

ASF – אסף



הדרך הקלה לאוניברסיטה

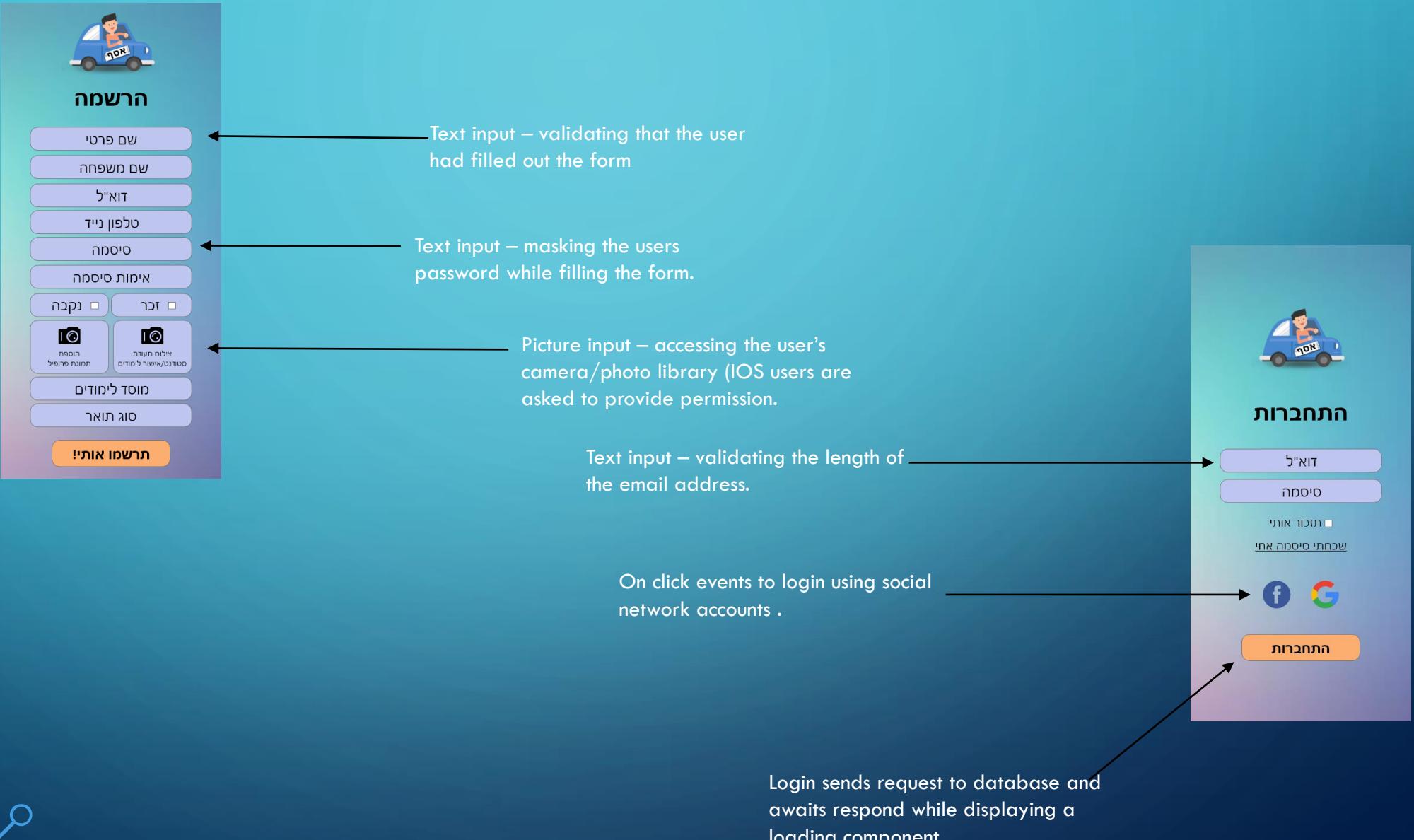
<https://github.com/motidahari/AsafApp.git>

*הריפורזיטורי פרטி מוחש לשגיבות דעתך, מצורף ריפורזיטורי אחר המשמש להרצאה בלבד בפלטפורמת Expo:

<https://github.com/motidahari/expo>

*במידה ותרצו להיחשף לקוד המלא נשלח לכם אותו באופן פרטי.

REGISTRATION AND LOGIN

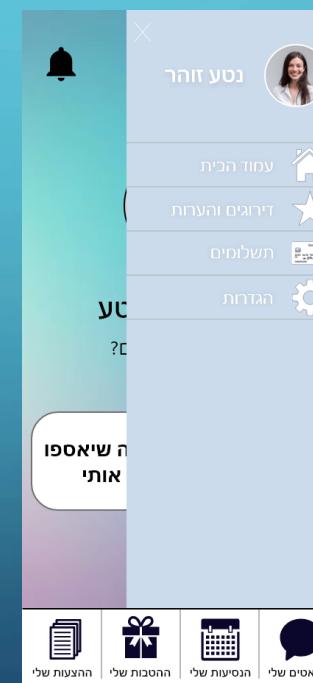


HOME PAGE AND PROFILE

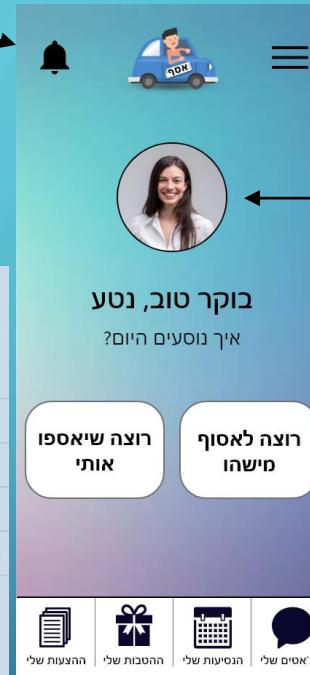
Sends a request to the database to update the listing of the user.



Navigator button to the notifications screen part of the header as well



Navigation popup menu part of the header component to switch between screens.



Username and avatar are displayed with data retrieved from the database.

Each user has two behaviors, one as a driver and the other as a passenger.

Footer quick navigation bar, used across the whole app

ADDING NEW DRIVE

Adding the drive to the specific time, date, location, num of available seats and price.



הוסף נסעה

מתי? DD/MM/YYYY

באיזה שעון? HH:MM

מאייפה? נסעה ממקום/כתובת

לאן? נסעה ממקום/כתובת

מספר מושבים?

כמה עליה לנסעה?

מחיר ממוצע לנסיעה דומה: 7 ש"ח

שפור נסעה זו כנסעה קבועה

פרסם נסעה

On click posts the new data to the database. After the update was inserted a success popup appears.

Driver can add a certain drive to be a regular route he's taking. This will be displayed in his upcoming drives screen.



DRIVER SUGGESTION SCREEN

After adding a new drive, the driver might get counter offers from the passengers. He can accept or decline or send a new price offer.

