University of Ottawa | Université d’Ottawa

SEG2105 | Fall 2020

SEG2105 – Introduction to Software Engineering

Course Professor: **Hussein Al Osman**

Android Project: Service Novigrad

Family Name, Name	Student ID
Vu, Sophie	300113938
Bayirli, Mustafa	300144781
Hibbert, Grayden	300074064
Polak, Anthony	300119082

December 6, 2020

Table of Contents

Table of Contents 1

Introduction 2

UML Class Diagram 3

Discussion of Lessons Learned 4

How to create a project environment on Github to all team members. 4

How to manage the workload successfully by partitioning the work between members. 4

How to design UML diagrams to depict a completed program and base the work and steps required to create an accurate representation of the UML diagram in practice. How to update the UML diagram to better reflect the state of our code when the code had to deviate from the expected path. 5

How to use database support like Firebase or SQLite in our project. 5

How to implement unit tests to evaluate and test our application. 6

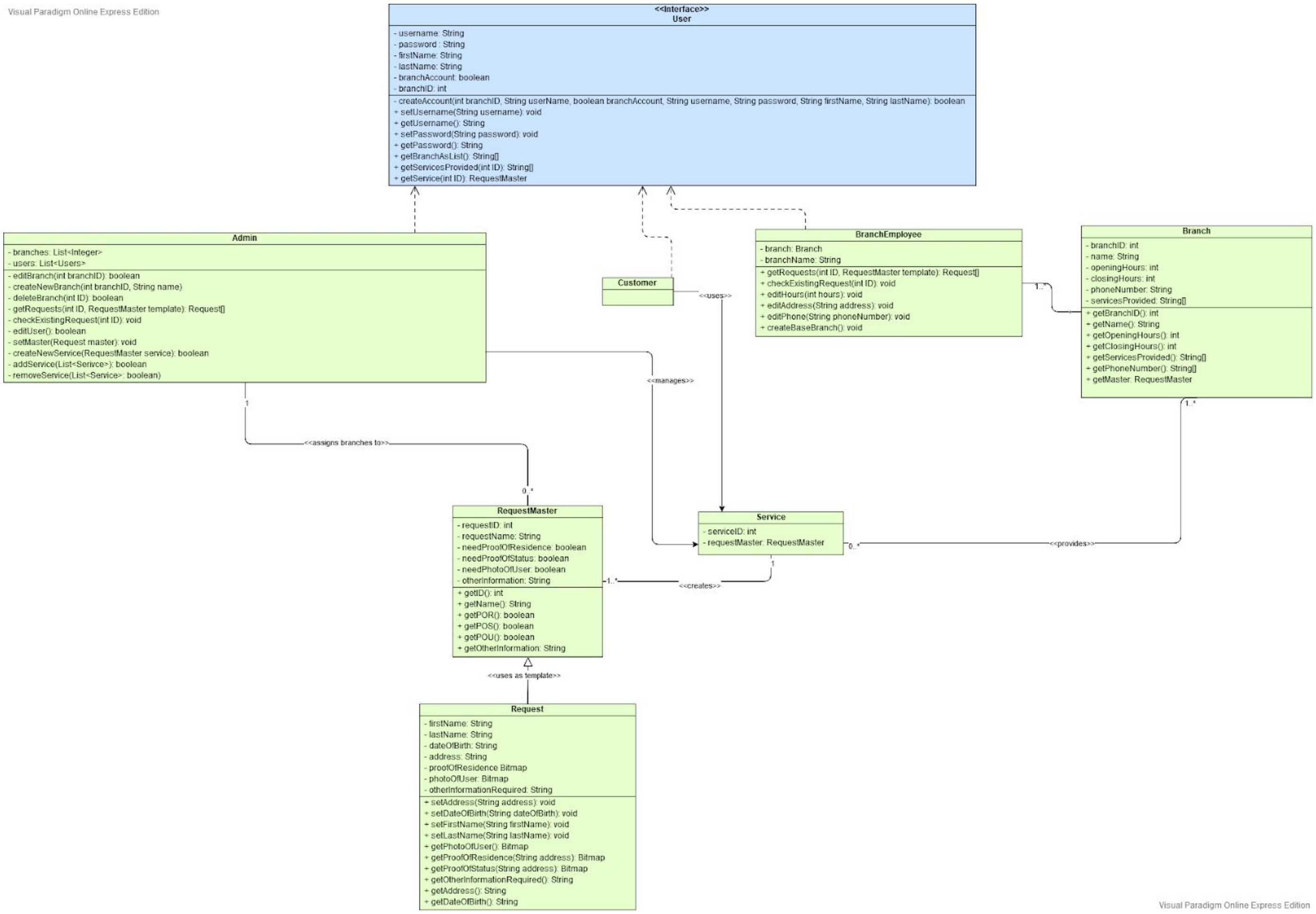
Roles and contributions of team members 7

Screenshots of the app 8

Introduction

Our task for this project was to create a software for the fictional province of Novigrad. The software must allow the Novigrad administration to provide specific government services to its residents. To reach this goal, we tested and developed an application that is able to run on Android devices called Service Novigrad. The development process of the application was divided into four deliverables, each consisting of different functionalities to implement. The first deliverable involved creating three different user types (Admin, BranchEmployee and Customer). By doing so, we were able to restrict the features that would be available for each type of user. The following three deliverables consisted of programming the functions and services that each user type is provided. We specified administration functionalities for the second deliverable. We then created user functionalities for the third deliverable and finalised the application for the fourth deliverable. The gradual build-up of the project allowed us to improve the application's functionality since each deliverable required the implementation of the experiences that were learned in this course as it was taught. The full description of the application's features can be found in the Project Description.

