

UniRide

Neel Patel, Danny Garcia, Patricia Saito

Team 11

Table of Contents

Functional Requirements:.....	3
Identification: entities, attributes, multiplicity.....	5
ERD Diagram.....	10
ERD Schema.....	11
MySQL Workbench Tables.....	12
GUI Screenshots.....	18
Lesson Learned.....	23
How to Setup.....	24

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

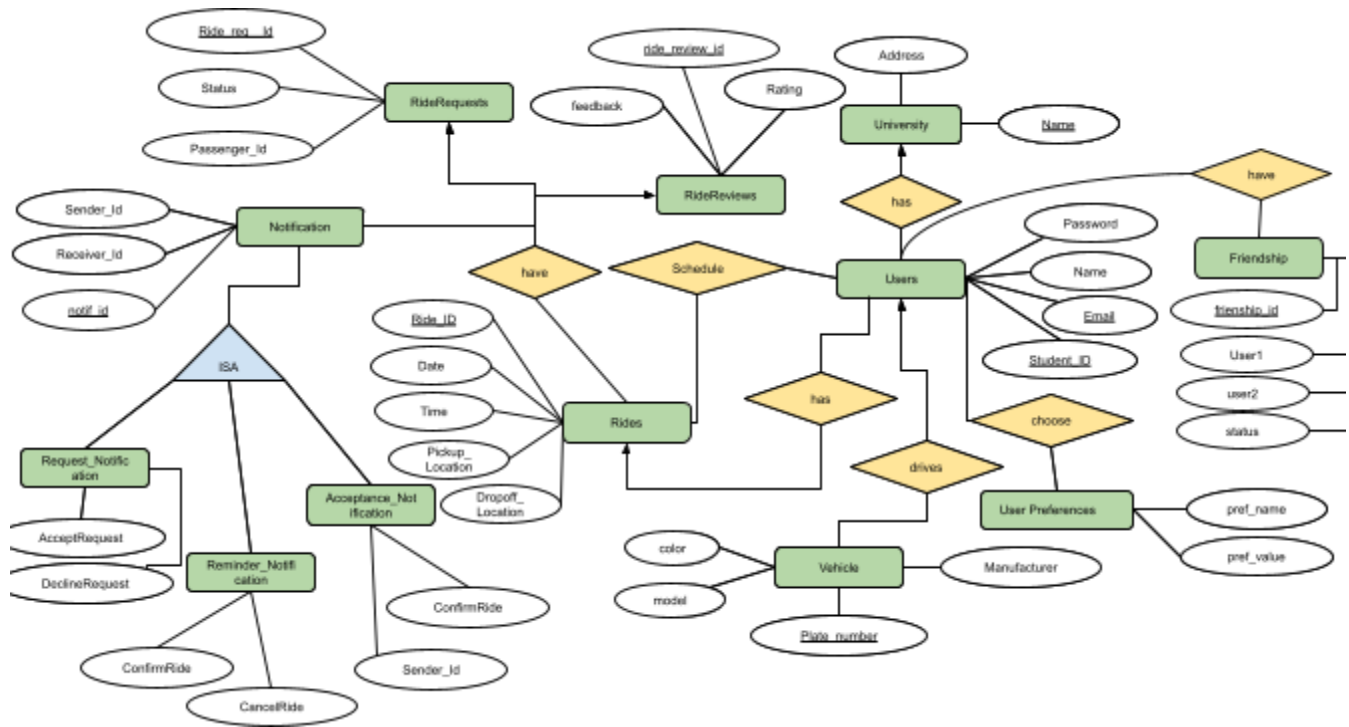
- Interaction(s): Users can receive ride acceptance notifications in the app

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



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

The screenshot shows the MySQL Workbench interface. The 'Navigator' pane on the left lists the 'uniride' database schema with various tables. The 'Query 1' editor shows the SQL query: `SELECT * FROM uniride.university;`. The 'Result Grid' pane displays the data for the 'university' table, which includes columns for 'UniversityId', 'Name', and 'Address'.