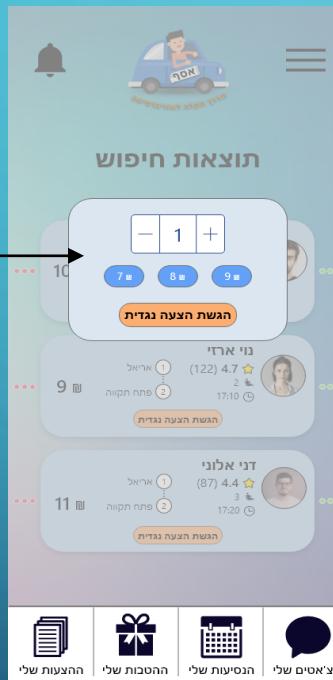
If the driver had accepted the offer this popup will appear and will navigate to the chat which includes the driver and the passenger with him.



On click posts the new data to the database.

SEARCHING A DRIVE AS A PASSENGER

Counter offer popup allows to use a pre-calculated price (according to the driver's price) and manually increase/decrease price.
Clicking the button in the bottom part send the request and updates the database for the driver.



Passenger's filter drives as they want.

Displays the relevant drives requested, with the time, number of seats available, driver info and price.
Passenger can approve or disapprove the drive suggested.

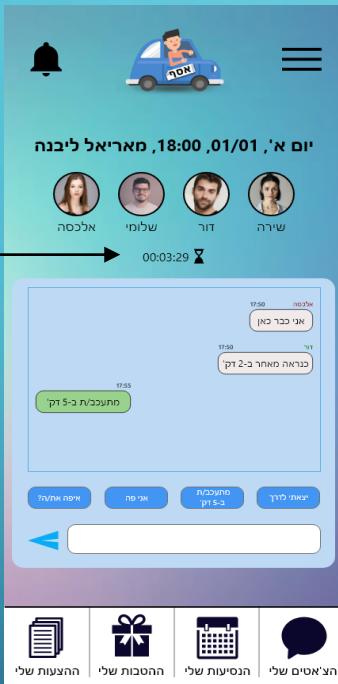


Clicking the counter offer button will display a popup where you can send the driver a different price.

On click requests relevant drive specified to the time ,date , origin and destination requested. While displaying a loading popup.

CHAT SCREENS

The chat info remains “alive” while the drive hadn’t been complete. After the drive was complete the chat will disappear from the user upcoming drives chat. After the drive was complete the passenger can leave a feedback and rating to the driver



Displays all upcoming drives that had been accepted with their information.



Displays the relevant drives requested, with the time, number of seats available, driver info and price. Passenger can approve or disapprove the drive suggested.

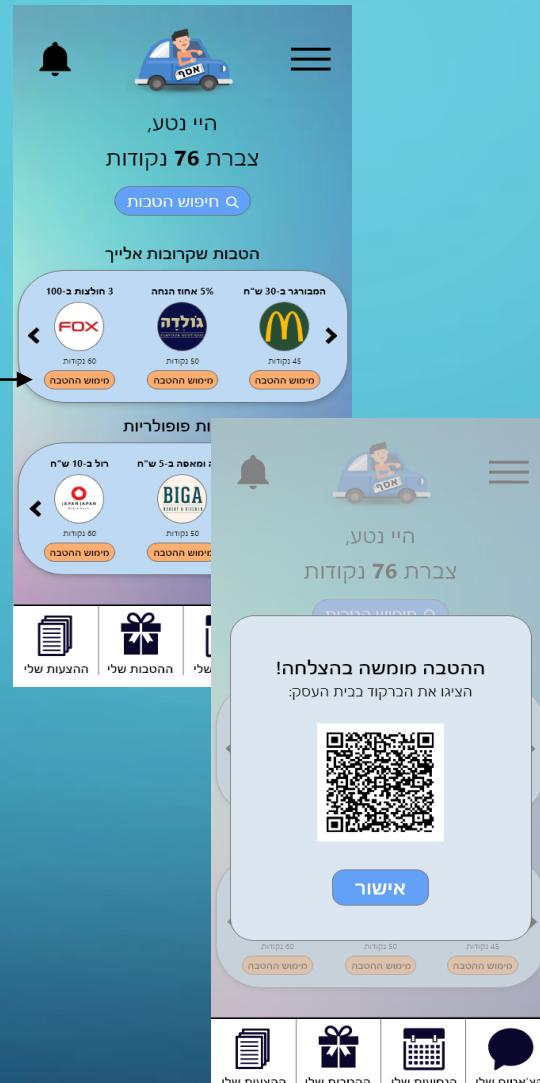
Message templates for quicker reply. When sending the message a request is sent to the database to update the content of the chat and render it to the chatbox.

Each chat contains users information retrieved from the database which includes their profile information.

PAYMENTS COUPONS

Drivers and passengers collect points according to their amount of use in the application and as well as their rating being the driver/passenger.

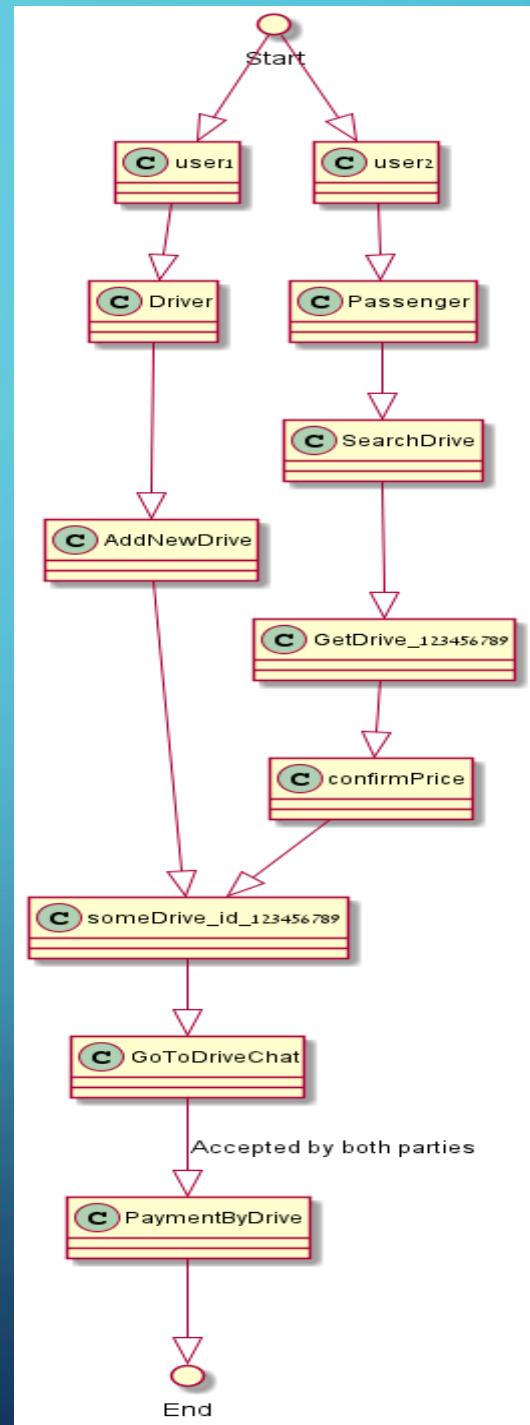
Clicking to redeem the coupon will display a barcode popup that the user will display at the store. This sends a request to the database ensuring that the user has enough credit to redeem it and then it will display the barcode.



