UW Carpool Application

11/21/2018

Presented by:

Jiatong He, Shuting Lian, Zhilun Chang, Zizheng Jiang

UNIVERSITY OF WATERLOO

Outline

- Introduction
- Application Demo
- Architecture & Design Patterns
- Technical Challenges
- Future Improvement

Introduction

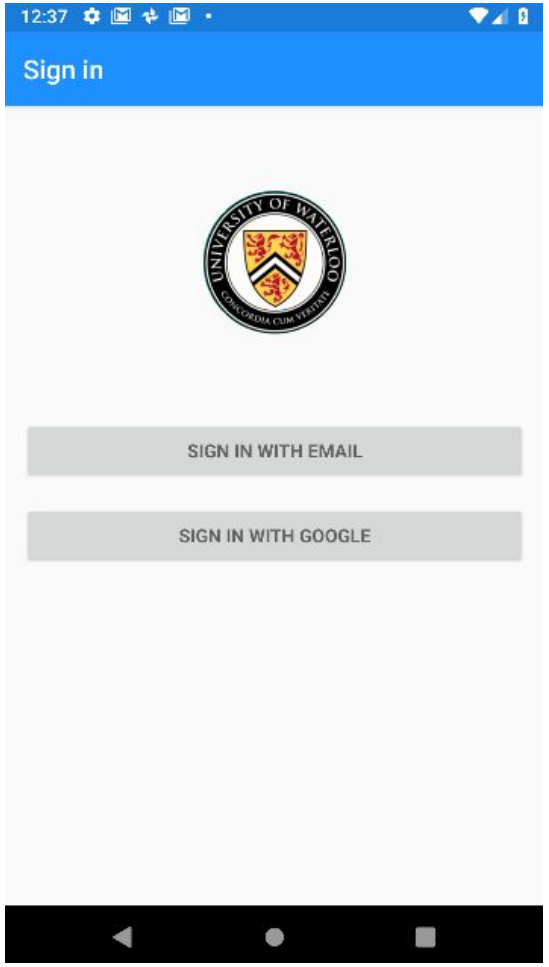
- **UW Carpool** Application – Android app

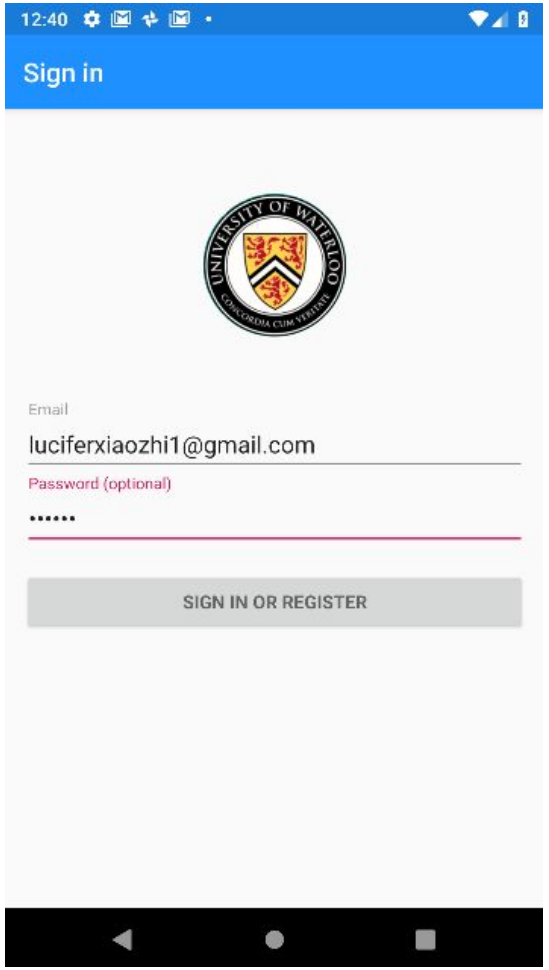
  - Organizes carpooling information for both **drivers** and **passengers.**