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

MySQL Workbench interface showing the 'users' table in the 'uniride' database. The table structure is as follows:

UserID	Email	Name	Password	Role	UniversityId
1	john.doe@psu.edu	John Doe	123	Driver	1
2	mary.smith@psu.edu	Mary Smith	hashedpassword2	Passenger	2
3	alex.jones@deanza.edu	Alex Jones	hashedpassword3	Both	3
4	lucy.white@psu.edu	Lucy White	hashedpassword4	Driver	1
5	james.brown@psu.edu	James Brown	hashedpassword5	Passenger	2
6	chris.taylor@psu.edu	Chris Taylor	hashedpassword6	Both	1
7	emma.wilson@deanza.edu	Emma Wilson	hashedpassword7	Passenger	3
8	olivia.green@psu.edu	Olivia Green	hashedpassword8	Driver	2
9	sophia.harris@psu.edu	Sophia Harris	hashedpassword9	Both	1
10	isabella.jackson@psu.edu	Isabella Jackson	hashedpassword10	Passenger	1
11	isa.martin@psu.edu	Isa Martin	hashedpassword11	Passenger	1
12	david.lee@psu.edu	David Lee	hashedpassword12	Driver	2
13	sara.clark@deanza.edu	Sara Clark	hashedpassword13	Both	3
14	mike.wilson@psu.edu	Mike Wilson	hashedpassword14	Driver	1
15	rachel.lee@psu.edu	Rachel Lee	hashedpassword15	Passenger	2
16	sam.roberts@psu.edu	Sam Roberts	hashedpassword16	Driver	1
17	diana.prince@psu.edu	Diana Prince	hashedpassword17	Passenger	2
18	clark.kent@deanza.edu	Clark Kent	hashedpassword18	Both	3
19	bruce.wayne@psu.edu	Bruce Wayne	hashedpassword19	Driver	1
20	lois.lane@psu.edu	Lois Lane	hashedpassword20	Passenger	2
21	robert.harker@deanza.edu	Robert Harker	hashedpassword21	Both	1

Friendship

MySQL Workbench interface showing the 'friendship' table in the 'uniride' database. The table structure is as follows:

FriendshipId	StudentId1	StudentId2	Status
10	1	2	pending
11	1	5	pending
12	3	1	pending
13	2	8	pending
14	3	9	pending
15	4	10	pending
16	5	11	pending
17	6	12	pending
18	7	13	pending
19	8	14	pending
20	9	15	pending
21	10	11	pending
22	12	13	pending
23	14	15	pending
24	16	17	pending
25	18	19	pending
26	20	21	pending
27	22	23	pending
28	24	25	pending

Rides

MySQL Workbench interface showing the 'rides' table in the 'uniride' database. The table contains 20 rows of ride data. The columns are: RideId, DriverId, Time, PickupLocation, DropoffLocation, Status, and date.

RideId	DriverId	Time	PickupLocation	DropoffLocation	Status	date
1	8	14:00:00	Santa Clara University	De Anza College	Scheduled	2023-01-20
2	4	17:30:00	SJSU	Santa Clara	In Progress	2023-01-21
3	8	09:00:00	De Anza College	SJSU	Completed	2023-01-21
4	3	09:00:00	SJSU Library	De Anza	In Progress	2023-01-20
5	7	12:00:00	Downtown Cupertino	Apple Park	Scheduled	2023-01-20
6	11	08:30:00	San Jose Airport	SJSU	In Progress	2023-01-21
7	12	10:00:00	De Anza College	Downtown San Jose	Completed	2023-01-22
8	13	13:00:00	SJSU	San Jose Airport	Scheduled	2023-01-23
9	14	16:00:00	Santa Clara	SCU	In Progress	2023-01-24
10	15	18:00:00	Cupertino	SJSU	Scheduled	2023-02-01
11	16	07:15:00	San Jose	SCU	In Progress	2023-02-02
12	17	09:30:00	Santa Clara	De Anza	Completed	2023-02-03
13	18	11:45:00	SJSU	Cupertino	Scheduled	2023-02-04
14	19	13:15:00	San Jose	San Jose	In Progress	2023-02-05
15	20	15:30:00	De Anza	Santa Clara	Completed	2023-02-06
16	21	17:45:00	Cupertino	SJSU	Scheduled	2023-02-07
17	22	19:00:00	SCU	San Jose	In Progress	2023-02-08
18	23	20:30:00	Santa Clara	De Anza	Completed	2023-02-09
19	24	22:00:00	SJSU	Cupertino	Scheduled	2023-02-10
20	25	23:30:00	SJSU	Cupertino	Scheduled	2023-02-10

Vehicle

MySQL Workbench interface showing the 'vehicle' table in the 'uniride' database. The table contains 24 rows of vehicle data. The columns are: LicensePlate, UserId, Manufacturer, Model, and Color.