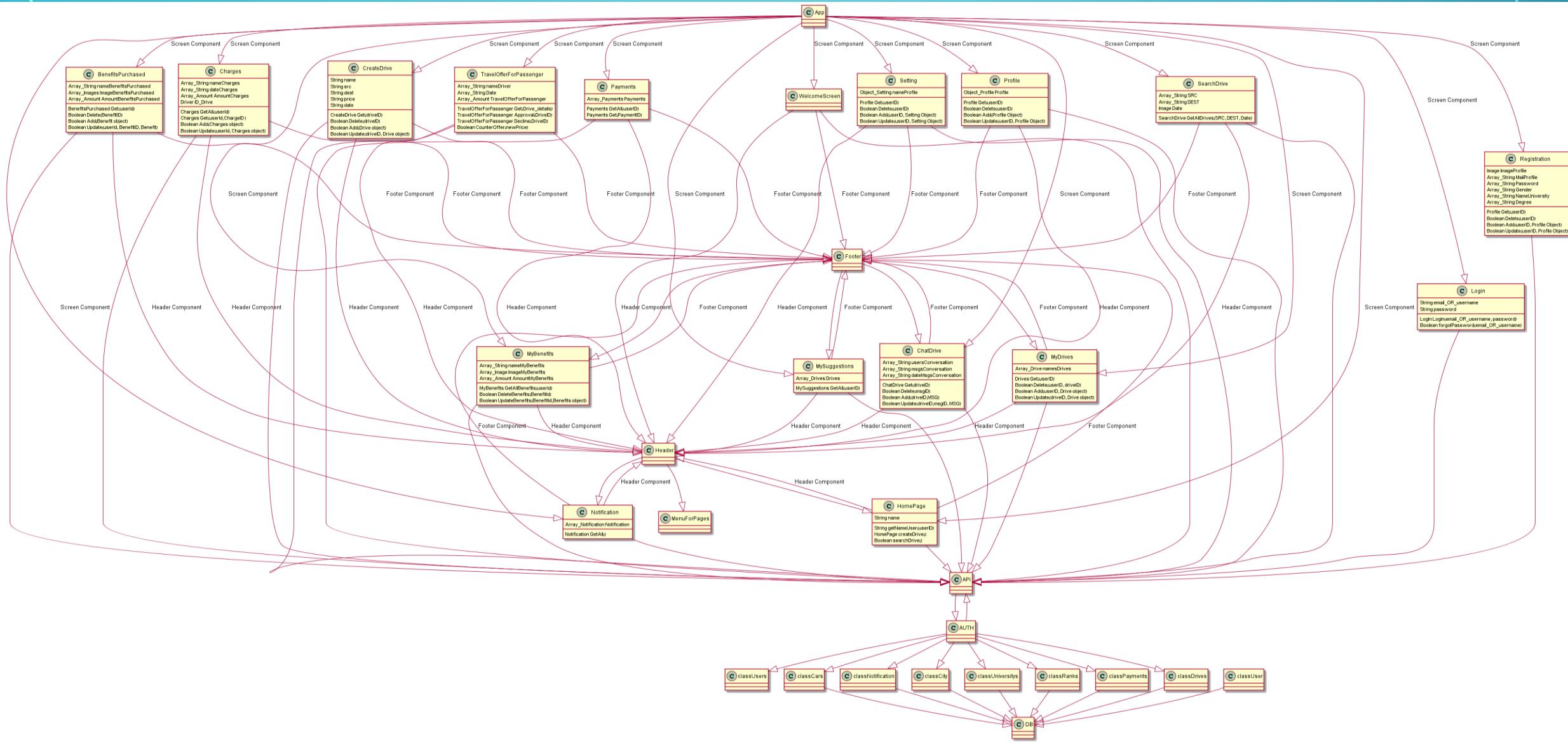
Users can choose with which platform they would like to pay. This step still needs to be implemented to ensure the users security and better KYC guidelines.



