Android Project: Service Novigrad App

SEG2105- Introduction to Software Engineering

Fall 2020

School of Electrical Engineering and Computer Science
University of Ottawa

Course Coordinator: Dr. Miguel A. Garzón

Group #31

Student Name and number: Jonah Cantello (300112348)
Student Name and number: Comtois Amelié (300061996)
Student Name and number: Nastexa Farah (300018467)
Student Name and number: Derek Hacault-Parodi (300122544)
Student Name and number: Ansh Patel (300102472)

Submission Date: December 6th, 2020

Introduction

Using Firebase, we have created an application that: allows employees (branches) to be made that can edit which services they provide, approve or deny service requests, change their phone number, address and open hours; allows customers to be made with the ability to submit service requests, search for branches and rate branches; and allows an admin to create and edit services along with deleting services, customers accounts and branches accounts.

The application uses a mix of Firebase's Real-Time Database system and the Authentication system to store data on users, services, and branches and to manipulate this data for customers and employees to communicate with one another. The administrator can make changes to the service database and the database of users at any time, in real-time, and other users will see those changes reflected immediately.

The application can be signed into from the first screen, bringing the user to their welcome screen, allowing the signed-in user to navigate between all of the offered features for their user type (e.g. Customer). Signing up will also bring the newly-created user to their welcome screen, offering an ergonomic and quick navigation cycle.

Challenges Faced/Lessons Learned

The first challenge we faced was learning Android Studio. It was new to us all, so we had to learn the best ways to coherently link the layouts to the code. This was the most difficult when a layout did not have an associated java file, such as the layouts used in lists. Learning how the java code interacted with the layouts and influenced elements of the interface was among the most significant challenges. We had to be incredibly attentive to which variables could be accessed from where when coding, and it influenced our organization practices throughout the whole project.

Another challenge that we encountered was in doing simultaneous work with GitHub as our group workspace. Editing specific files (especially instances in which multiple students worked on the same file) would generate conflicts in the GitHub repository, occasionally requiring extra time to resolve. As such, we needed to agree with clarity on the schedule of who would be working on which files and when these changes would be pushed to the repository.

A significant difficulty in designing an application with a graphical interface is that it can take different shapes for different devices and resolutions. Thus, it was necessary to proportion visual elements by screen size so that the application runs similarly on various devices. This provided the second hurdle of keeping our design strategies consistent throughout the project, not only between editing one's own work but also between each other's work.

This project also tasked us with learning to understand how Firebase operates as a database so as to make full use of its relevant capabilities in our project. This meant that we had to learn Firebase's unique methods, along with their unique restrictions and requirements. For

instance, one of the most apparent obstacles was that the admin account needed the ability to delete user accounts, but we had designed the Admin account to function using the Client SPK. Firebase requires the Admin SPK to delete other user accounts for security reasons, so we had to work around that limitation. Our program had to be flexible and mould around the Firebase code we were given to work with.

UML Code

```
class Account {
  email;
  passwords;
}
class UserAccount
{
  isA Account;
  firstName;
  lastName;
  username;
  role;
}

class Branch
{
  isA UserAccount;
  Service[] services;
  Map mapOfHours;
  public void addService(){}
  public void removeService(){}
  public void editWorkingHours(){}
  }

class Customer
{
  public void createServiceRequest(){}
  public void rateABranch(){}
  public void searchForBranch(){}
  isA UserAccount;
1 -- * ServiceRequests;
}
```

```
class Administrator
{
  isA Account;
  public void createService(){}
  public void deleteService(){}
  public void alterService(){}
  public void deleteCustomerAccount(){}
  public void deleteBranchAccount(){}

  1 -- * Service;
}

class Service
{
  String name;
  Double cost;
  String[] documents;
}

class ServiceRequests{
  Integer id;
  Integer branchID;
  Service associatedService;
  Map responses;
  Boolean approved;
  customerID;
  1 -- * Service;
}
```