LicensePlate	UserId	Manufacturer	Model	Color
ABC789	4	Honda	Civic	Red
BCD444	16	Audi	A4	Black
CDE333	25	Tesla	Model 3	White
DEF012	7	Chevrolet	Impala	Silver
FGH555	17	Mercedes	C-Class	Silver
GHI245	3	Hyundai	Elantra	Green
HJK666	18	BMW	X5	White
JKL777	11	Nissan	Altima	Grey
KLM777	19	Audi	Q7	Blue
LMN456	8	Ford	Focus	Black
MNO910	12	Subaru	Outback	White
NPQ666	20	Tesla	Model S	Red
PQR111	13	Volkswagen	Jetta	Black
QRS999	21	Lexus	RX	Black
STU222	14	Mazda	CX-5	Red
TUV000	22	Porsche	Cayenne	Green
VWX333	15	BMW	3 Series	Blue
XYZ111	23	BMW	5 Series	Grey
YZ123	1	Toyota	Corolla	Blue
ZAB222	24	Audi	A6	Blue

RideRequests

MySQL Workbench interface showing the 'requests' table in the 'uniride' database. The table structure is as follows:

RequestId	RideId	PassengerId	Status
1	1	3	Pending
2	2	5	Accepted
3	3	6	Accepted
4	3	7	Declined
5	4	10	Pending
6	4	10	Pending
7	5	9	Accepted
11	6	1	Pending
12	7	2	Accepted
13	8	3	Rejected
14	9	4	Pending
15	10	5	Accepted
16	6	6	Accepted
17	7	7	Pending
18	8	8	Rejected
19	11	12	Pending
20	12	13	Accepted
21	13	14	Rejected
22	14	15	Pending
23	15	16	Accepted
24	11	17	Pending
25	17	18	Pending

RideReviews

MySQL Workbench interface showing the 'reviews' table in the 'uniride' database. The table structure is as follows:

ReviewId	RideId	Feedback	Rating
1	1	Great ride!	5
2	2	Smooth journey, thank!	4
3	3	Quick and comfortable!	5
4	4	Driver was late.	3
5	5	Pleasant trip.	4
6	6	Excellent service!	5
7	7	Very friendly driver.	4
8	8	Comfortable ride.	4
9	9	Driver was a bit late.	3
10	10	Good experience overall.	4
11	11	Very punctual.	5
12	12	Comfortable car.	4
13	13	Nice conversation.	4
14	14	A bit crowded.	3
15	15	Easygoing driver.	4

UserPreferences

The screenshot shows the MySQL Workbench interface with the 'userpreferences' table selected in the 'uniride' database. The table contains 25 rows of user preferences.

PreferenceId	UserId	Pref_Name	Pref_Value
2	2	Notification Frequency	Weekly
3	3	Privacy	Public
4	4	Language	Spanish
5	5	Notification Frequency	Daily
6	6	Privacy	Friends Only
7	7	Language	English
8	8	Notification Frequency	Never
9	9	Privacy	Public
10	10	Language	English
15	1	test	test
16	11	Language	French
17	12	Notification Frequency	Monthly
18	13	Privacy	Private
19	14	Language	German
20	15	Notification Frequency	Weekly
21	16	Language	English
22	17	Notification Frequency	Weekly
23	18	Privacy	Public
24	19	Language	Spanish
25	20	Notification Frequency	Daily

ScheduledRides

The screenshot shows the MySQL Workbench interface with the 'scheduledrides' table selected in the 'uniride' database. The table contains 15 rows of scheduled ride data.