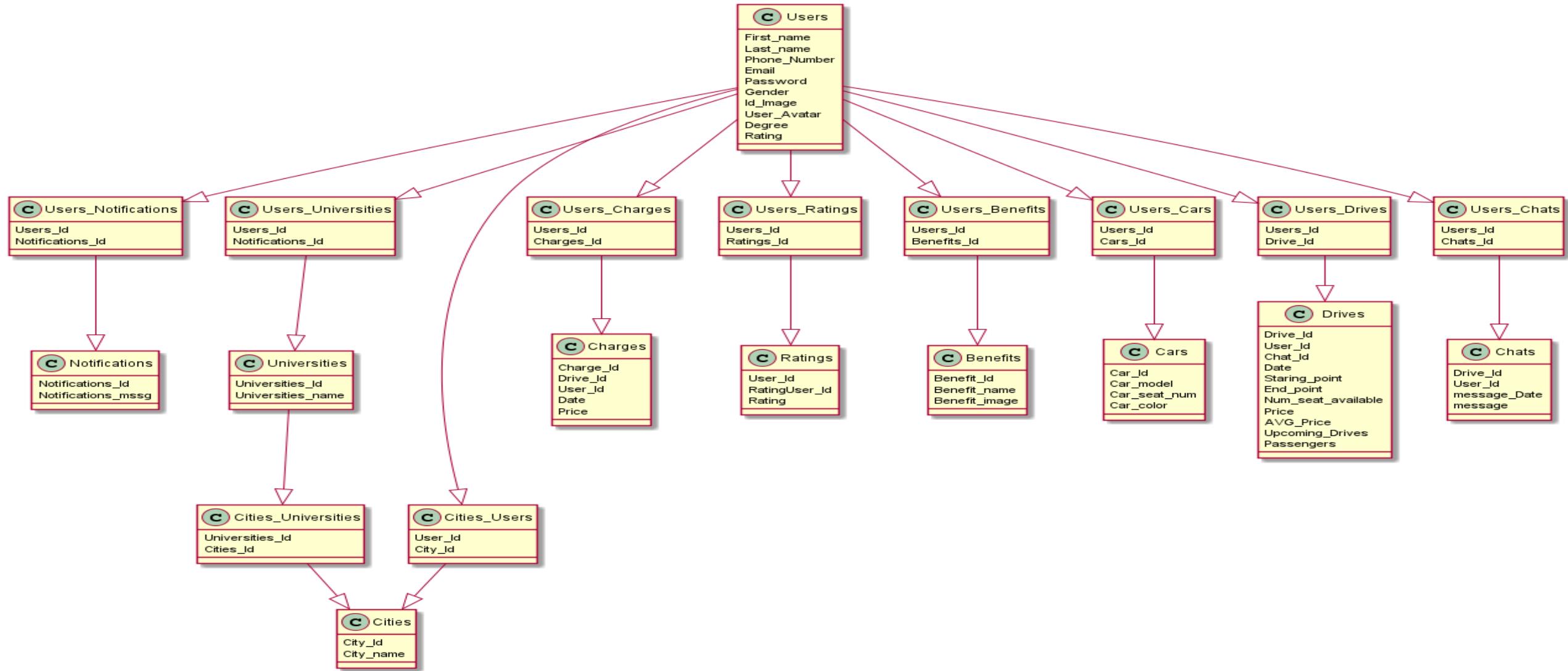
ACTIVITY DIAGRAM



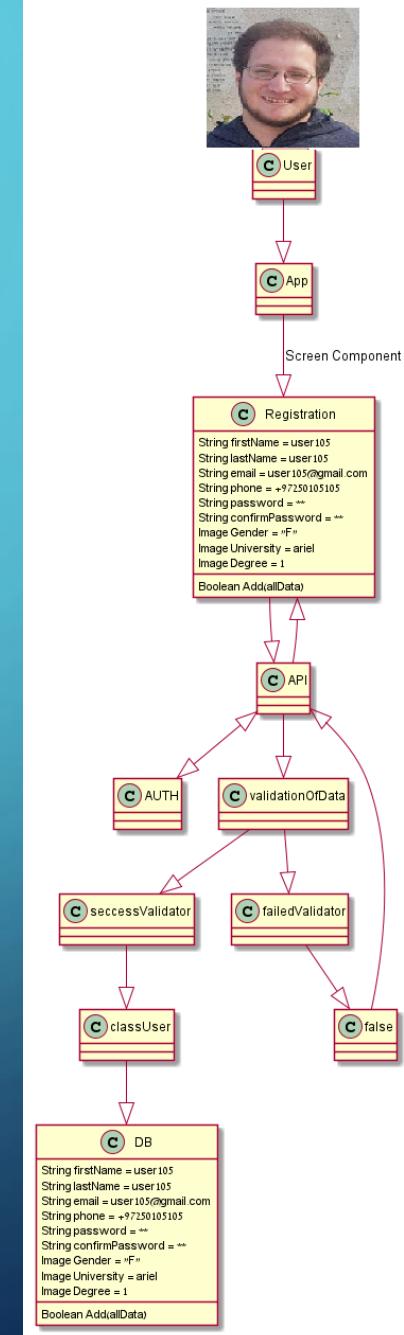
CLASS DIAGRAM



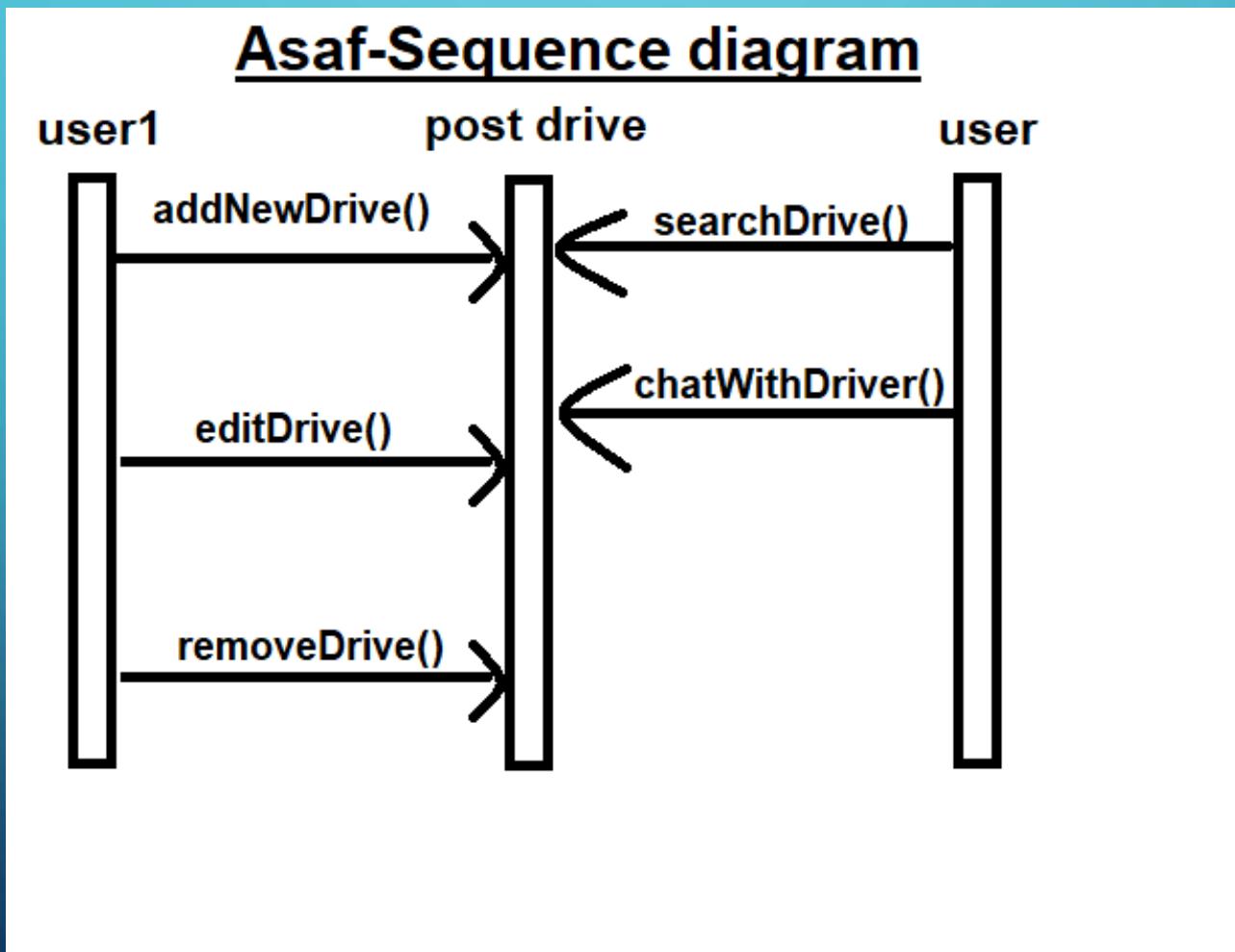
ERD DIAGRAM



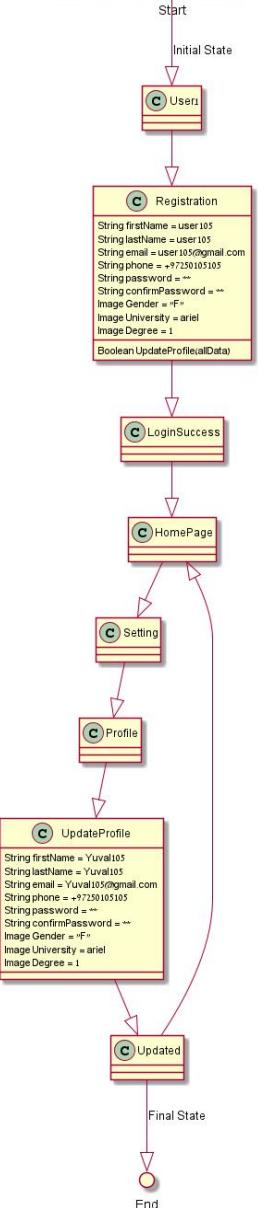
OBJECT DIAGRAM



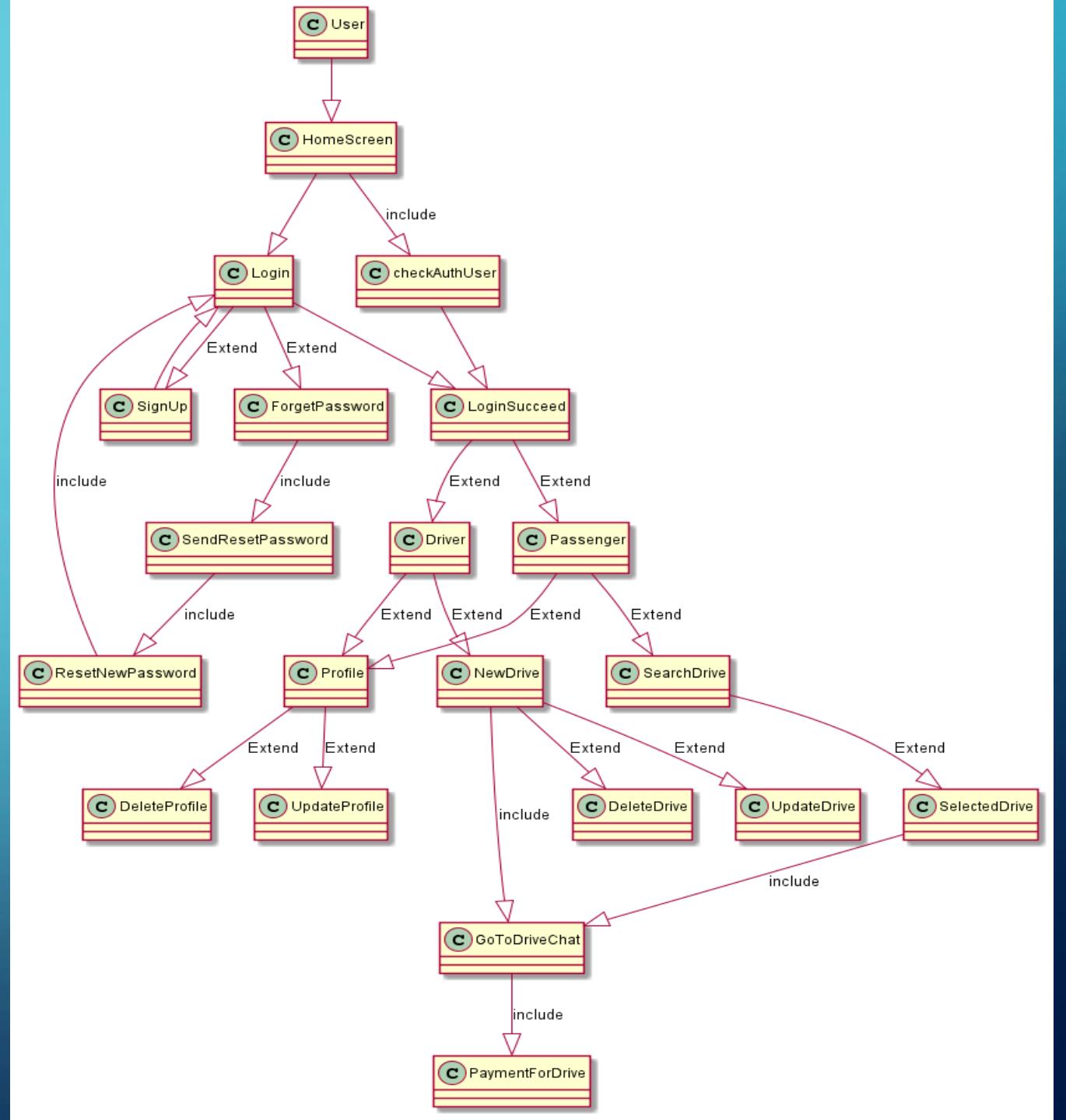
SEQUENCE DIAGRAM



STATE MACHINE DIAGRAM



USE CASE DIAGRAM



MAIN OBJECTS AND METHODS IN OUR APP

The app root components in our project contain useState which allows us to create an object and a function to set the value of the state in which the object is.

Userdata:

This state allows us to manipulate and access the user's data.

```
const [userData, setUserData] = useState({  
  token: "",  
  id: "",  
  firstName: "",  
  lastName: "",  
  email: "",  
  phoneNumber: "",  
  gender: "",  
  imageUniversity: "",  
  imageId: "",  
  university: "",  
  degree: "",  
  rating: "",  
  numRating: "",  
  numDrives: "",  
  modelCar: "",  
  colorCar: "",  
  dateOfBirth: "",  
});
```

RTL/LTR:

The app configures the layout according to left hand side or right hand side.

```
try {  
  ReactNative.I18nManager.allowRTL(false);  
} catch (e) {  
  console.log(e);  
}
```

Context provider:

This hook allows us to use "global variables" in order to share their value across all child components without using them as props in the child component such as users data.

```
import React, { useState, useLayoutEffect, createContext } from 'react';
```

isSignedIn:

this state confirms that the user logged in, in case he didn't he user will not be able to access and navigate to the rest of the screens. This allows for better data security.

Gravatar:

Gravatar library allows us to access the user's profile image using his email address.

```
import { Gravatar, GravatarApi } from 'react-native-gravatar';

<Gravatar options={{
    email: userData.email,
    parameters: { "size": "200", "d": "mm" },
    secure: true
}}
style={{
    height: "100%",
    width: "100%",
    padding: 5
}} />
```

```
<Context.Provider value={{
    userData, setUserData,
    loadingPopup, setLoadingPopup,
    isSignedIn, setIsSignedIn,
    cities, universities, checkCityForUniversity,
    searchDrive, setSearchDrive,
    suggestionsDriver, setSuggestionsDriver,
    detailsDrive, seDetailsDrive
}}>
```

```
<NavigationContainer>
  <Stack.Navigator>
    {!isSignedIn ? (
      <>
