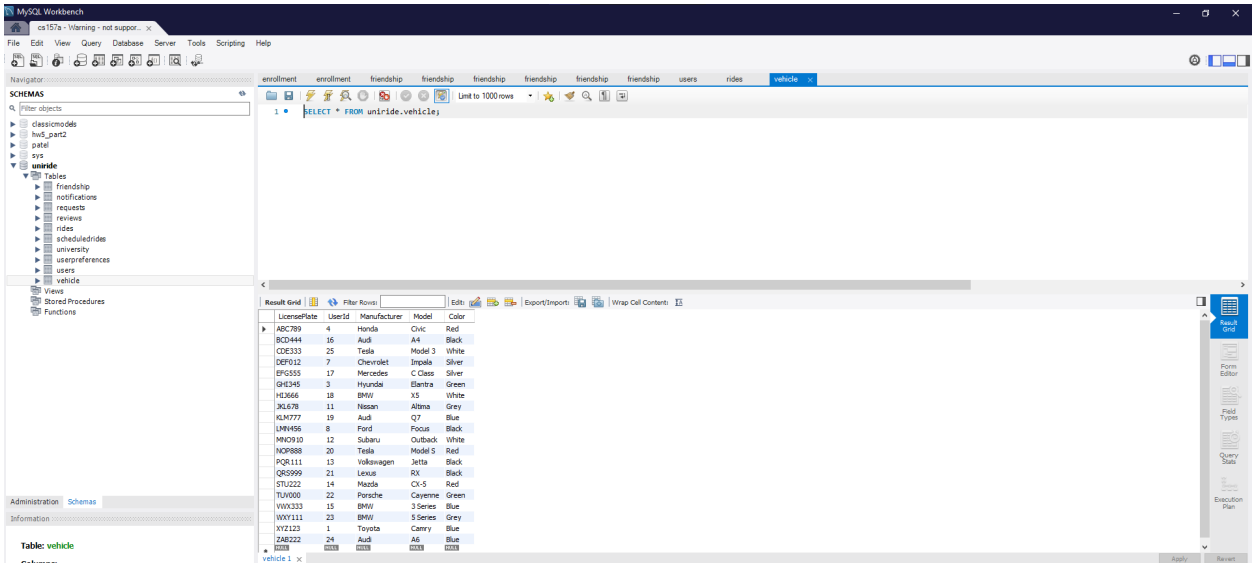

## Friendship



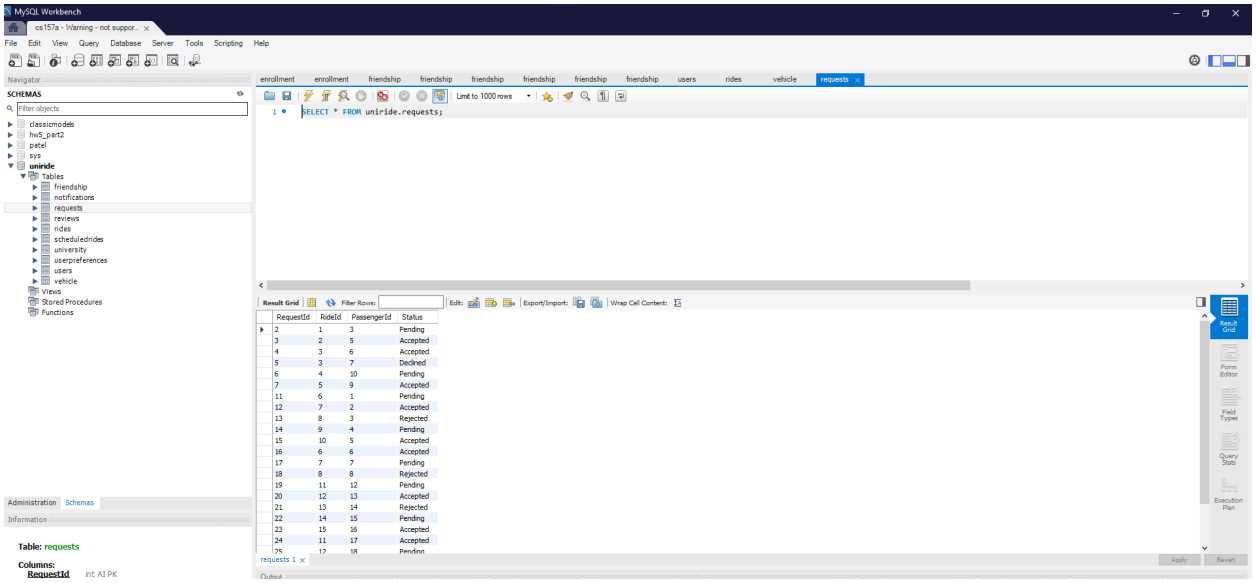
## Rides



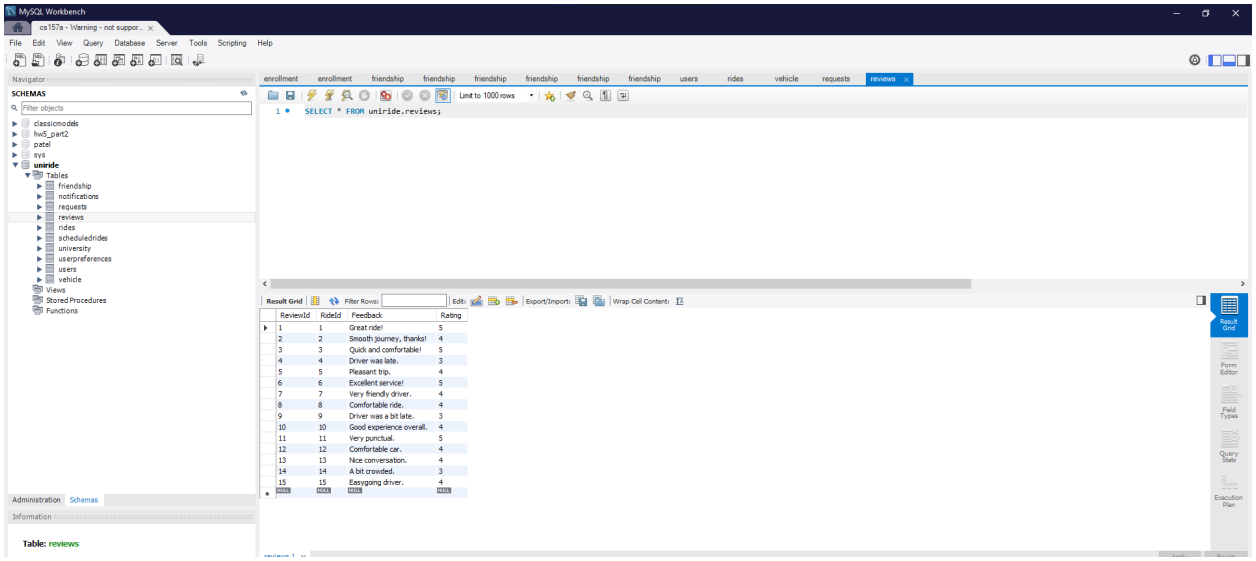