Final UML Diagram

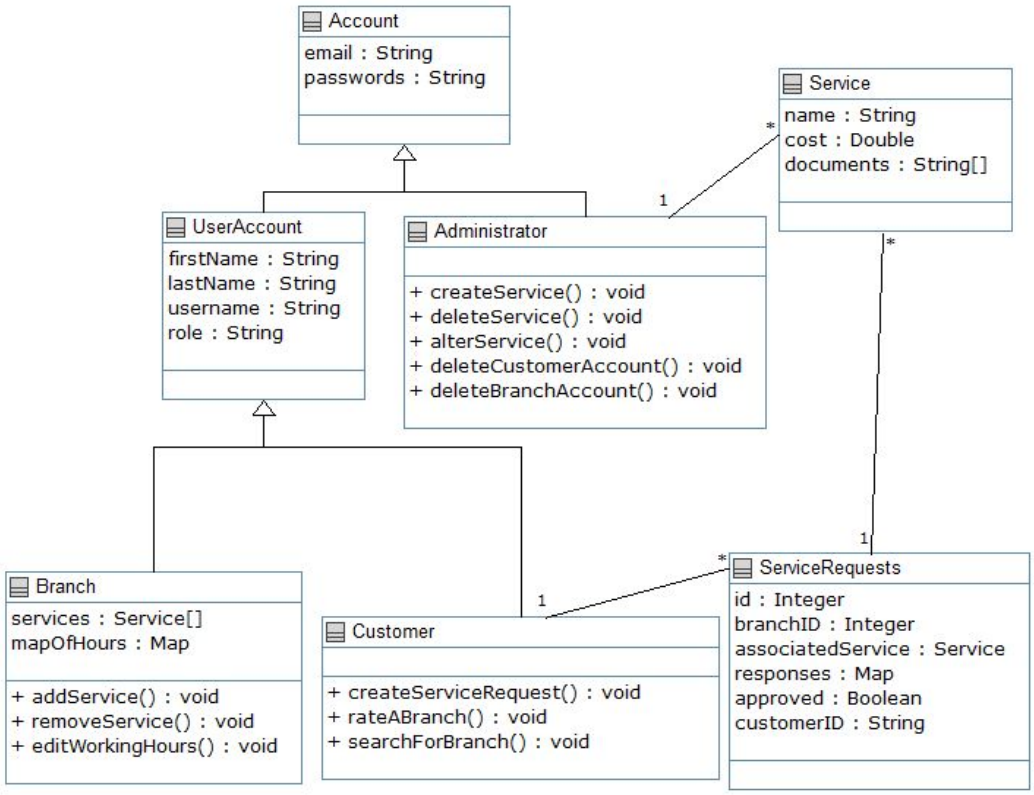


Table of contribution:

Deliverable 1:

UML 1	Collective progress
Account Creation	Nastexa; Ansh
Welcome Screen	Nastexa

Deliverable 2:

UML 2	Collective progress
Unit Test Cases	Derek
Admin--Add Services	Jonah; Derek

Admin--Edit/Delete Services	Jonah; Derek
Admin--Delete Accounts	Jonah; Derek

Deliverable 3:

UML 3	Collective progress
Branch--Profile Info	Amelie; Ansh
Branch--Service Selection	Jonah
Branch--Working Hours	Amelie
Branch--Approve/Reject	Amelie

Deliverable 4:

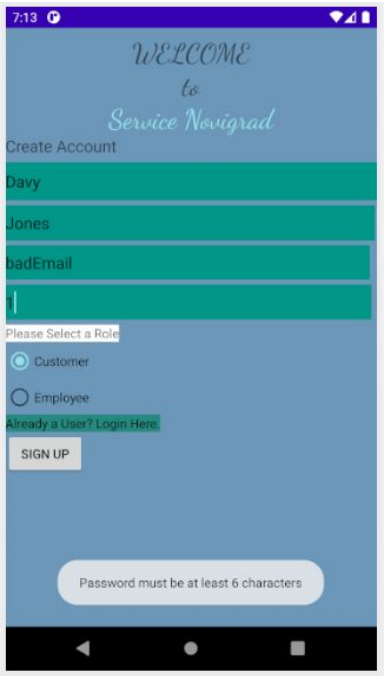
UML 4	Collective progress
Unit Test Cases	Derek
Customer--Search for Branch	Amelie
Customer--Submit Request	Amelie
Customer--Rate a Branch	Amelie
Final Report	Collective progress

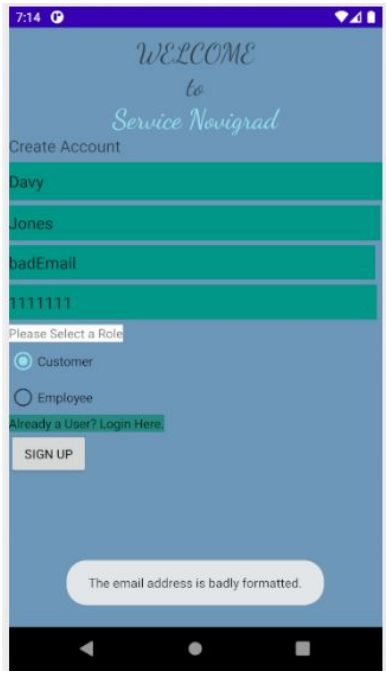
Screenshots:

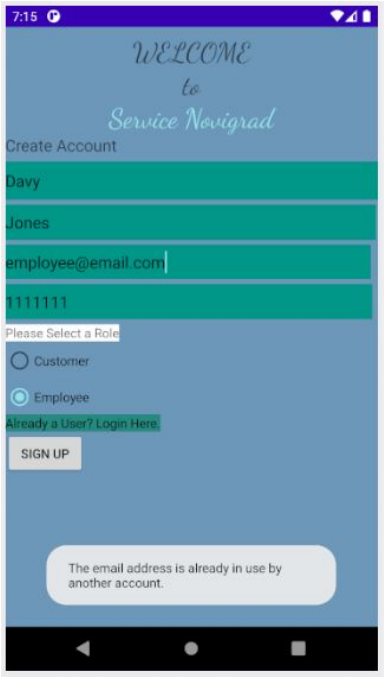
To go back on any screen (except back to the log in screen), press the back button on your phone/emulator.

Creating an account:

All fields are validated.