UML Class Diagram


Visual Paradigm Online Express Edition
<<Interface>>
User
- username: String
- password : String
- firstName: String
- lastName: String
- branchAccount: boolean
- branchID: int
- createAccount(int branchID, String userName, boolean branchAccount, String username, String password, String firstName, String lastName): boolean
+ setUsername(String username): void
+ getUsername(): String
+ setPassword(String password): void
+ getPassword(): String
+ getBranchAsList(): String[]
+ getServicesProvided(int ID): String[]
+ getService(int ID): RequestMaster
Admin
- branches: List<Integer>
- users: List<Users>
- editBranch(int branchID): boolean
- createNewBranch(int branchID, String name)
- deleteBranch(int ID): boolean
- getRequests(int ID, RequestMaster template): Request[]
- checkExistingRequest(int ID): void
- editUser(): boolean
- setMaster(Request master): void
- createNewService(RequestMaster service): boolean
- addService(List<Serivce>): boolean
- removeService(List<Service>: boolean)
Customer
<<uses>>
BranchEmployee
- branch: Branch
- branchName: String
+ getRequests(int ID, RequestMaster template): Request[]
+ checkExistingRequest(int ID): void
+ editHours(int hours): void
+ editAddress(String address): void
+ editPhone(String phoneNumber): void
+ createBaseBranch(): void
1..*
Branch
- branchID: int
- name: String
- openingHours: int
- closingHours: int
- phoneNumber: String
- servicesProvided: String[]
+ getBranchID(): int
+ getName(): String
+ getOpeningHours(): int
+ getClosingHours(): int
+ getServicesProvided(): String[]
+ getPhoneNumber(): String[]
+ getMaster: RequestMaster
1..*
<<manages>>
1
<<assigns branches to>>
0..*
RequestMaster
- requestID: int
- requestName: String
- needProofOfResidence: boolean
- needProofOfStatus: boolean
- needPhotoOfUser: boolean
- otherInformation: String
+ getID(): int
+ getName(): String
+ getPOR(): boolean
+ getPOS(): boolean
+ getPOU(): boolean
+ getOtherInformation: String
Service
- serviceID: int
- requestMaster: RequestMaster
0..*
<<provides>>
1
1..*
<<creates>>
<<uses as template>>
Request
- firstName: String
- lastName: String
- dateOfBirth: String
- address: String
- proofOfResidence Bitmap
- photoOfUser: Bitmap
- otherInformationRequired: String
+ setAddress(String address): void
+ setDateOfBirth(String dateOfBirth): void
+ setFirstName(String firstName): void
+ setLastName(String lastName): void
+ getPhotoOfUser(): Bitmap
+ getProofOfResidence(String address): Bitmap
+ getProofOfStatus(String address): Bitmap
+ getOtherInformationRequired(): String
+ getAddress(): String
+ getDateOfBirth(): String
Visual Paradigm Online Express Edition

Discussion of Lessons Learned

In this project, we learned different lessons about how splitting the project into four phases helped us to understand each phase on itself and trying not to move to the next phase until we complet what we are working on. Also as the project covers the entire semester, it is good to implement the project as our course moves on. That ensures and applies our knowledge on each step. The following are some of the lessons we learned while working on this project.

1. **How to create a project environment on Github to all team members.**

   We learned this in deliverable 1 as not every member of the team knew how to use github. Setting up the team in Github was fairly easy considering there was documentation and instructions provided that demonstrated how to set up and join a team. There was some issue involving importing the project for a couple of members, however after this was solved after some time and it never presented itself again. All in all, setting up and using github was an easy, but important, lesson to learn from this project.

2. **How to manage the workload successfully by partitioning the work between members.**

   Each person in the group had a specific role when it came to managing tasks and what asked in the deliverable. This allowed up to work without accidentally writing the same or conflicting code which could potentially create problems and increase the workload on the members unnecessarily. This is a practice generally done in real work environments, where each employee has a specific area that they operate in, so learning how to do this as a team was a worthwhile endeavor.

3. **How to design UML diagrams to depict a completed program and base the work and steps required to create an accurate representation of the UML diagram in practice. How to update the UML diagram to better reflect the state of our code when the code had to deviate from the expected path.**
    - How the Classes, Association, Attributes Operations, Generalizations successfully implemented and worked.
    - Domain analysis, to get sufficient information about the project.
    - Define the requirement, analyze and sort it according to its functionality.
    - How the use-cases applied and helped us to understand how the user will interact with the entire system to execute a specific task.

4. **How to use database support like Firebase or SQLite in our project.**

We learned how to use databases in our project. In particular, we used SQLite database to create 4 tables that correspond to 4 important aspects of the project that need to be saved and retrieved frequently. We used a table to store user data such as their usernames and passwords, first and last names, user type (such as admin or customer), and the associated branch (if the user type was of branch employee). The remaining tables store branch data, service data, and request data. Using a SQLite database was very useful as it allowed us to access any of the data in the database from anywhere in the application, and it allowed us to manipulate and use the data easier than if we were to try and save it locally. Database management was a great lesson to learn, especially considering how databases are a core component of programming for large applications.