        <Stack.Screen name="Registration" component={Registration} options={{ headerShown: false }} />
        <Stack.Screen name="Login" component={Login} options={{ headerShown: false }} />
      </> :
      <>
        <Stack.Screen name="HomePage" component={HomePage} options={{ headerShown: false }} />
        <Stack.Screen name="ChatDrive" component={ChatDrive} options={{ headerShown: false }} />
        <Stack.Screen name="MySuggestionsDrive" component={MySuggestionsDrive} options={{ headerShown: false }} />
        <Stack.Screen name="CreateDrive" component={CreateDrive} options={{ headerShown: false }} />
        <Stack.Screen name="SearchDrive" component={SearchDrive} options={{ headerShown: false }} />
        <Stack.Screen name="Notification" component={Notification} options={{ headerShown: false }} />
        <Stack.Screen name="Rating" component={Rating} options={{ headerShown: false }} />
        <Stack.Screen name="Profile" component={Profile} options={{ headerShown: false }} />
        <Stack.Screen name="MySuggestionsPassenger" component={MySuggestionsPassenger} options={{ headerShown: false }} />
        <Stack.Screen name="Menu" component={Menu} options={{ headerShown: false }} />
        <Stack.Screen name="Payments" component={Payments} options={{ headerShown: false }} />
        <Stack.Screen name="MyChats" component={MyChats} options={{ headerShown: false }} />
        <Stack.Screen name="MyBenefits" component={MyBenefits} options={{ headerShown: false }} />
        <Stack.Screen name="WelcomeScreen" component={WelcomeScreen} options={{ headerShown: false }} />
        <Stack.Screen name="BenefitsPurchased" component={BenefitsPurchased} options={{ headerShown: false }} />
        <Stack.Screen name="Charges" component={Charges} options={{ headerShown: false }} />
        <Stack.Screen name="MyDrives" component={MyDrives} options={{ headerShown: false }} />
        <Stack.Screen name="MySuggestions" component={MySuggestions} options={{ headerShown: false }} />
        <Stack.Screen name="Setting" component={Setting} options={{ headerShown: false }} />
        <Stack.Screen name="Login" component={Login} options={{ headerShown: false }} />
        <Stack.Screen name="Registration" component={Registration} options={{ headerShown: false }} />
      </>
    )
  </Stack.Navigator>
</NavigationContainer>
```