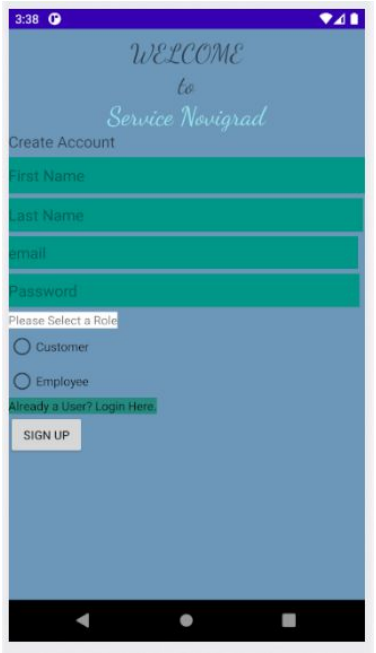


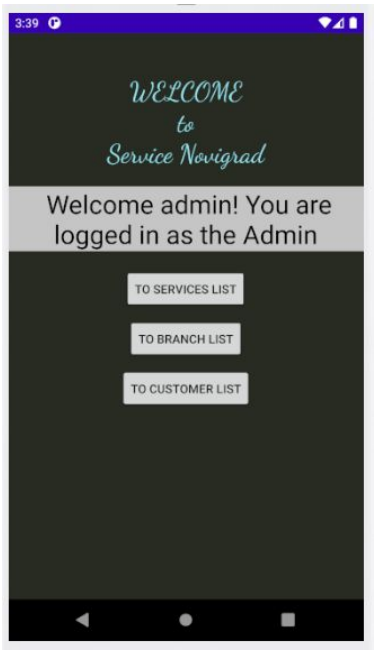




Admin Account:

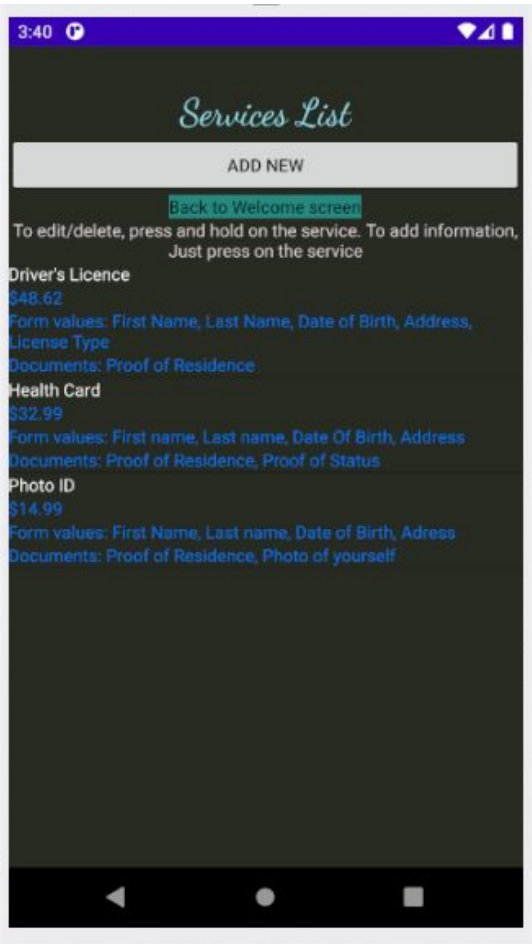
Clicking on “Already a User? Login here”, then logging in using the email “admin” and the password “admin” brings you to the admin menu.



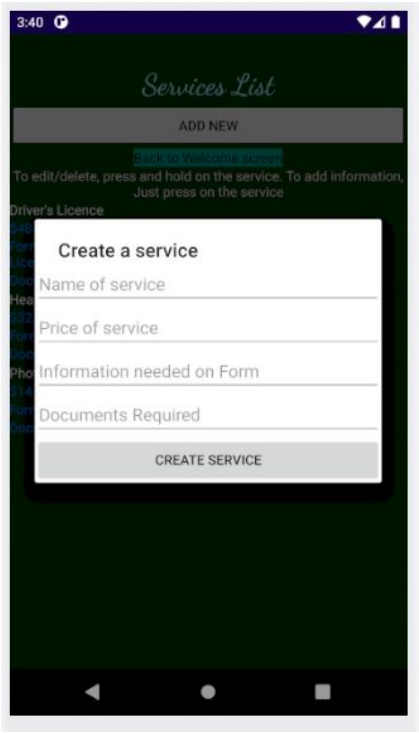


Admin Features

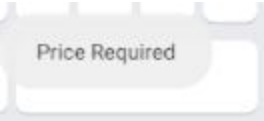
Pressing on “To service List” brings you to the next image:

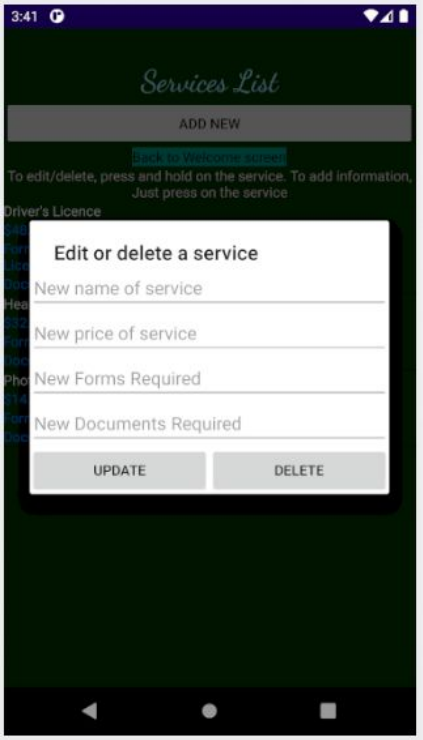


This screen shows a scrollable list of all services created. The first line is the service’s name, followed by the price, then the information needed on the form, then the documents required.