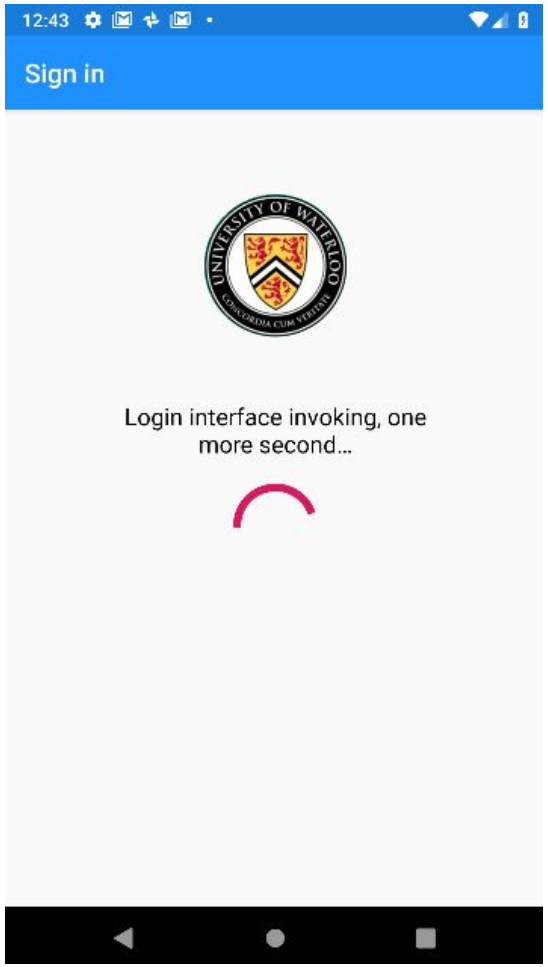
    - **Drivers** provide travel information, which includes *Departure City*, *Arrival City*, *Departure Address*, *Arrival Address*, *Departure Date & Time*, *Phone Number*, *Vacancies* and *Price*.

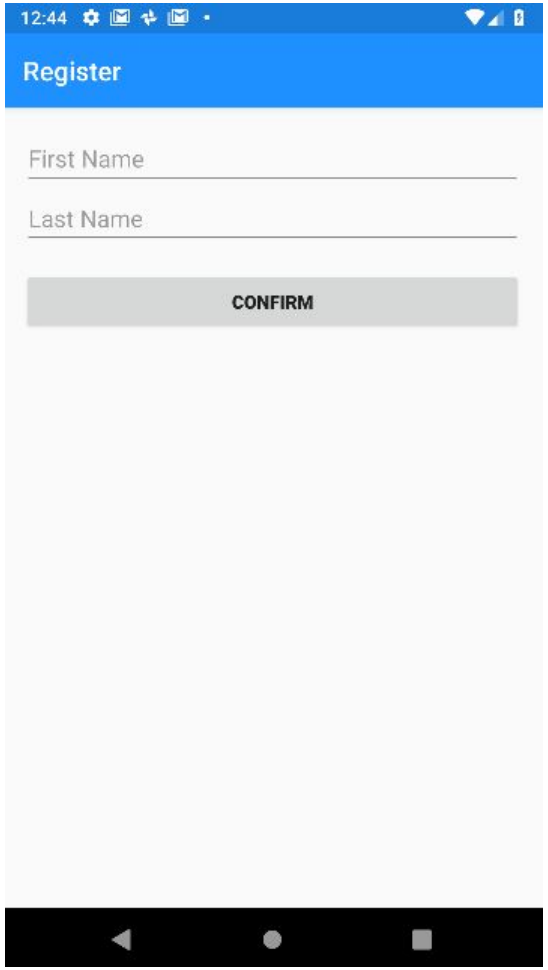
    - **Passengers** enter *Departure City*, *Arrival City,* and *Date* and then search for available information.