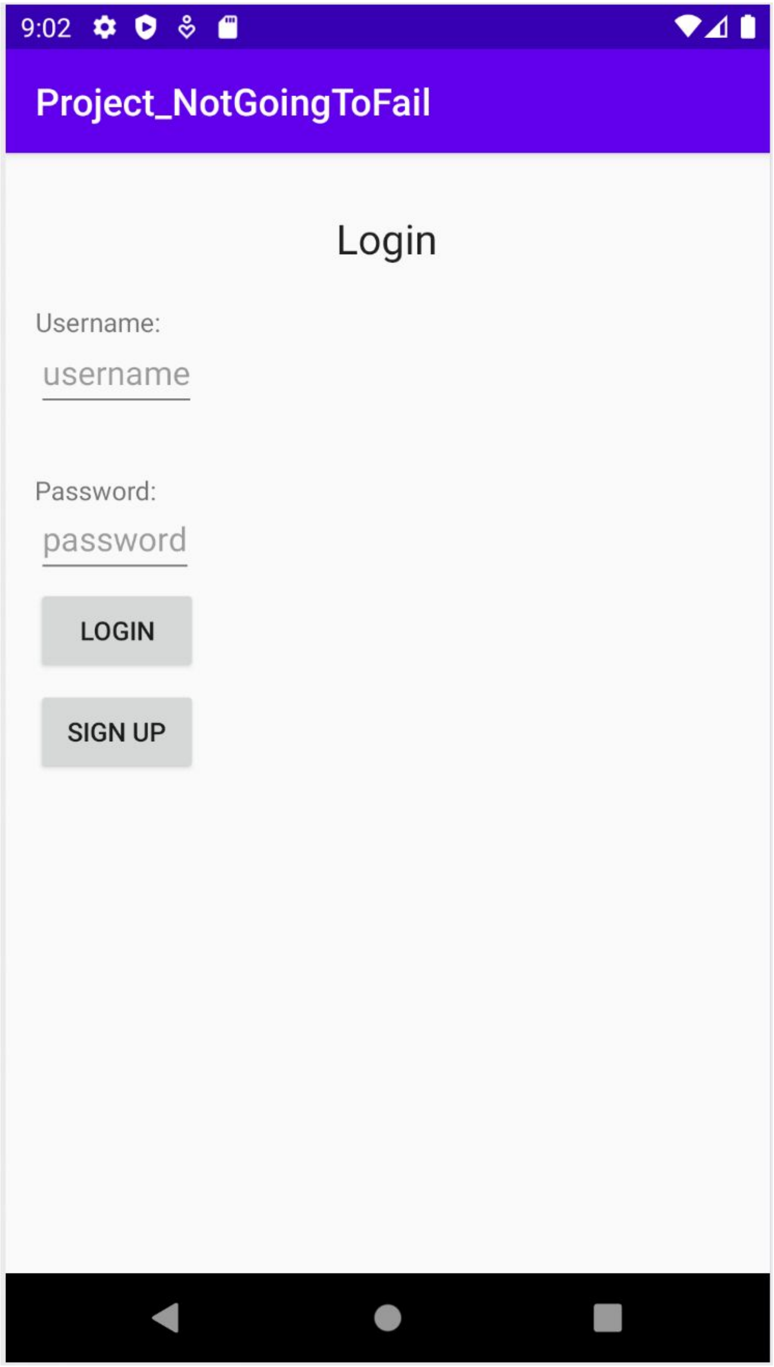
5. How to implement unit tests to evaluate and test our application.

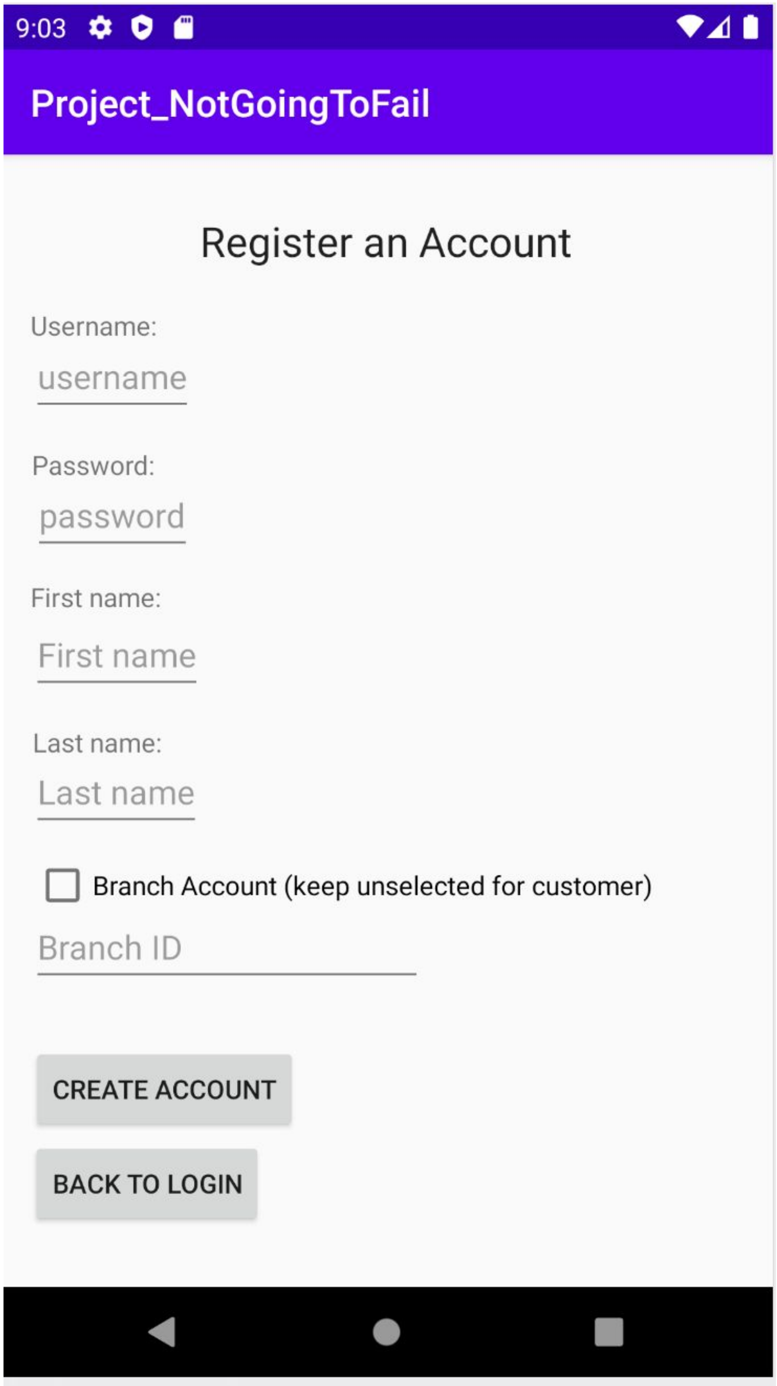
Implementing unit tests to test out application was an important aspect of what we learned. This is because unit tests can continuously test the application to make sure that it is performing in the intended manner, even after code may have been changed. For example, when the administrator user was created in the register activity instead of being predefined there were some checks in place to make sure that the branch class had an associated id for a respective branch, as a branch employee cannot exist without being an employee at a branch. When the administrator register was removed and made so that it was predefined credentials, a check to make sure that a branch employee had an associated branch id also affected customers registering for an account. Unit tests should have caught that issue, however there were none implemented at that time, so a bug ended up being in the project for weeks without being fixed. The lesson to be learned here is that unit testing helps solve issues as they arrive and not after they are found.