Pressing the "Add New" button creates a dialogue box where you can input a services name, price, information required, and documents required. You can not create a service without filling each of these in.

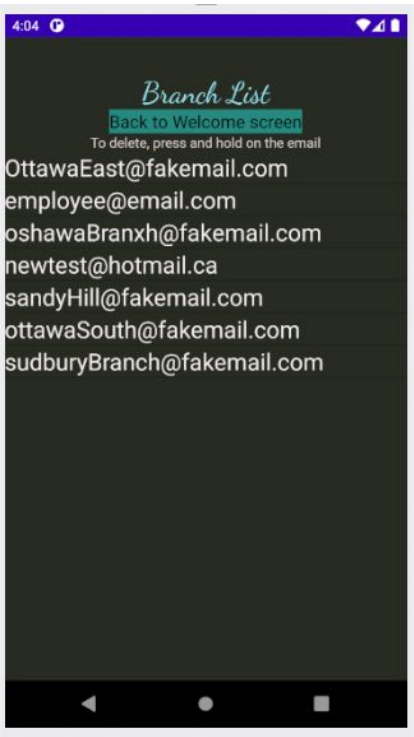


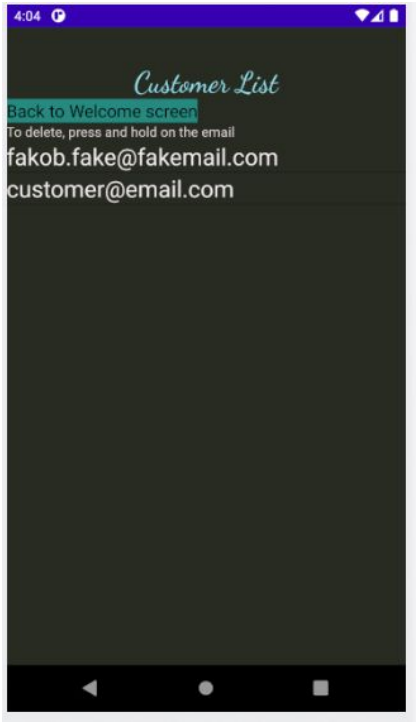


Long tapping on one of the services brings up a dialogue box, which allows you to change one of the values of the service.

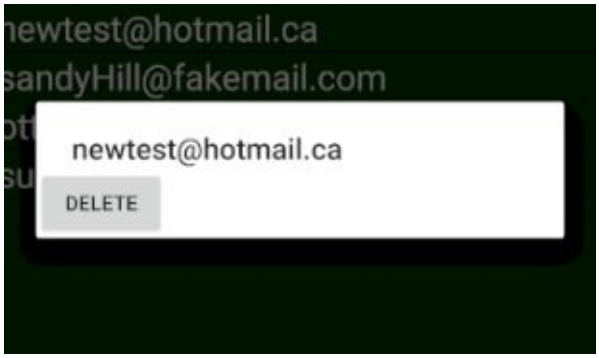
Pressing the “To Branch List” and “To Customer List”

Brings you to the branch list and customer list, respectively



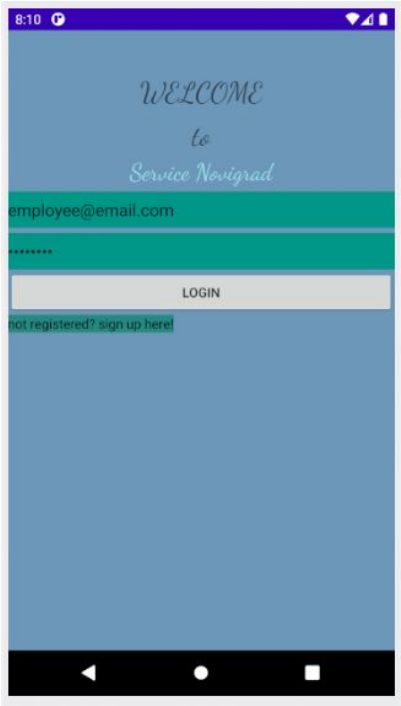


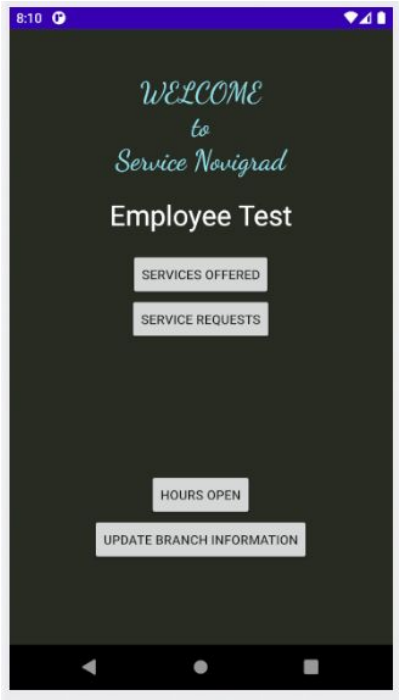
Long hold or press on one of the list items brings up a dialogue box that allows you to delete the branch or customer.