ScheduledId	RideId	ScheduledDate	ScheduledTime
1	1	2023-12-20	15:00:00
2	2	2023-12-21	17:30:00
3	3	2023-12-22	09:00:00
4	4	2023-12-23	12:00:00
5	5	2023-12-24	13:00:00
6	6	2023-12-25	14:00:00
7	7	2023-12-26	14:30:00
8	8	2023-12-27	15:00:00
9	9	2023-12-28	15:30:00
10	10	2023-12-29	16:00:00
11	11	2023-12-30	16:30:00
12	12	2023-12-31	17:00:00
13	13	2024-01-01	17:30:00
14	14	2024-01-02	18:00:00
15	15	2024-01-03	18:30:00

Notifications

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema, with the 'notifications' table selected under the 'uniride' database. The main window shows the SQL query editor with the query: `SELECT * FROM uniride.notifications;`. Below the query editor, the 'Result Grid' displays the data from the 'notifications' table. The table has columns: NotifId, RideId, ReceiverId, SenderId, Message, and NotificationType. The data is as follows:

NotifId	RideId	ReceiverId	SenderId	Message	NotificationType
1	1	1	2	I'll be waiting outside the library.	Message
2	2	4	5	I'm at the pickup point.	Pickup Alert
3	3	3	6	Thanks for the ride!	Thank you Message
4	4	8	10	Are you on your way?	Query
5	5	7	9	Thanks for the lift!	Appreciation
6	6	11	1	I'm near the pickup location.	Pickup Alert
7	7	12	2	Running a bit late, sorry!	Delay Alert
8	8	13	3	See you soon.	Message
9	9	14	4	Can you wait for 5 minutes?	Query
10	10	15	5	Thanks for choosing me.	Appreciation
11	11	16	12	On my way to the pickup spot.	Pickup Alert
12	12	17	13	Stuck in traffic, sorry!	Delay Alert
13	13	18	14	Thank you for choosing me.	Thank You
14	14	19	15	See you soon at the pickup loca...	Pickup Confirmation
15	15	20	16	Let's meet at the usual spot.	Meeting Point

GUI Screenshots

The image displays two screenshots of the UniRide web application interface.

Top Screenshot: Home Page

- Header:** A dark navigation bar containing links for [Home](#), [Register](#), and [Login](#).
- Content:**
 - Welcome to RideShare!**
 - Join our community and share rides with fellow university students.
 - RideShare is the go-to platform for university students to find and offer rides within their community. Whether you're a driver looking to fill extra seats in your car or a passenger in need of a lift, RideShare connects you with your peers. Sign up today and start your journey with us!
 - Buttons: [Register Now](#) and [Login](#).

Bottom Screenshot: User Registration

- Header:** A dark navigation bar containing links for [Home](#), [Register](#), and [Login](#).
- Form:**
 - User Registration**
 - Email:**
 - Name:**
 - Password:**
 - Role:** (dropdown menu)
 - University:** (dropdown menu)
 - [Register](#) button

[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	<div>Decline Request</div>
Chris Taylor	Accepted	<div>Decline Request</div>

HomeDashboardRidesSearchProfileLogout

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	<div>Request Ride</div>
Alex Jones	null	09:00:00	De Anza College	SJSU	<div>Request Ride</div>
David Lee	null	10:00:00	San Jose Airport	SJSU	<div>Request Ride</div>
Sam Roberts	null	07:15:00	Cupertino	SJSU	<div></div>

HomeDashboardRidesSearchProfileLogout

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	<div>Add Friend</div>
Emma Wilson	emma.wilson@deanza.edu	Passenger	De Anza College	<div>Add Friend</div>
Lisa Martin	lisa.martin@sjsu.edu	Passenger	San Jose State University	Self
Natasha Romanoff	natasha.romanoff@sjsu.edu	Driver	San Jose State University	<div>Add Friend</div>

HomeDashboardRidesSearchProfileLogout

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	<div>Edit</div> <div>Delete</div>
Music	No Preference	<div>Edit</div> <div>Delete</div>
Smells from	Van	<div></div> <div></div>

Your Vehicles

Make	Model	Color	License Plate	Actions
Nissan	Altima	Grey	JKL678	<div>Edit</div> <div>Delete</div>

License Plate:

Manufacturer:

Model:

Color:

Add Vehicle

Your Friends

Your Friends

You don't have any friends yet. Start adding!

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 Tomcat 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