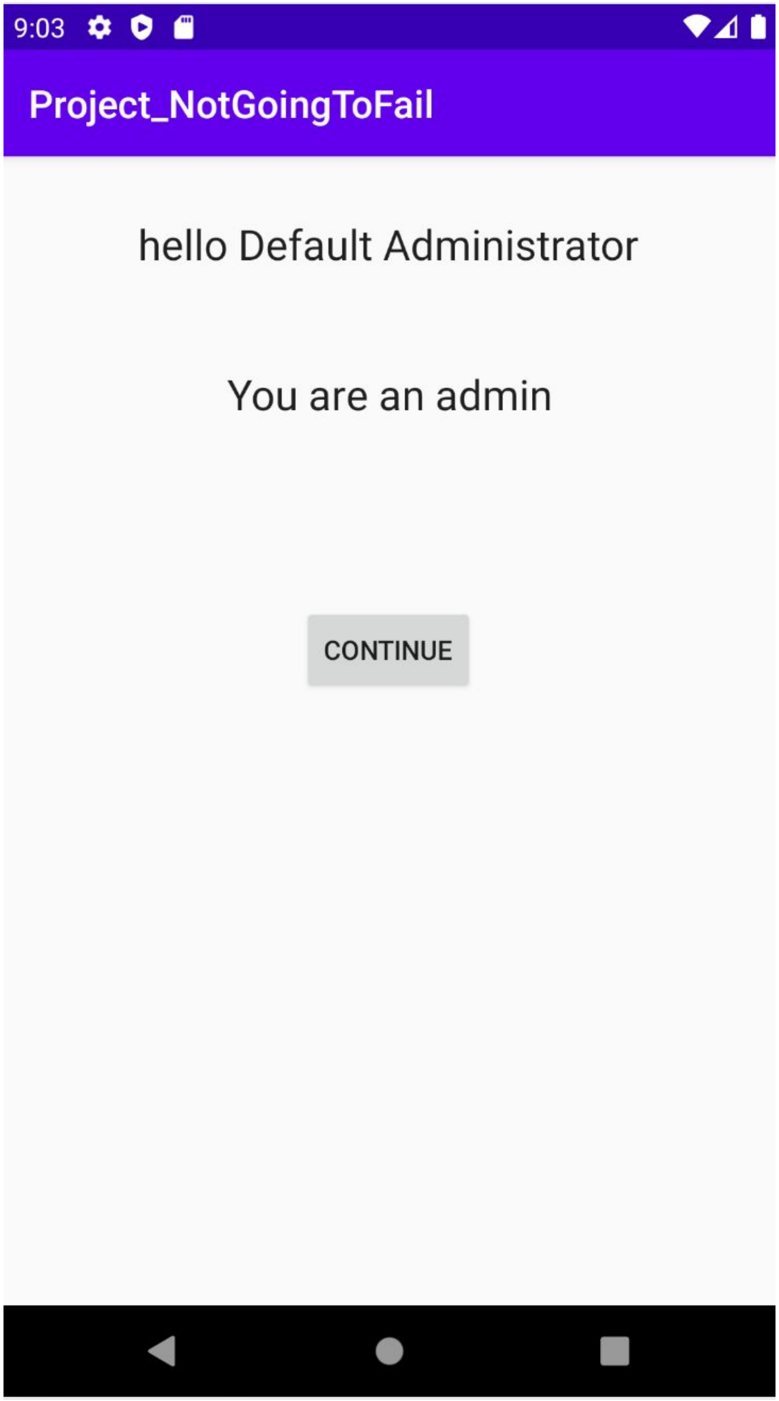
Roles and contributions of team members

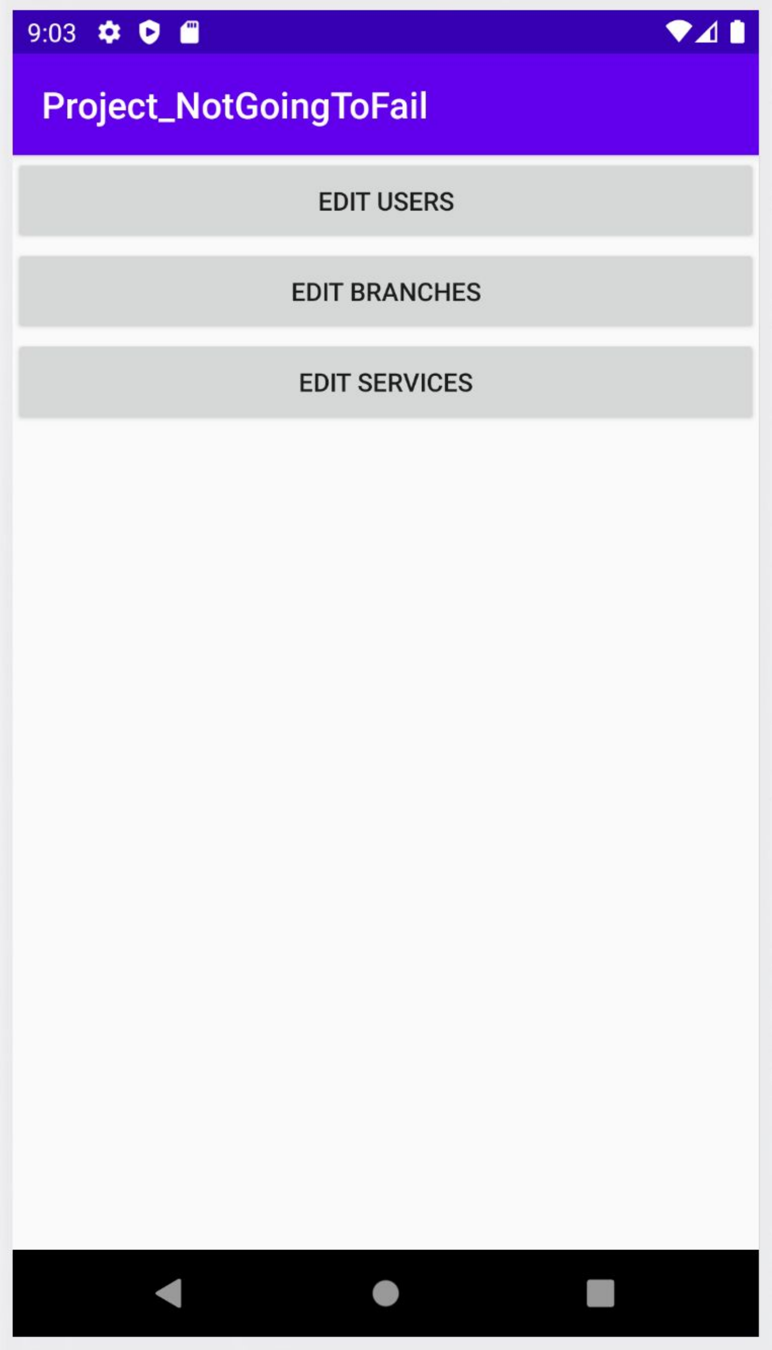
	Deliverable 1	**Deliverable 2**	**Deliverable 3**	**Deliverable 4**
Vu, Sophie	Worked on UML class diagram	Updated UML class diagram	Updated UML class diagram, worked on Junit tests	Finalised UML class diagram, worked on report
Bayirli, Mustafa	Worked on UML class diagram	Worked on UML Class diagram	Worked On UML Class diagram, junit test	Worked on Final report
Hibbert, Grayden	Created github team and repo, Wrote database code for userData, activity code for project	Wrote database code for branchData, activity code for project	Wrote database code for serviceData, activity code for project	Wrote database code for requestData, activity code for project
Polak, Anthony	Wrote activity code, worked on updating the gradle, worked on System Domain Diagram	Worked on code for creating branches in the admin function, and worked on fixing errors in the code.	Wrote Service code, Made XML files for classes	Helped with customer service Request code and branch service code, made XML files for said classes