## Vehicle



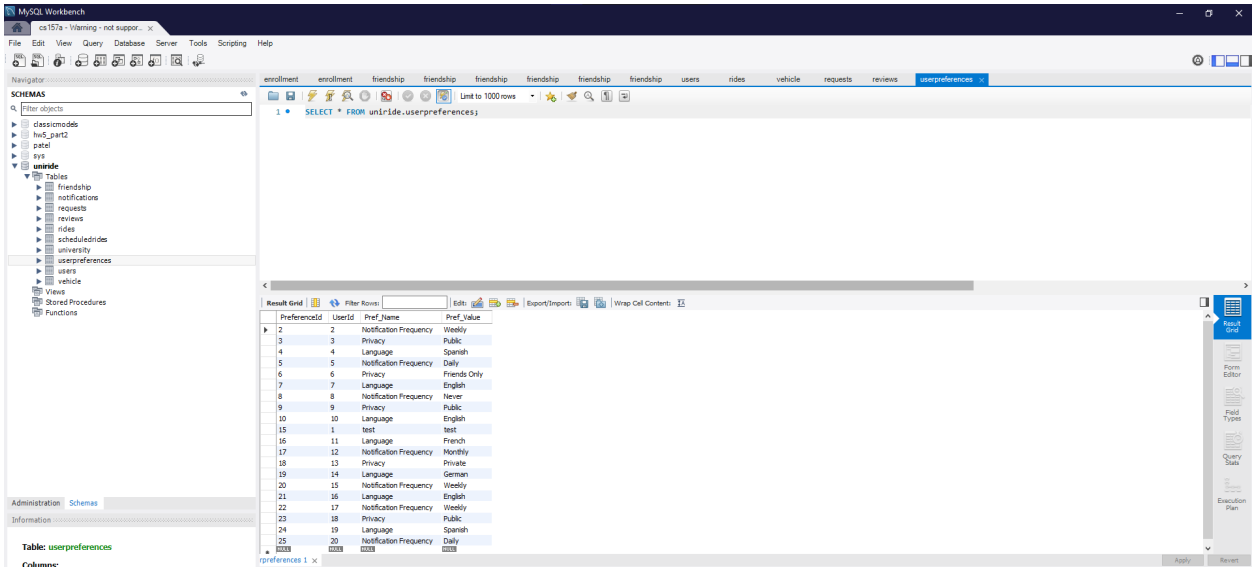
## RideRequests



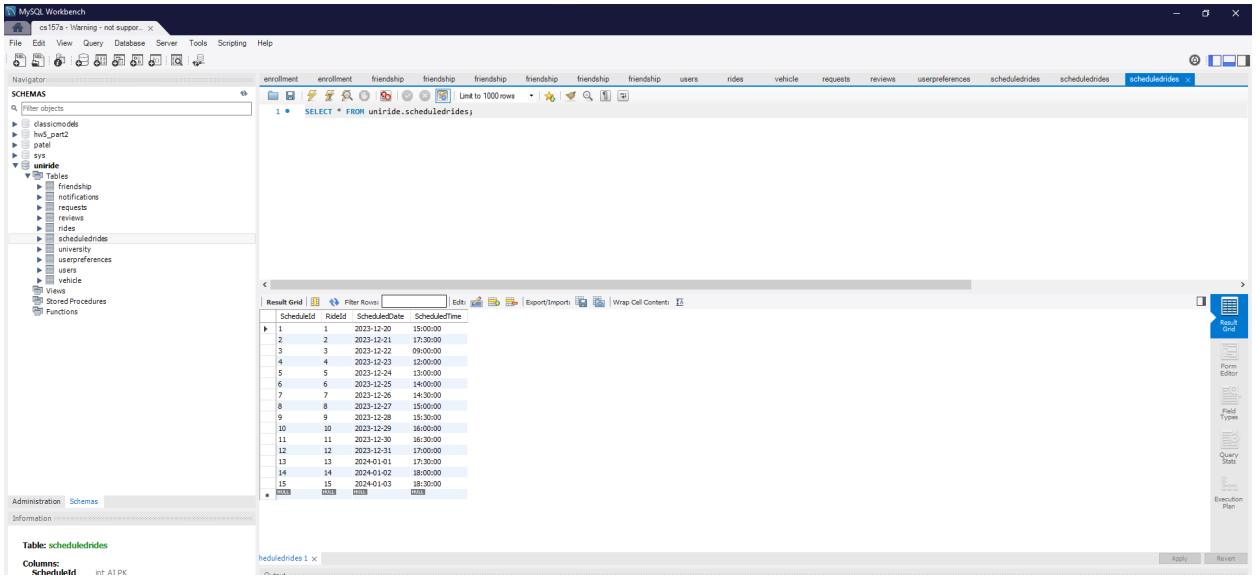

## RideReviews



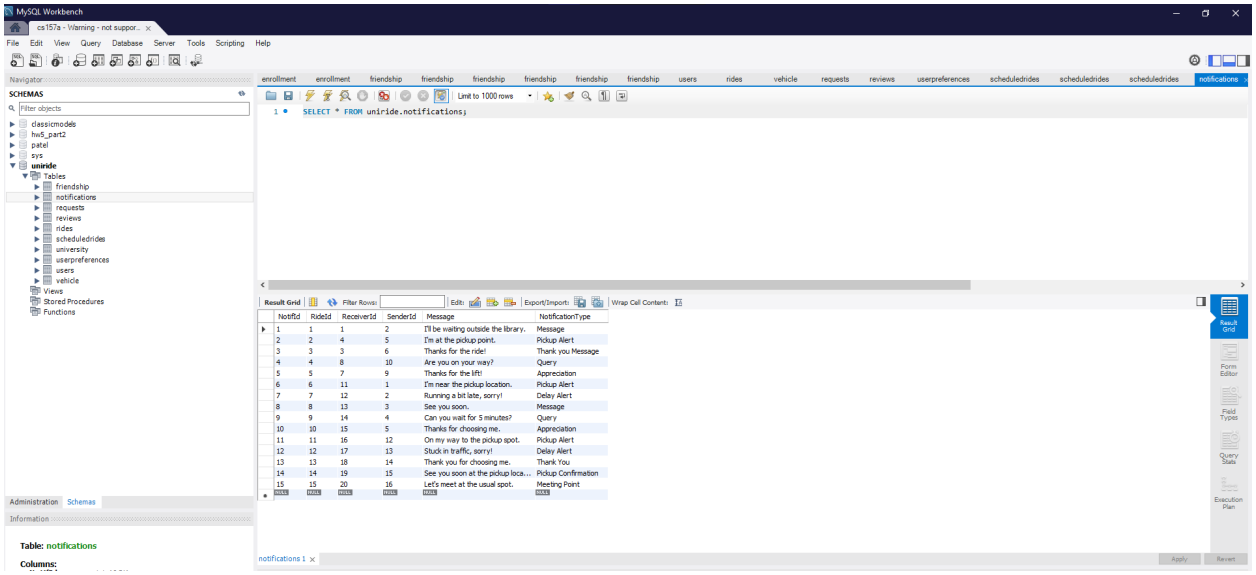