Hooks:

These hooks help us adding more functional operations and solving multiple complex problems, which involve data and app configuration. For additional information regarding hooks in react –

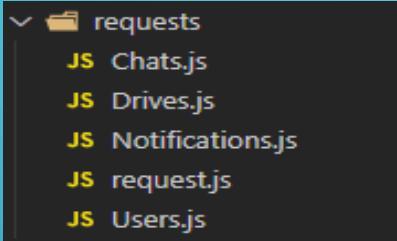
<https://reactjs.org/docs/hooks-reference.html>

```
import React, { useState, useLayoutEffect, useRef, useEffect, createContext, useReducer, useContext } from 'react';
```

For example:

```
const { userData, setUserData, loadingPopup, setLoadingPopup, isSignedIn, setIsSignedIn } = useContext(Context)
```

These const values in the homepage are retrieved from the main app component. As explained in the context provider above.



This is the core of our app data retrieval and update from our AWS server developed using NodeJS

request.js :

this file is responsible for all the paths of our requests from the server.

```
request:  
  registrations:  
    // @route POST api/addUser  
    // @desc add new user  
    // @body {username: "testname", "password": "avater", "gender": "male", "university": "university", "degree": "degree"}  
    // @header none  
  getUsers:  
    // @route GET api/getAllUsers  
    // @desc get user's details  
    // @header none  
  getOtherUsers:  
    // @route GET api/getOtherUsers  
    // @desc get other user's details  
    // @body {email: "test@123.com"}  
    // @header none  
  getAll:  
    // @route GET api/getAll  
    // @desc gets array of all users  
    // @body none  
    // @header none  
  update:  
    // @route PUT api/updateUser  
    // @desc update user's details  
    // @body {fields to be update}  
    // @header token  
  delete:  
    // @route DELETE api/deleteUser  
    // @desc delete user  
    // @body none  
    // @header token  
  login:  
    // @route POST api/login  
    // @desc log in existing user  
    // @body {email: "test@123.com", "password": "avater"}  
    // @header none  
  
Drives:  
  add: "https://tspqr32w.execute-api.us-east-1.amazonaws.com/production/api/addDrive",  
  // @desc add new drive to user  
  // @body {name: "name", "type": "public", "private": "private"}  
  delete:  
    // @route DELETE api/deleteDrive  
    // @desc delete specific drive  
    // @body "driveId"  
    // @header none  
  
getOfferFares:  
  // @route POST api/getOfferFares  
  // @desc get offers to all drives that match to the request (as passenger)  
  // @body {time: "time", "lat": "lat", "lon": "lon"}  
  // @header token  
  getPassengerOfferFares:  
  // @route GET api/getPassengerOfferFares  
  // @desc get all my offers as a passenger
```

Users.js :

this file is responsible for the requests concerning the user information.

Same goes to the rest of the files:

Notifications.js

Drives.js

Chats.js

```
import requests from './request';  
import axios from 'axios';  
import token from '../logInLogOut';  
  
const users = {  
  registration: async (userData) => {  
    try {  
      const config = {  
        headers: {  
          'Content-Type': 'application/json'  
        }  
      };  
      console.log("headers = " + config.headers);  
      console.log("user = " + JSON.stringify(userData));  
      const body = JSON.stringify(userData);  
      const res = await axios.post(requests.users.registration, body, config);  
      console.log("res = " + JSON.stringify(res.data));  
      return res;  
    } catch (err) {  
      console.error("err = " + err.response.data);  
      console.error("request failed");  
      return null;  
    }  
  },  
  getUsers: async (token) => {  
    try {  
      const config = {  
        headers: {  
          'Content-Type': 'application/json'  
        },  
        'x-auth-token': token  
      };  
      console.log("config.headers = " + config.headers);  
      const res = await axios.get(requests.users.getUsers, config);  
      return res;  
    } catch (err) {  
      console.error("err = " + err.message);  
      console.error("request failed");  
      return null;  
    }  
  },  
  getOtherUsers: async (userData, token) => {  
    try {  
      const config = {  
        headers: {  
          'Content-Type': 'application/json'  
        },  
        'x-auth-token': token  
      };  
      console.log("config.headers = " + config.headers);  
      const body = JSON.stringify(userData);  
      const res = await axios.post(requests.users.getOtherUsers, body, config);  
      console.log("res = " + JSON.stringify(res.data));  
      return res;  
    } catch (err) {  
      console.error("err = " + err.response.data);  
      console.error("request failed");  
    }  
  },  
  getAll: async (token) => {  
    try {  
      const config = {  
        headers: {  
          'Content-Type': 'application/json'  
        },  
        'x-auth-token': token  
      };  
      console.log("config.headers = " + config.headers)
```

```
import requests from "./request";
import axios from "axios";
import token from './.../logicApp';
```

Axios is the library that allows us to send requests (GET,POST,PUT,DELETE).

Additionally the library includes methods to customize the header and body of the requests.

logicApp.js:

this file manages the functions which corresponds with token related to the user.

```
import AsyncStorage from '@react-native-async-storage/async-storage';
```

The token is used with async-storage library as a substitute to regular sessions.

For all of our requests from the sever except on registration and Login, we send the server the request with the token inside the Config header, which ensures that the request will be executed if and only if the user exist and logged in to the App.

```
import AsyncStorage from '@react-native-async-storage/async-storage';

const token = {
  addTokenToStorage: async function (token) {
    try {
      await AsyncStorage.setItem('@token', token)
      console.log("token is saved")
      console.log("token = ", token)
    } catch (e) {
      console.log("token is not save")
      console.log("error", e)
    }
  },
  getTokenFromStorage: async function () {
    try {
      const token = await AsyncStorage.getItem('@token')
      if (token !== null) {
        return token
      } else {
        return ""
      }
    } catch (e) {
      console.log("empty val")
      console.log("error", e)
      return ""
    }
  },
  removeTokenFromStorage: async function () {
    try {
      await AsyncStorage.removeItem('@token')
      console.log('removed')
    } catch (e) {
      console.log('error when remove token')
      console.log('error', e)
    }
  }
}

export default token;
```

```
import * as ImagePicker from 'expo-image-picker';
```

Image Picker:

Once the component (screen) is accessed the App asks for the user's permission to access his photo gallery and allows us to save his images in our database as a path.