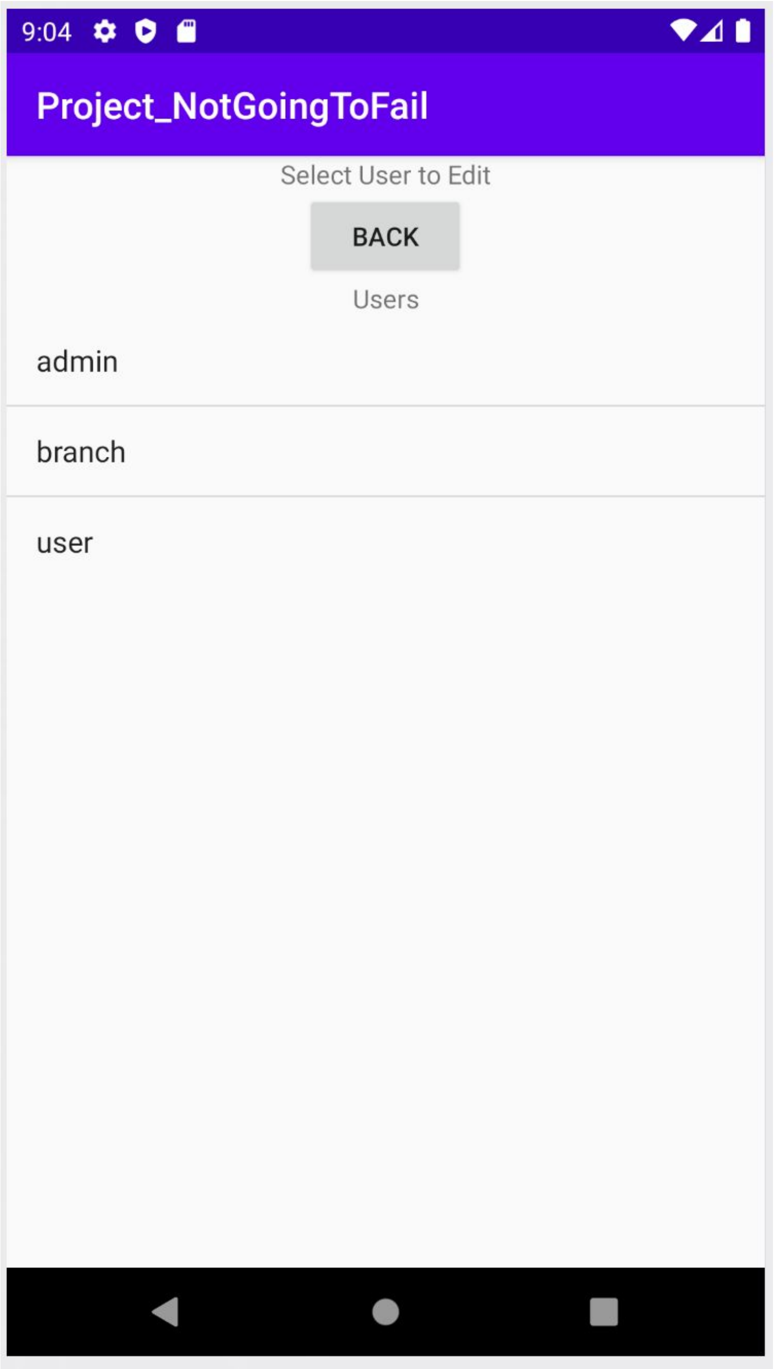
Screenshots of the app

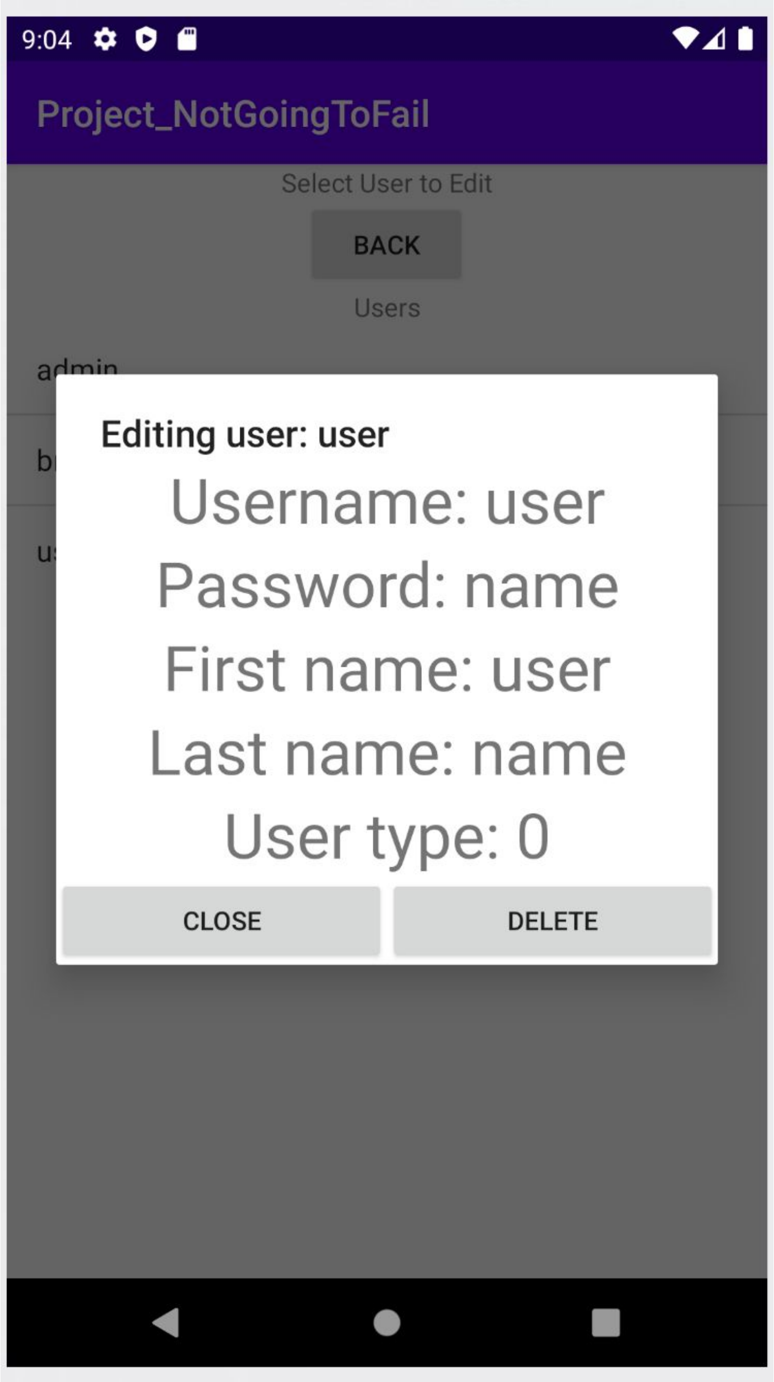
Description of Image	Image
Login	 9:02 Project_NotGoingToFail Login Username: username Password: password LOGIN SIGN UP