## UserPreferences



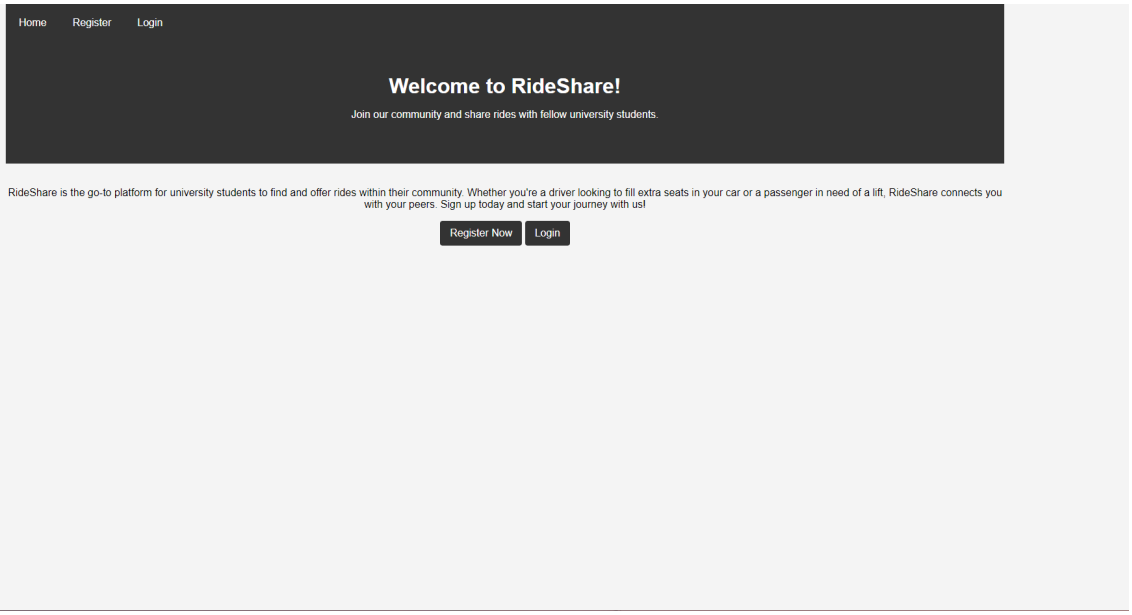

## ScheduledRides

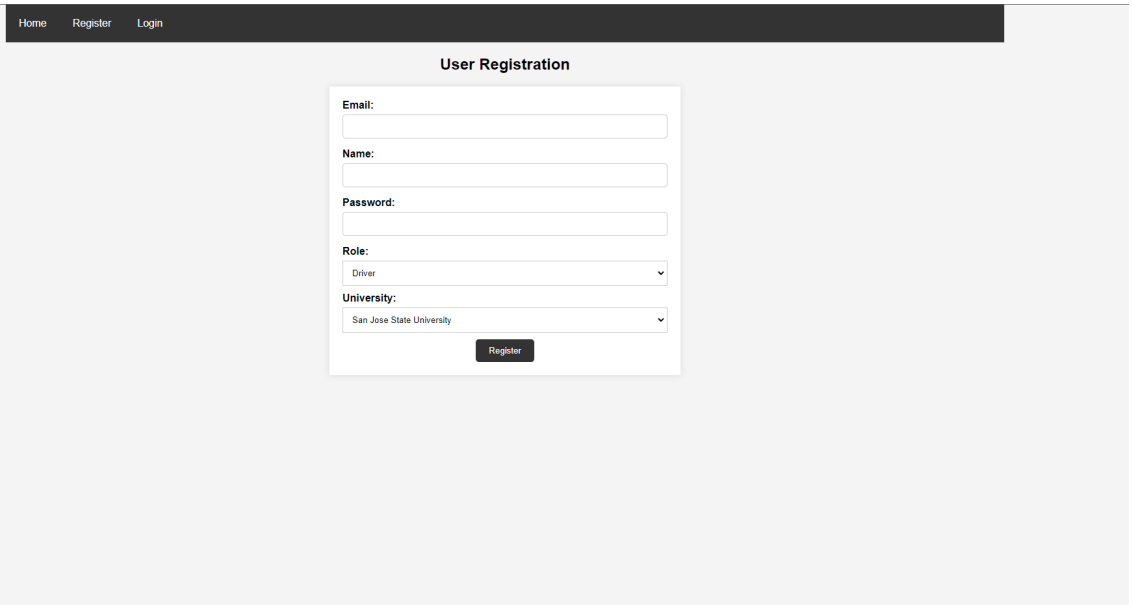


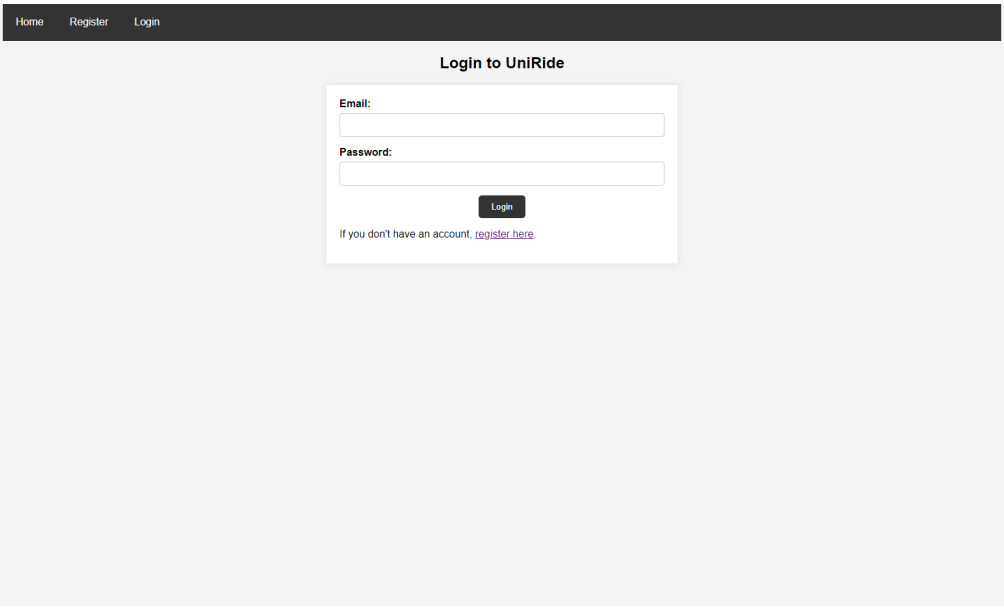