Demo



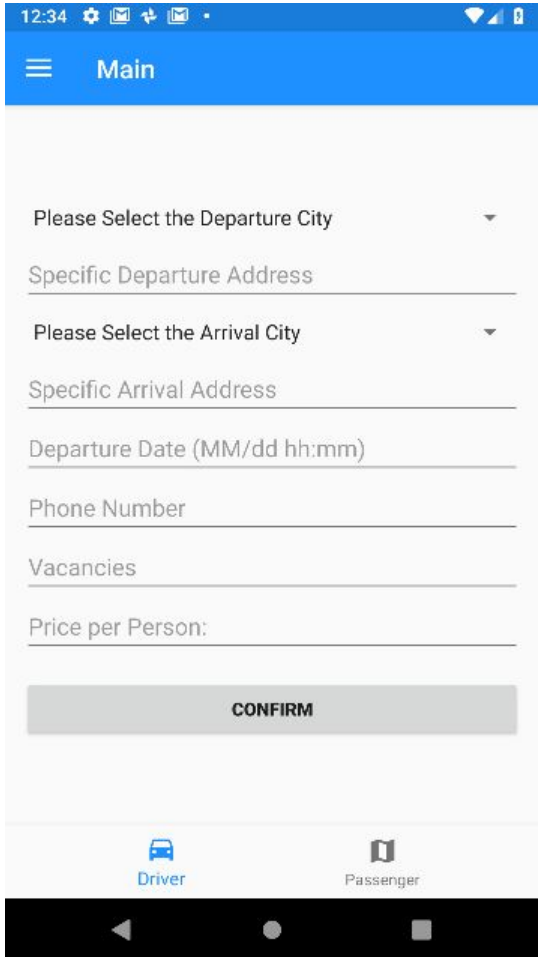


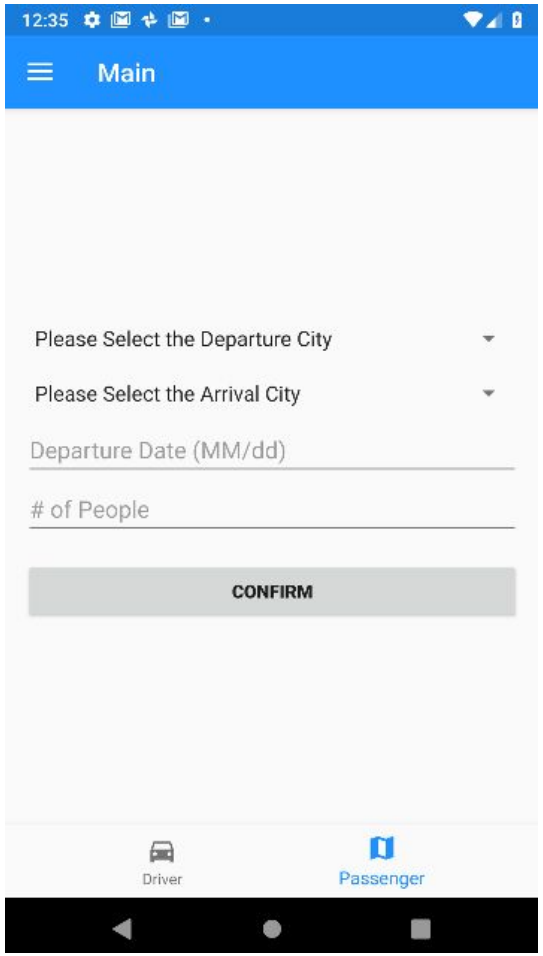


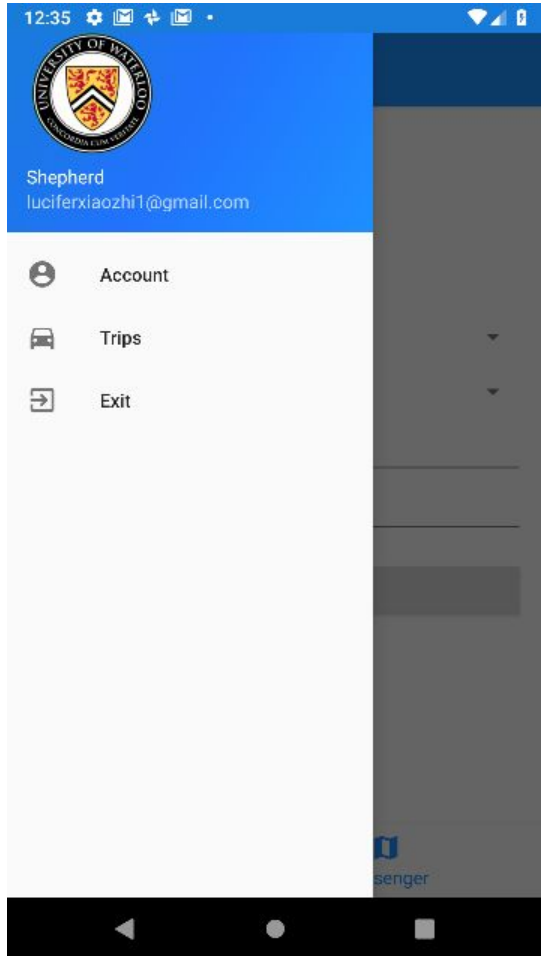


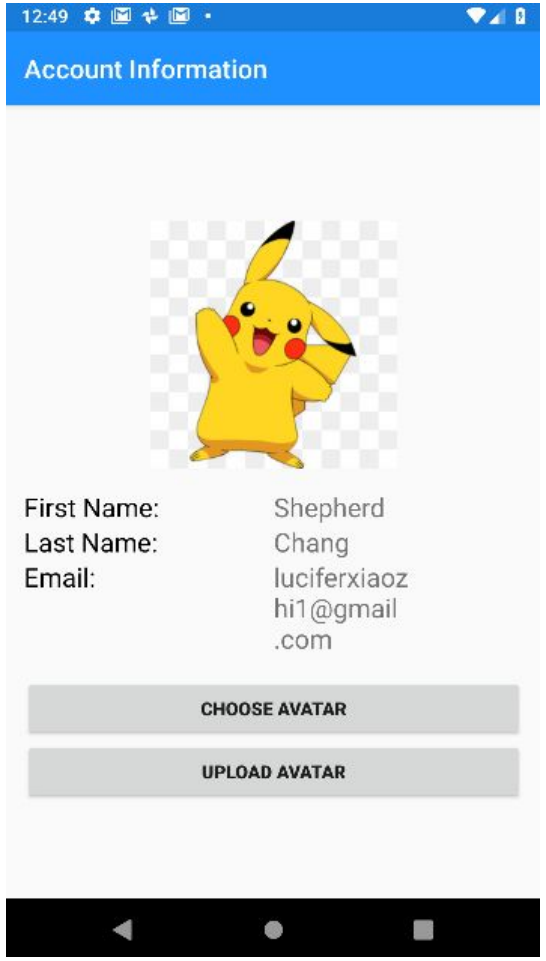
UNIVERSITY OF WATERLOO

Demo









UNIVERSITY OF WATERLOO

Application Catalog

```
➜ waterloocarpool git:(master) ✗ tree
.
├── backend
│   ├── UserInfo.kt
│   ├── api
│   │   ├── Auth.kt
│   │   ├── Store.kt
│   │   └── TaskDecorators.kt
│   └── bean
│       ├── Trip.kt
│       └── User.kt
├── frontend
│   ├── activities
│   │   ├── AccountInfoActivity.kt
│   │   ├── EmailLoginActivity.kt
│   │   ├── GoogleLoginActivity.kt
│   │   ├── LoginActivity.kt
│   │   ├── MainActivity.kt
│   │   ├── RegisterActivity.kt
│   │   ├── TripDetailsActivity.kt
│   │   └── TripsList.kt
│   ├── adapters
│   │   └── UserAdapter.kt
│   └── fragments
│       ├── DriverFragment.kt
│       └── PassengerFragment.kt
└── global
    └── Constants.kt
```

```
➜ res git:(master) ✗ tree
.
├── drawable
│   ├── ic_account_circle.xml
│   ├── ic_driver_black_24dp.xml
│   ├── ic_exit.xml
│   ├── ic_launcher_background.xml
│   ├── ic_passenger_black_24dp.xml
│   ├── side_nav_bar.xml
│   └── trip_line_background.xml
├── drawable-v24
│   └── ic_launcher_foreground.xml
├── layout
│   ├── activity_account_info.xml
│   ├── activity_email_login.xml
│   ├── activity_google_login.xml
│   ├── activity_login.xml
│   ├── activity_main.xml
│   ├── activity_register.xml
│   ├── activity_trip_details.xml
│   ├── activity_trips_list.xml
│   ├── app_bar_main.xml
│   ├── fragment_driver.xml
│   ├── fragment_passenger.xml
│   ├── line_trip.xml
│   └── nav_header_main.xml