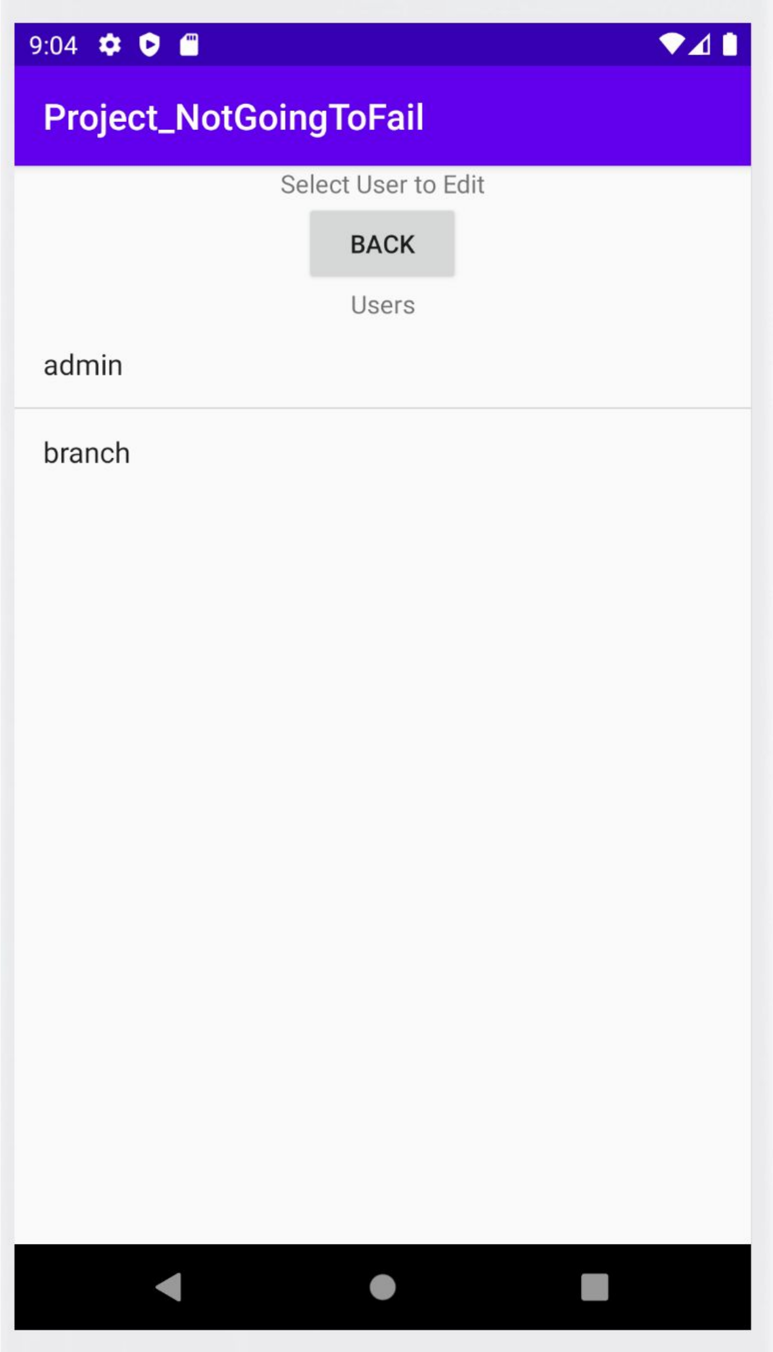
Register	 Project_NotGoingToFail Register an Account Username: username Password: password First name: First name Last name: Last name □ Branch Account (keep unselected for customer) Branch ID