# Notifications

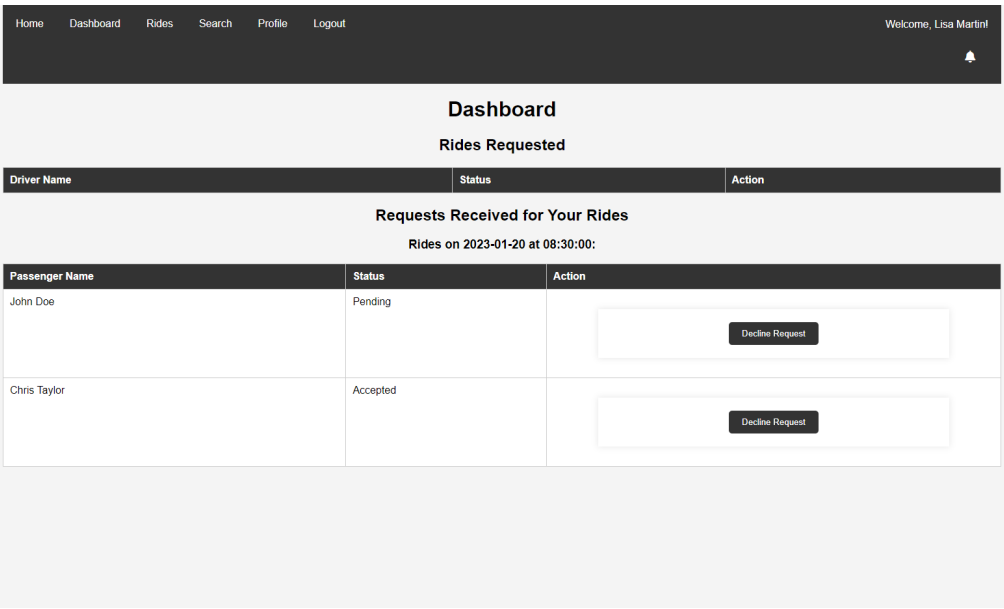


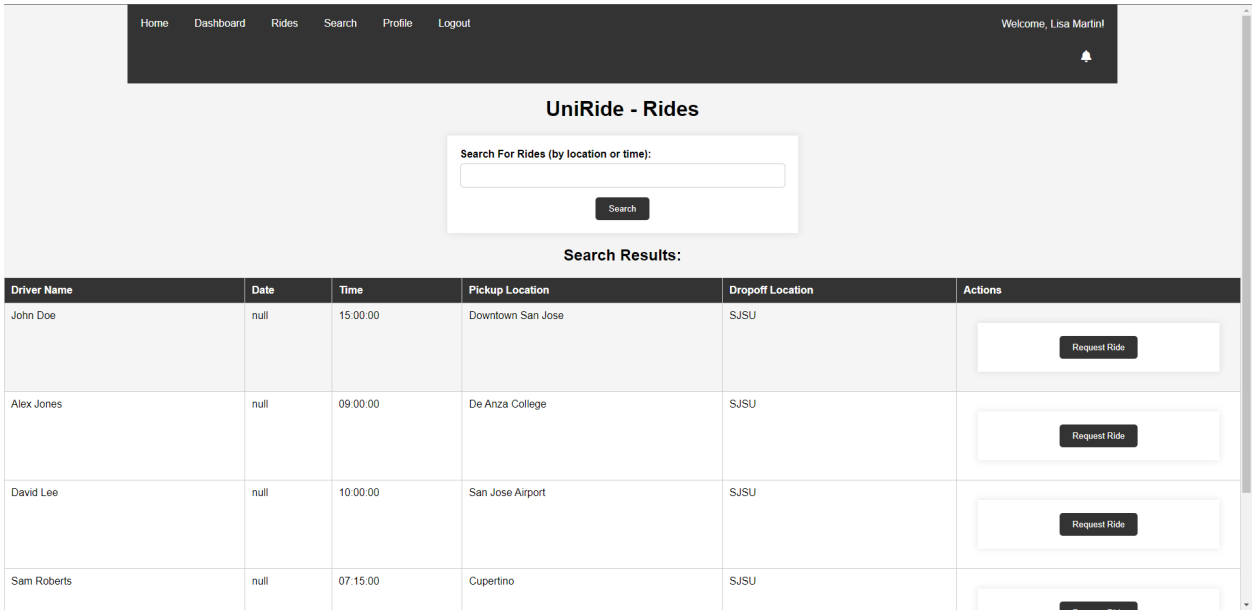
# GUI Screenshots

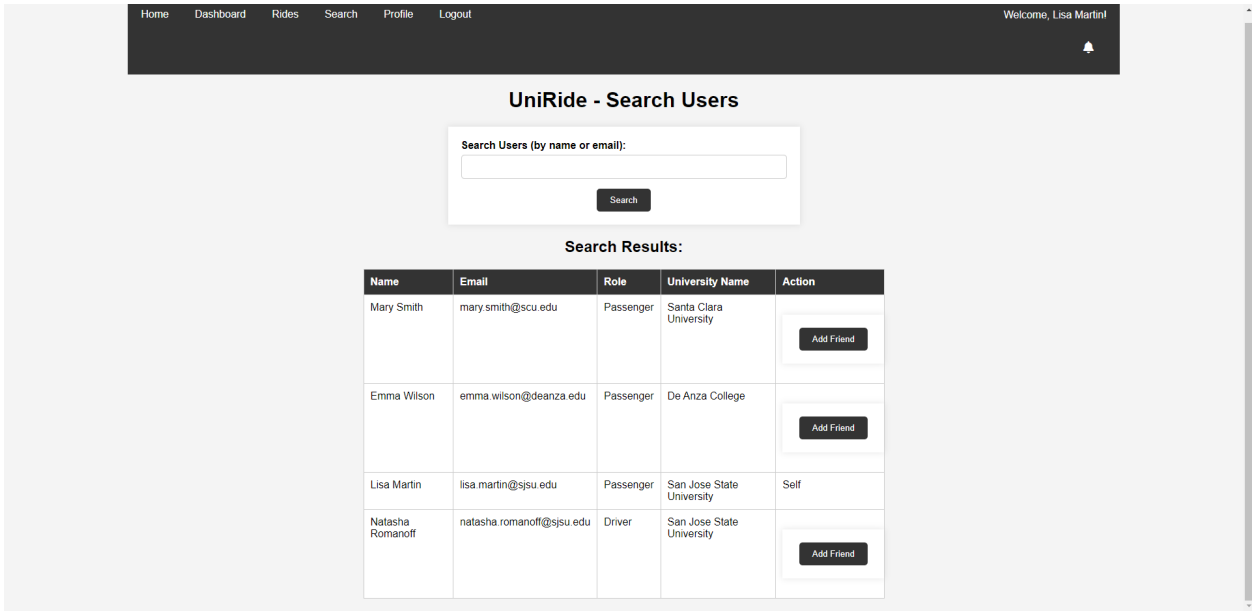