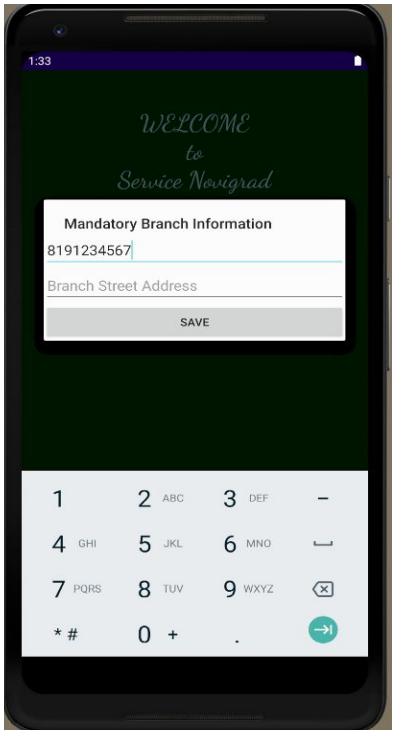
Branch/Employee Account:

Logging in using the credentials of a branch account brings you to the main page for branches.

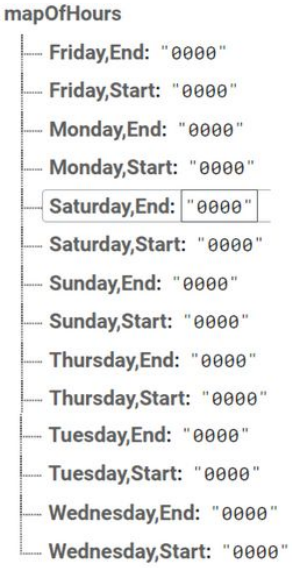




When first entering the branch main menu (e.g. you just created a branch account), a dialogue box appears, allowing you to enter a phone number and an address. This dialogue box can not be closed until both fields contain a valid input.

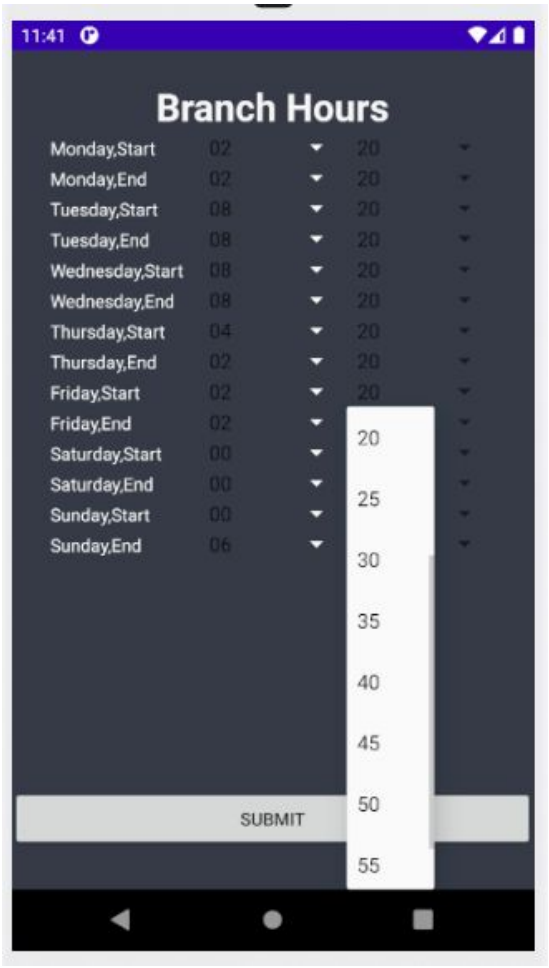


Upon successfully entering a valid phone number and address, the app assigns all start and closing hours to be 00:00. In order for the branches to be operational, we require the branch to go to “Hours Open” and edit the hours.



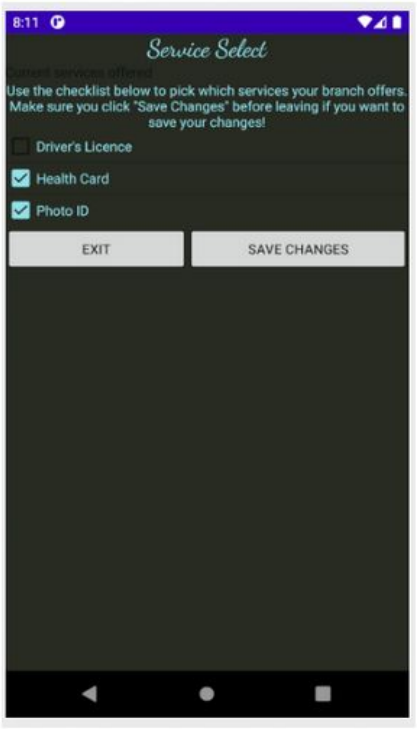
Pressing on “Hours Open”

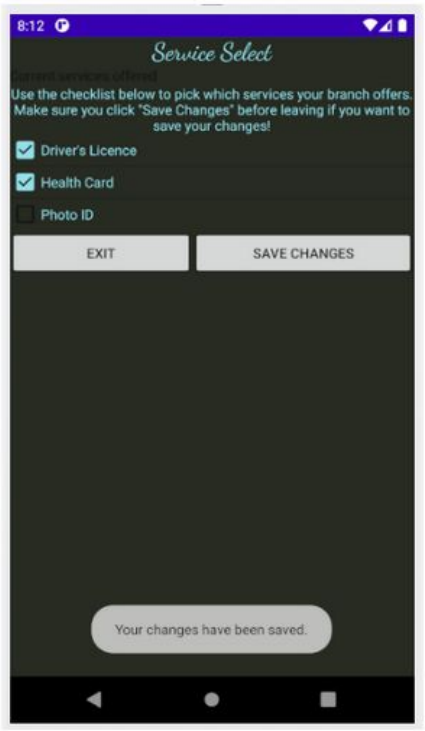
Brings you to a screen containing all the start hours and minutes. Pressing on an hour or minute brings a list of hours and minutes that can be selected. The save button saves the selection to the map of hours and brings you back to the branch menu.