├── menu
│   ├── activity_main_drawer.xml
│   └── navigation.xml
├── mipmap-anydpi-v26
│   ├── ic_launcher.xml
│   └── ic_launcher_round.xml
├── mipmap-hdpi
│   ├── ic_launcher.png
│   ├── ic_launcher_foreground.png
│   └── ic_launcher_round.png
├── mipmap-mdpi
│   ├── ic_launcher.png
│   ├── ic_launcher_foreground.png
│   └── ic_launcher_round.png
├── mipmap-xhdpi
│   ├── ic_launcher.png
│   ├── ic_launcher_foreground.png
│   └── ic_launcher_round.png
├── mipmap-xxhdpi
│   ├── ic_launcher.png
│   ├── ic_launcher_foreground.png
│   └── ic_launcher_round.png
├── mipmap-xxxhdpi
│   ├── ic_launcher.png
│   ├── ic_launcher_foreground.png
│   └── ic_launcher_round.png
└── values
    ├── colors.xml
    ├── dimens.xml
    ├── strings.xml
    └── styles.xml
```

UNIVERSITY OF WATERLOO

Design Patterns

Model-View-Presenter design pattern: It is the most basic pattern on Android platform.

Singleton design pattern: Each user in our application will have one and only one account and their username implemented by **Auth** and **Register**.

Data Access Object design pattern: The **Parcelable** in Android offers the data access object interface.

Technical Challenges - Recycler View

Why do we have this challenge?

When getting data from the database, the application itself doesn't know how many instances will be acquired, so an adapter is needed to display the data in real time.

1. Recycler View is much better than List View:
   - Load the displayed content in real time.
   - Less memory usage and faster response.

2. Needs to design each line's layout for this view. Needs to be monitored and ensure that the old data is cleared every time we generate a new adapter.

Technical Challenges - Parcelable

Why do we have this challenge?

When user select a carpool provided by a driver from the recycler view, we need to pass some additional information to the TripDetails Layout.

1. Parcelable or Serializable?

- We choose parcelable because the data is already in our memory.
- Parcelable is more efficiency and suitable for our application. (We don't need to transfer data by network and the data doesn't need to be saved).

2. If the layout use parcelable, it may cause conflicts if we want to display data in this layout through other ways.

Technical Challenges - Database (NoSQL)

Why do we have this challenge?

We need to search in the database based on the information entered by the passengers.

1. Need to add indexes for some attributes to make sure some query can work correctly.

2. NoSQL doesn’t support range query based on timestamp type. So we use the filter to optimize the database query statement.

Future improvement

To make our application more **user-friendly**, we are currently implementing some other details such as user profiles functionality (change/upload their avatar). We also want to let users know their current **trips information** by communicating with the databases, but this means that our database needs to be redesigned, which may cause all of activities which communicate with the database need to be rewritten. We are still working on it and considering whether there is a better way to achieve this function.


UNIVERSITY OF
WATERLOO
Thank you!