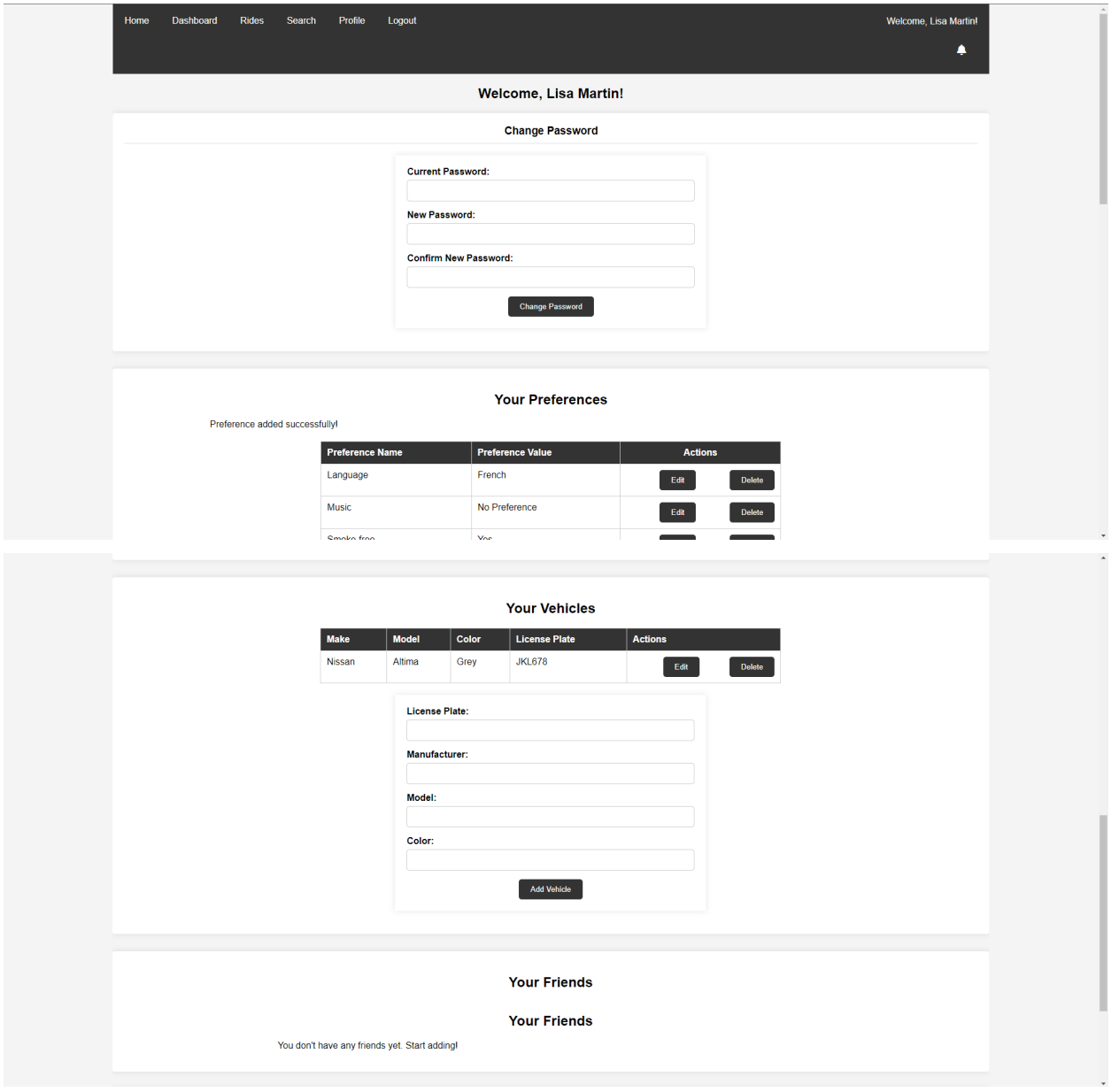




Home
Register
Login
Login to UniRide
Email:
Password:
Login
If you don't have an account, register here