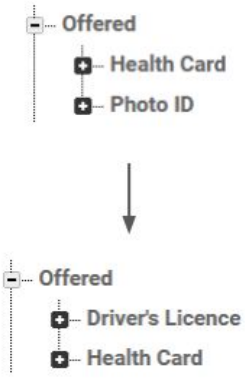


Pressing on “Services Offered”

Brings you to a screen containing a list of all services. Upon pressing “Save Changes,” a link to all selected services are stored in firebase under the branch that is currently logged in.

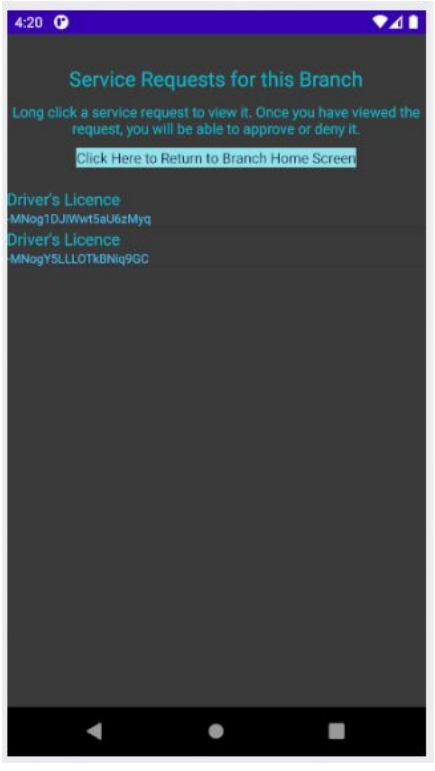




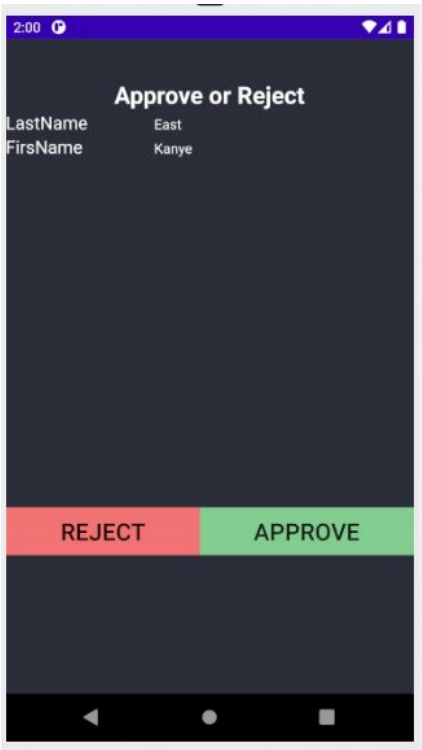


Pressing on “Service Requests”

Brings you to a screen containing all the service requests for that specific branch.

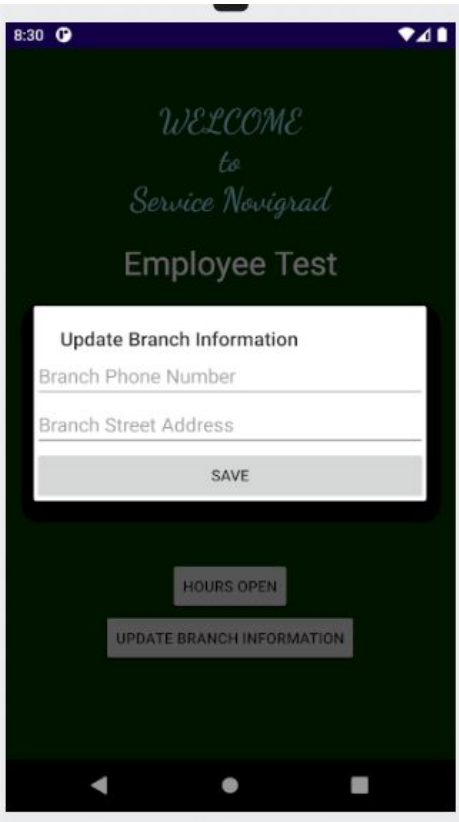


Long holding on one of the list items lets you see the information and documents added to the service requests, along with the ability to reject or approve the service request.