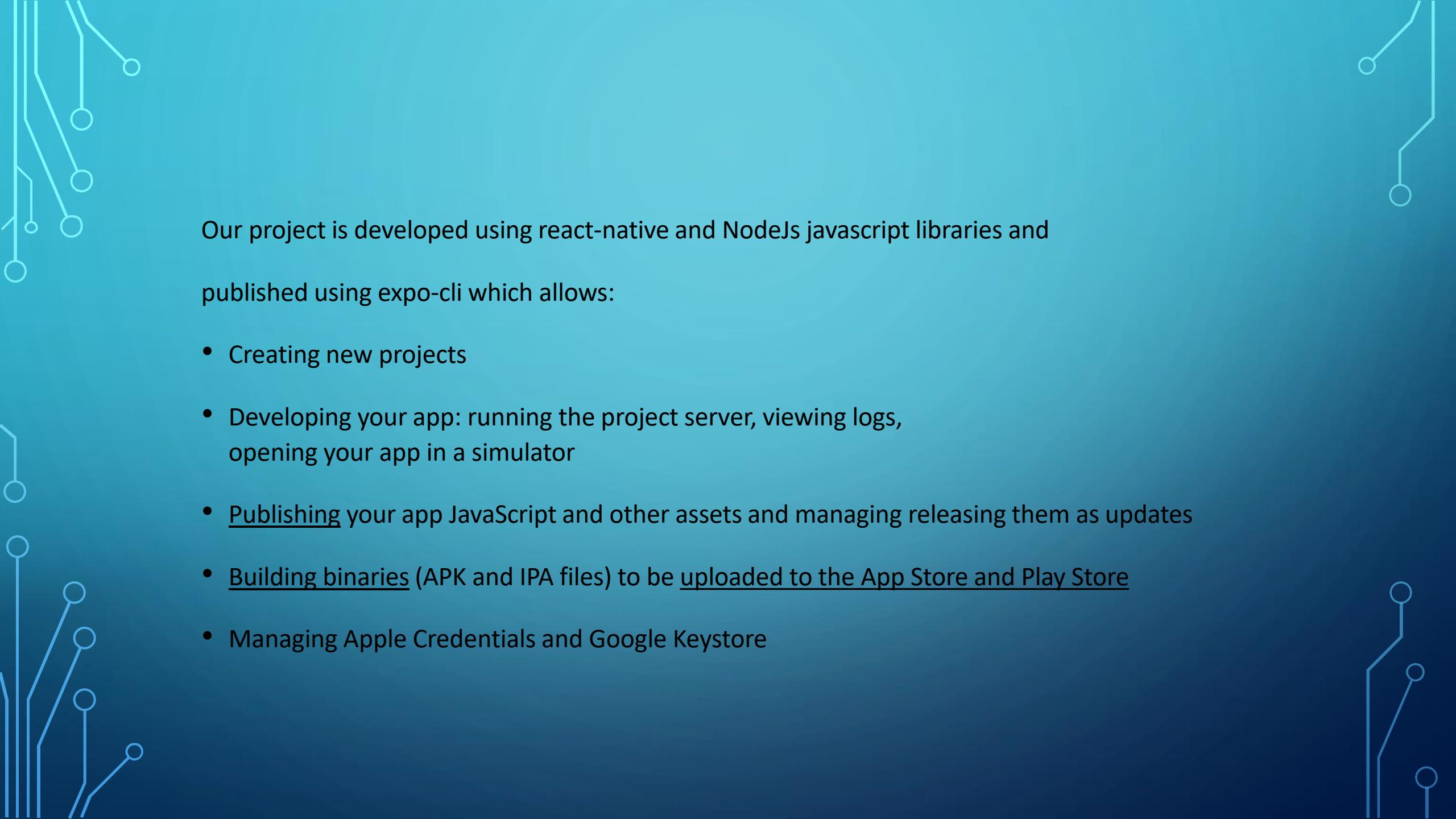
```
useEffect(async () => {
  if (Platform.OS !== 'web') {
    const { status } = await ImagePicker.requestMediaLibraryPermissionsAsync();
    if (status !== 'granted') {
      alert('Sorry, we need camera roll permissions to make this work!');
    }
  }
}, []);
```



```
const pickImageUniversity = async () => {
  let result = await ImagePicker.launchImageLibraryAsync({
    mediaTypes: ImagePicker.MediaTypeOptions.All,
    allowsEditing: true,
    aspect: [4, 3],
    quality: 1,
  });
  if (!result.cancelled) {
    console.log("result.uri = " + result.uri)
    setImageUniversity(result.uri);
    console.log("data.imageUniversity = " + data.imageUniversity)

  }
  console.log("data.imageUniversity = " + data.imageUniversity)
};

const pickImageId = async () => {
  let result = await ImagePicker.launchImageLibraryAsync({
    mediaTypes: ImagePicker.MediaTypeOptions.All,
    allowsEditing: true,
    aspect: [4, 3],
    quality: 1,
  });
  if (!result.cancelled) {
    setImageId(result.uri);
  }
  console.log("data.imageId = " + data.imageId)
};
```



Our project is developed using react-native and NodeJs javascript libraries and published using expo-cli which allows:

- Creating new projects
- Developing your app: running the project server, viewing logs, opening your app in a simulator
- Publishing your app JavaScript and other assets and managing releasing them as updates
- Building binaries (APK and IPA files) to be uploaded to the App Store and Play Store
- Managing Apple Credentials and Google Keystore



אפליקציה לרכב מונען "אסף"



- האפליקציה מאפשרת לסטודנטים למצוא טרמפים לאוניברסיטה וממנה, משרות בין נהגים לבקשת נסיעות ומאפשרת לנגן לקבוע את מחיר הנסעה ולנוסע להצעה הנדרשת (באמצעות בידים)
- האפליקציה מקלת על מציאת טרמפים לאוניברסיטה בקלות ויעילות ובכך מצמצמת את כמות המכוניות על הכבישים, מזערת את צריכה הדלק, פקקים, זיהום אוויר ושומרת על איכות הסביבה. כמו כן, מייצרת קשר חברתי בין הסטודנטים

מטרות עסקיות



מודל עסק

- גזירת עלות של 5% על כל תשלום באפליקציה
- כל משתמש יוכל לצבור נקודות (נהגים צוברים נקודות ע"י ביצוע טרמפים וקבלת דירוג גבוה, נוסעים צוברים נקודות ע"י דירוג הנהגים). את הנקודות ניתן למשוך ככף בתחום האפליקציה, או בבתי עסק קרובים לאוניברסיטאות באמצעות דילימ שנקבעו אל מול בית העסק ע"י היוצרים בתמורה להשקעה