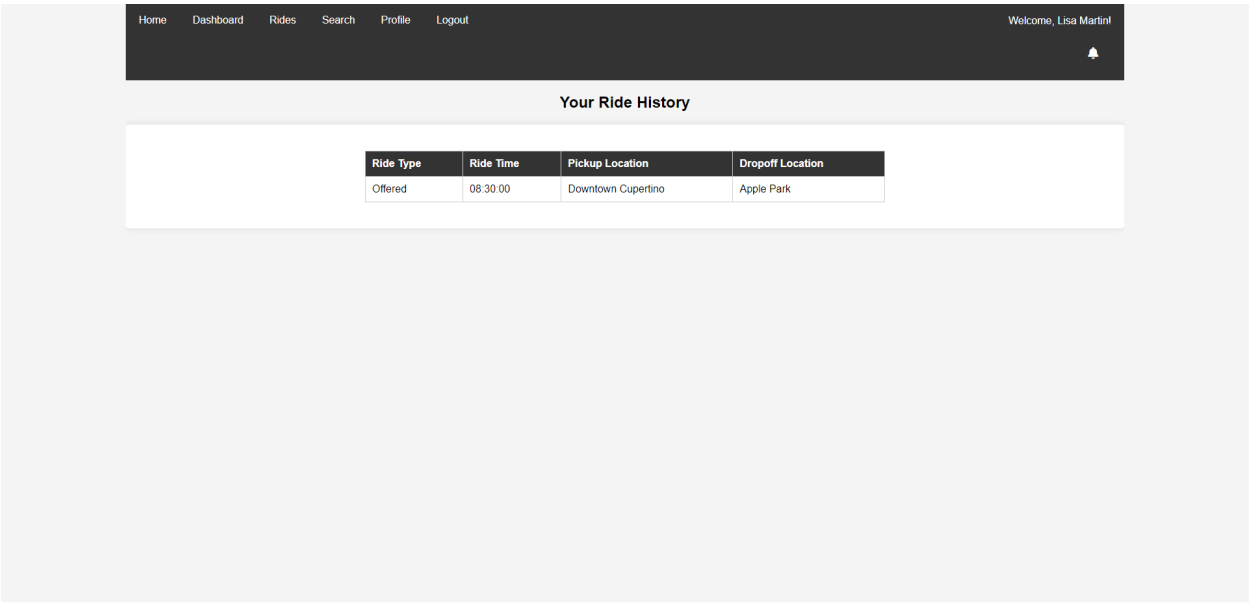

Home
Dashboard
Rides
Search
Profile
Logout
Welcome, Lisa Martin!
Dashboard
Rides Requested
Driver Name
Status
Action
Requests Received for Your Rides
Rides on 2023-01-20 at 08:30:00:
Passenger Name
Status
Action
John Doe
Pending
Decline Request
Chris Taylor
Accepted
Decline Request