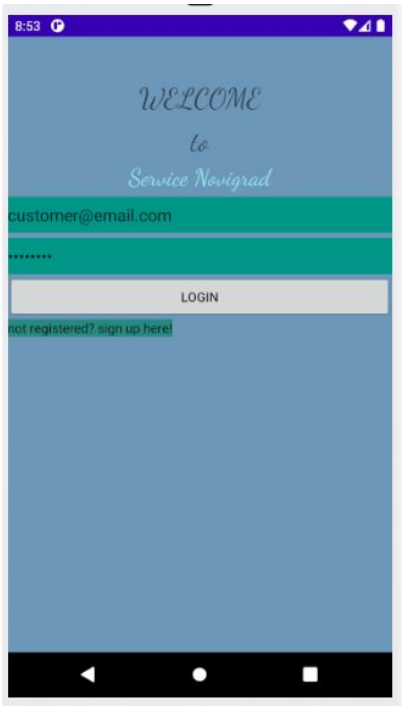
Pressing on the “Update Branch Information”

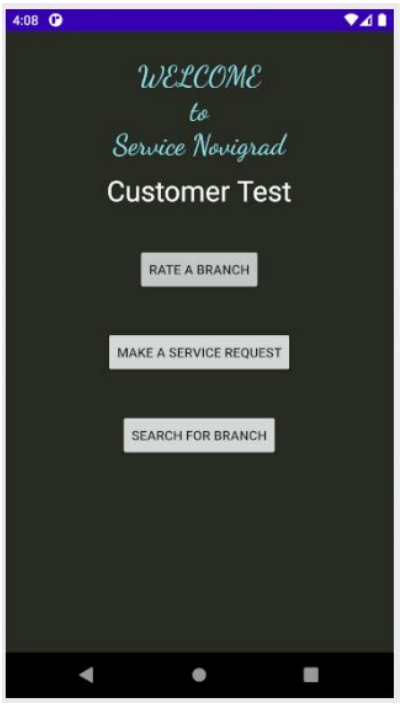
Brings up a dialogue box that allows you to change the phone number and address of your branch.



Customer Account:

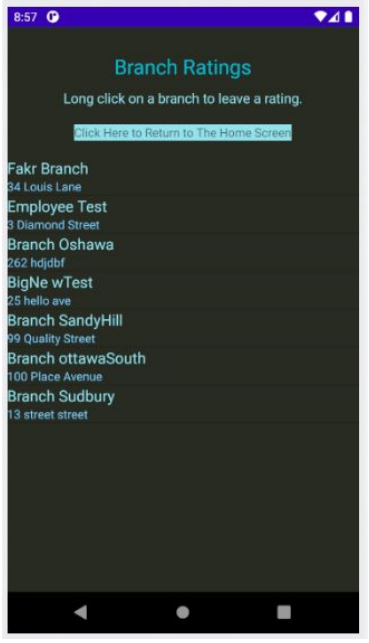
Logging in using the credentials of a customer account brings you to the main page for customers.

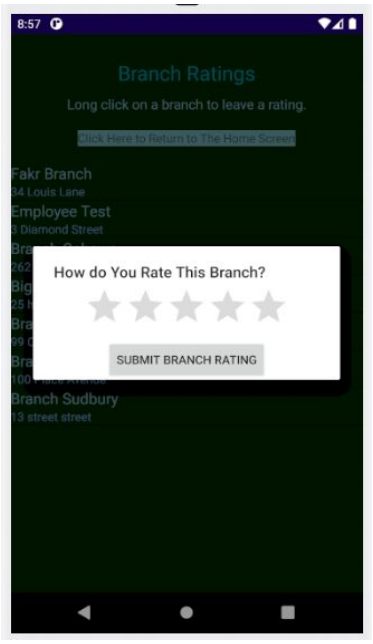




Pressing on the “Rate A Branch”

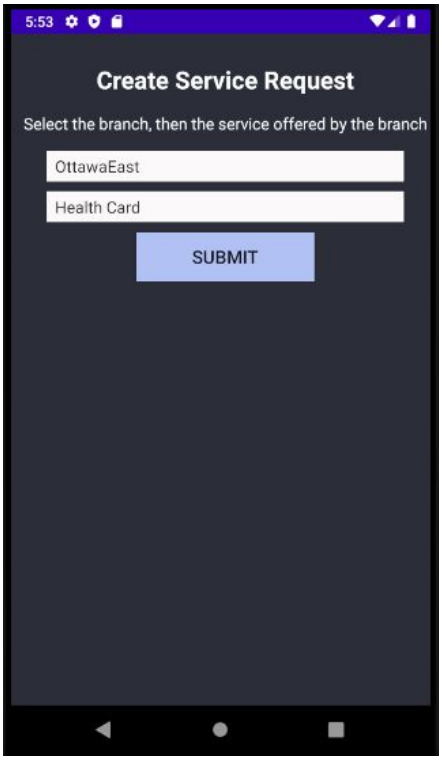
Brings you to a screen that displays all branches. **Long tapping** on one of the branches lets you submit a rating to the branch.





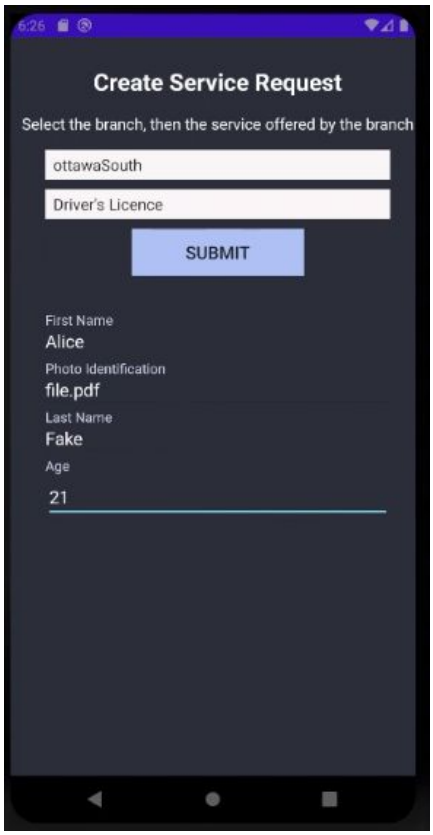
Pressing “Make A Service Request”

Brings you to a screen to make a service request.