טוווח ארוך: לאחר חצי שנה פעילות, האפליקציה תהיה מיעדת לכל האוניברסיטאות בארץ. בארץ ישם 312,660 סטודנטים מתוכם 236,850 סטודנטים לתואר ראשון. ממרץ 2022, האפליקציה תהיה בהיקף של 80,000 הורדות ו-30,000 משתמשים פעילים

טוווח קצר: בחצי השנה הראשונה האפליקציה תהיה מיועדת לאוניברסיטת אריאל בלבד המכילה 16,000 סטודנטים. עד אפריל 2021, האפליקציה תהיה בהיקף של 5,000 הורדות ו-700 משתמשים פעילים. עד יוני 2021, האפליקציה תהיה בהיקף של 8,000 הורדות ו-1,500 משתמשים פעילים



קהל העל

קהל היעד הם סטודנטים, נשים וגברים, בגילאי 18-35 מכלל הארץ

פלח אוכלוסייה נהגים

נשים וגברים בגילאי 27-35, נשואים, בעלי משרה מלאה ורמת הכנסתה בינונית, גרים במרכז, סטודנטים לתואר ראשון/שני, בעלי רכב.

תחומי עניין: טכנולוגיה, בישול, מוסיקה, צדרות, טיולים, אקטואליה, איקות הסביבה, קולנוע, כלכלה, ספרים.



אפיון פרטונה

נתע זורה



מאפיינים דמוגרפיים

גיל: 25

מין: נקבה

marshah/עובדת: אחמד"שิต בקסטרו

סטודנט משפחתי: בזוגיות

מקום מגורים: מודיעין

ארכיטיפ: חברותית

מאפיינים אישיותיים

תכונות אופי מרכזיות



BIO

נתע גרה ביחידת דיר עם בן זוגה, קנתה את האוטו
שלה במהלך השירות שלה בתור קצינת חינוך, ומazel לא
mpsikha להיות הנציגת התורנית של החבורה. היא
לומדת פסיכולוגיה באוניברסיטה אריאל 3 פעמים
שבוע עד אחר הצהרים. עובדת בקינון במודיעין מאז
השחרור במשרה מלאה על מנת לחסוך בסף לחותנה.
נתע אוהבת לאכול סושי עם חברות, ולראות סרט עם
קובי פעם בשבוע. נתע היא לא בן בוקר, ומה
שמעיר אותה זה הפליליסט שלה בזמן נסעה. נתע
שונאת ריח של פירות הדדר.

**אהבת לנסוע עם אנשים
חברותיים**

אוריננטציה טכנולוגית

שימוש באינטרנט ו-Web



שימוש בממשק תוכנה Software



שימוש באפליקציות Mobile Apps



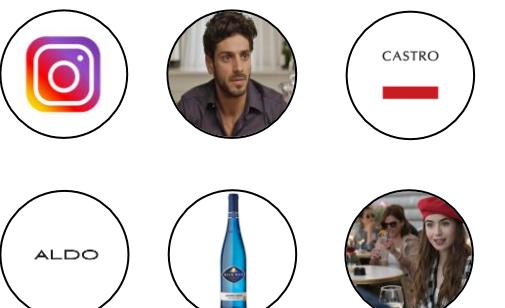
שימוש ברשותות חברותיות Social Networks



תרחישי שימוש אפשריים

- ❑ בסוף יום העבודה, בשבייל למצוא נסעים למחזרת
- ❑ על מנת למשך קופון הנחה בגין פניהה
- ❑ לאחר נסעה על מנת להשאר ביקורת על הנושא המuczben
- ❑ בתגובה המבחנים, ערבות לפני המבחן, על מנת למצוא טרמפיקטים

מותגים ובעלי השפעה



מושיבציות

- ❑ מציאת טרמפיקטים בלחיצת כפתור
- ❑ חישכון בזמן בחיפוש נסעים
- ❑ חישכון בכسطף
- ❑ שילטה על מחיר הטרםפ
- ❑ קלות חיפוש

מטרות

- ❑ מציאת נסעים קרוב לבית
- ❑ מציאת נסעים בכמה דקות
- ❑ השתתפות בעלות הדלק עם כמה שיוצרת אנשים

תסכולים

- ❑ אפליקציה לאינטואטיבית
- ❑ מבחר קטן של נסעים
- ❑ הנוסעים לא זמינים ביצ'אט באפליקציה
- ❑ נסעים לא נעימים ומטרידים



זיהוי ולמידה ממתחרים בשוק

מתחרים מרכזים



Whatsapp



Waze carpool

- 77% מהסטודנטים [...] לא יכולים ליותר על הרכב הפרטி, 52% [...] מגיעים ברכב הפרטி, 23% מגיעים בתחבורה הציבורית, 10% מגיעים בטרמפים (השכמה ראשון לציון, 2018)

- "מבדקה מדגמית של כ-20 אוניברסיטאות ומכילות מובילות ומוכרות, עולה כי כמעט בכל הבדיקות נרשמו זמני הגעה ארוכים, חלקיים אף כפולים, של התחבורה הציבורית לעומת כלי הרכב הפרטי" (TheMarker, 2016)

המתחרים נותנים מענה לטרמפים אך בצורה מסורבלת ובذر"כ גם יקרה יותר, לנוסף אין אמרה לגבי המחיר, והמתחרים כלל לא מייעדים לסטודנטים בלבד



תכונות הליבה של הממשק

- מיציאת טרמף לפי תאריך, מקום מוצא ומקוםיעד, כאשר הנהג יצא לדרכו
- פרסום טרמף לפי תאריך, מקום מוצא ומקוםיעד, כמוות מושבים ברכב ומחייבים
- הוספת הערות חריגות של הנהג על הנושא בתום הנסיעה
- שמירת נתוני הנסיעה הקבועים
- דירוג הנהג ע"י הנוסעים בתום הנסעה
- הצעה מחיר נגדית ע"י הנושא
- צפיה ביום נסיעות
- צ'אט בין הנהג לנוסעים לאחר אישור הטרמף
- ל תשלום על נסיעות
- הנסעה

פלואו נהג: המשמש כניסה למסך הרשמה/התחברות (1) > במסך הראשי (2) ניתן לבחור אם מדובר בנהג או נוסע ולהיכנס לפרופיל האישית {איקון תמונה} (3), בתרפיט התחתון אפשר לבחור בנסיבות שלך {איקון לוח שנה} (4), במסך הצלאים שלך {איקון צ'אט} (5), ההטבות שלך {איקון מתנה} (6), בתרפיט העליון ניתן להגיע למסך התראות שלך (7) > לאחר הבחירה בנהג או נוסע, הנהג מ מלא את פרטי הנסעה במסך הוספת נסעה (8) > לאחר פרסום הנסעה הנהג מקבל הצעות במסך התראות (7) מנוסעים פוטנציאליים, הוא יכול לאשר/לסרב > לאחר האישור, נפתח מסך אישור נסעה (9). ניתן להגיע מתרפיט המברגר למסך הבית (10), מסך דירוגים והערות (11), מסך תלמידים (12) ומסך הגדרות (13).

עג מסך ניהול