Home
Dashboard
Rides
Search
Profile
Logout
Welcome, Lisa Martin!
UniRide - Rides
Search For Rides (by location or time):
Search
Search Results:
Driver Name
Date
Time
Pickup Location
Dropoff Location
Actions
John Doe
null
15:00:00
Downtown San Jose
SJSU
Request Ride
Alex Jones
null
09:00:00
De Anza College
SJSU
Request Ride
David Lee
null
10:00:00
San Jose Airport
SJSU
Request Ride
Sam Roberts
null
07:15:00
Cupertino
SJSU


Home
Dashboard
Rides
Search
Profile
Logout
Welcome, Lisa Martin!
UniRide - Search Users
Search Users (by name or email):
Search
Search Results:
Name
Email
Role
University Name
Action
Mary Smith
mary.smith@scu.edu
Passenger
Santa Clara University
Add Friend
Emma Wilson
emma.wilson@deanza.edu
Passenger
De Anza College
Add Friend
Lisa Martin
lisa.martin@sjsu.edu
Passenger
San Jose State University
Self
Natasha Romanoff
natasha.romanoff@sjsu.edu
Driver
San Jose State University
Add Friend


Home
Dashboard
Rides
Search
Profile
Logout
Welcome, Lisa Martin!
Welcome, Lisa Martin!
Change Password
Current Password:
New Password:
Confirm New Password:
Change Password
Your Preferences
Preference added successfully!
Preference Name
Preference Value
Actions
Language
French
Edit
Delete
Music
No Preference
Edit
Delete


Your Vehicles
Make
Model
Color
License Plate
Actions
Nissan
Altima
Grey
JKL678
Edit
Delete
License Plate:
Manufacturer:
Model:
Color:
Add Vehicle
Your Friends
Your Friends
You don't have any friends yet. Start adding!


Home
Dashboard
Rides
Search
Profile
Logout
Welcome, Lisa Martin!
Your Ride History
Ride Type
Ride Time
Pickup Location
Dropoff Location
Offered
08:30:00
Downtown Cupertino
Apple Park

# Lesson Learned

Neel Patel:

For me, the project was quite challenging because of the technologies we used. The backend was challenging because we used JSP and Tomcat which is very old architecture and almost out of use so I was not familiar with the code practices in this technical stack. I also found the frontend styling challenging because in my past projects, I used React with Tailwind CSS while this project was purely HTML and CSS and due to the scale of the project it made it hard to keep the files clean and organized which led to a lot of redundant code and css styles.

Patricia Saito:

Personally, my biggest difficulty was keeping up with the unfamiliar architecture we were using in this project while trying to implement something new. Keeping in communication with a team while working on project features that were often still up for debate earlier on in the project was definitely a challenge in particular. Once we got everything together on everyone's devices, we still occasionally ran into problems with who was editing what at what times, but we worked around those issues effectively. I definitely also learned a lot about using JSP and Tomcat to implement 3-tiered architecture on a single device.

Danny Garcia:

The challenges that I faced was setting up the 3-tiered architecture in the beginning as I did not have any experience with setting up a Tomcat server. In most of my past class projects, I was just expected to either work on the backend or work on the front end or also working with both of them together. However, I have never connected the two with a database so I really liked that I learned that I was able to set up a 3 tiered architecture and work first hand with how data is stored for an application or a website. Being unfamiliar with the 3-tier architecture was also one of the reasons why I ran into so many bugs while coding. Overall, the project gave me a lot of hands on experience to learn how to set up a 3 tier architecture and be able to efficiently implement it.

# How to Setup

1. Download and extract the zip file
2. Open IntelliJ Ultimate IDE and import an existing project
    a. Open the extracted folder
3. Right click and go to module settings
4. Click on project tab on the left side
    a. SDK > Add SDK > Download JDK > Choose “Oracle JDK 21.0.1” Version 21
    b. Download and select for the project
5. Next go to Libraries tab in the same window to add the SQL connector
    a. Click “+” and add Java module
    b. Choose the path to the SQL Connector JAR file
6. Add Tomact Server
    a. Go to Top Menu Bar and click on Run
    b. Click “Edit Configurations”
    c. Add “Tomcat Local Server”
    d. On the bottom of the window click “+” and add build artifact and add UniRide war
    e. Then switch to deployment tab and do the same as 6d
    f. In the same window, change the application context to “/uniride”
    g. Apply and exit
7. Run > Tomcat
8. Website should open in browser