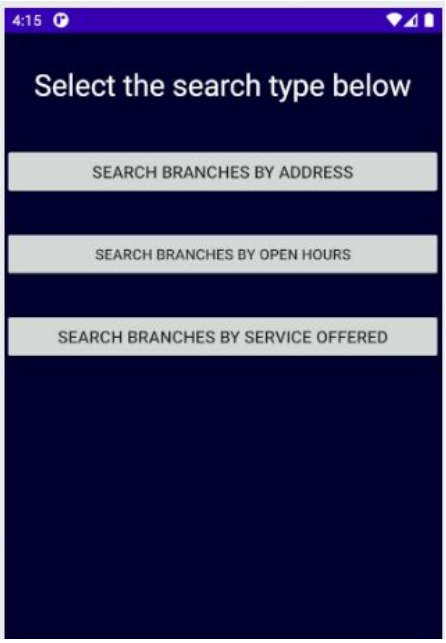
The top spinner lets you pick which branch you want the service from, and the bottom spinner lets you pick what service you want out of the services offered by the branch.

Once you fill in the required fields, tap the “Submit” button to submit the request.

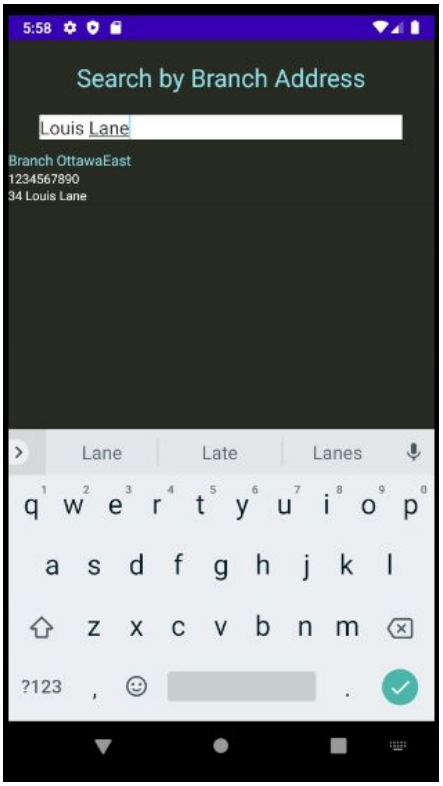


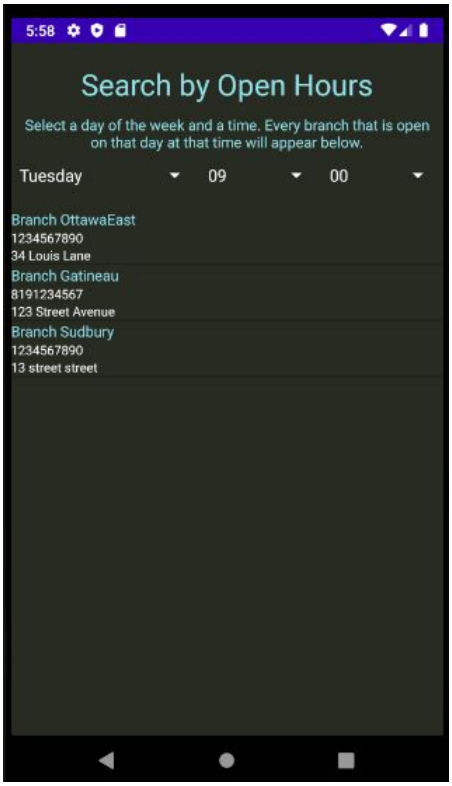
Pressing “Search For Branch”

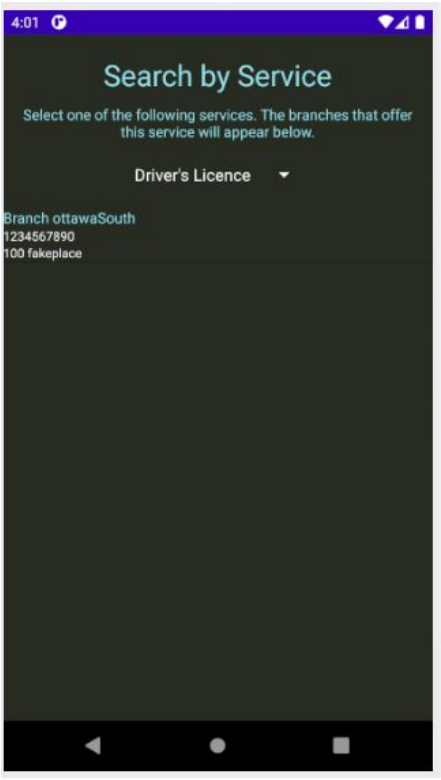
Lets you choose how you want to search for a branch.



Selecting one of the three buttons will allow you to type in your search parameter.







Long pressing any of the branches on a list will bring you to the same screen that pressing the “Make A Service Request” button on the customer main menu would.