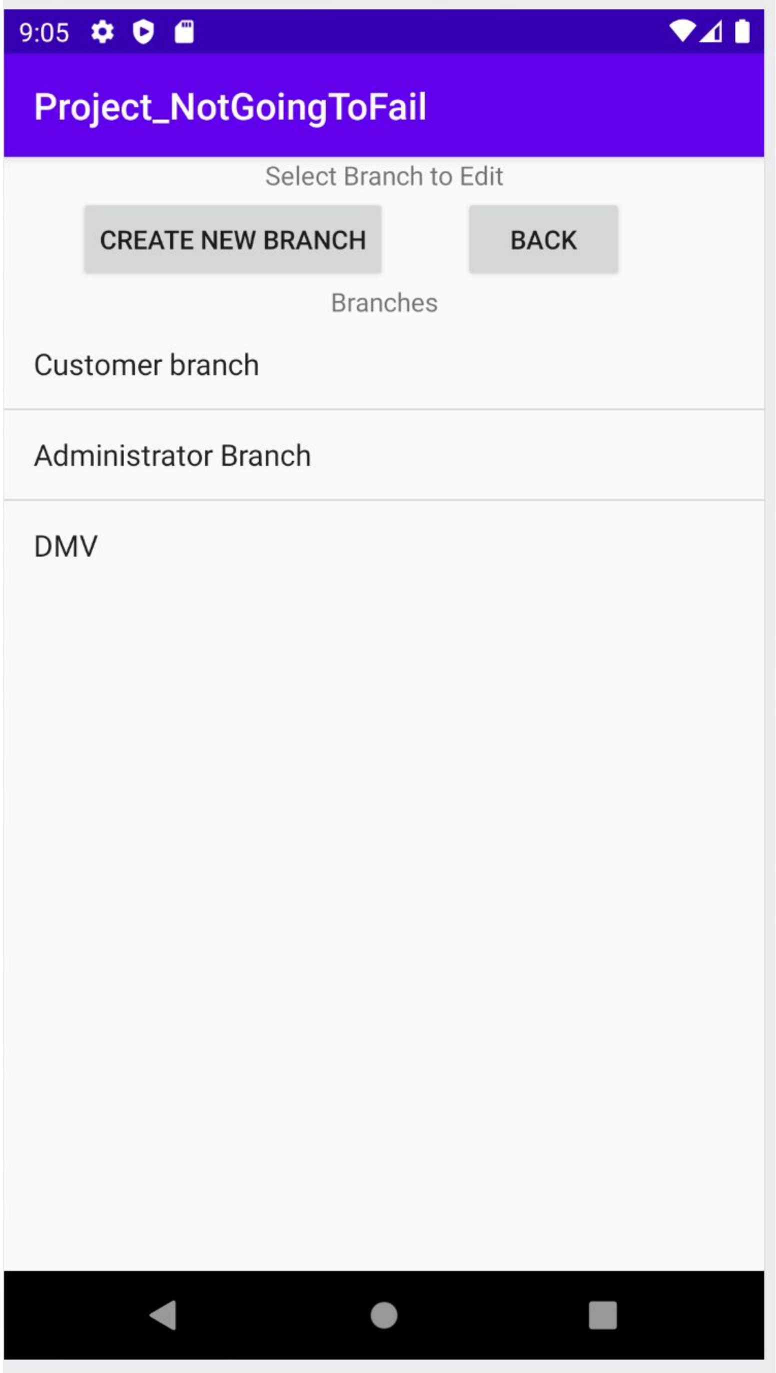
Admin Splash Page	 hello Default Administrator You are an admin

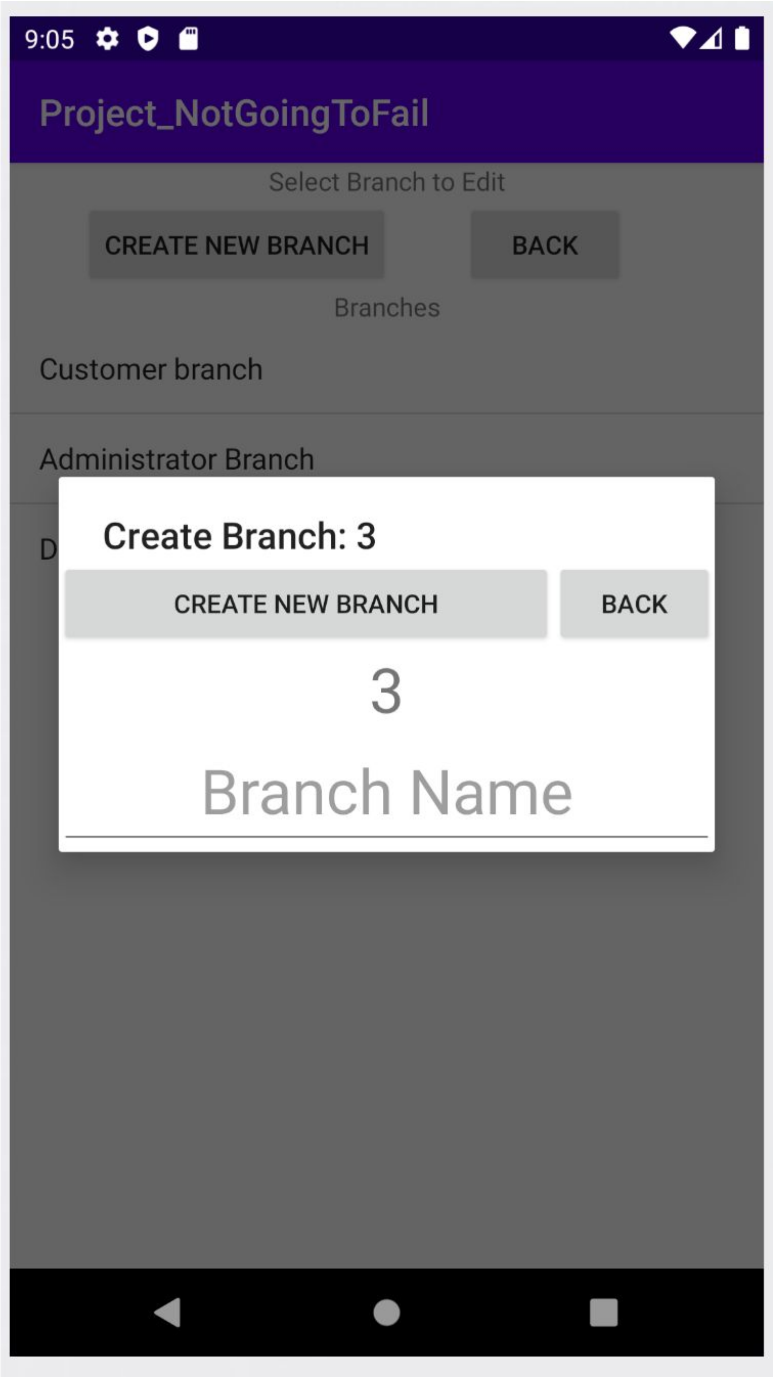
Admin Option Page	 9:03 Project_NotGoingToFail EDIT USERS EDIT BRANCHES EDIT SERVICES

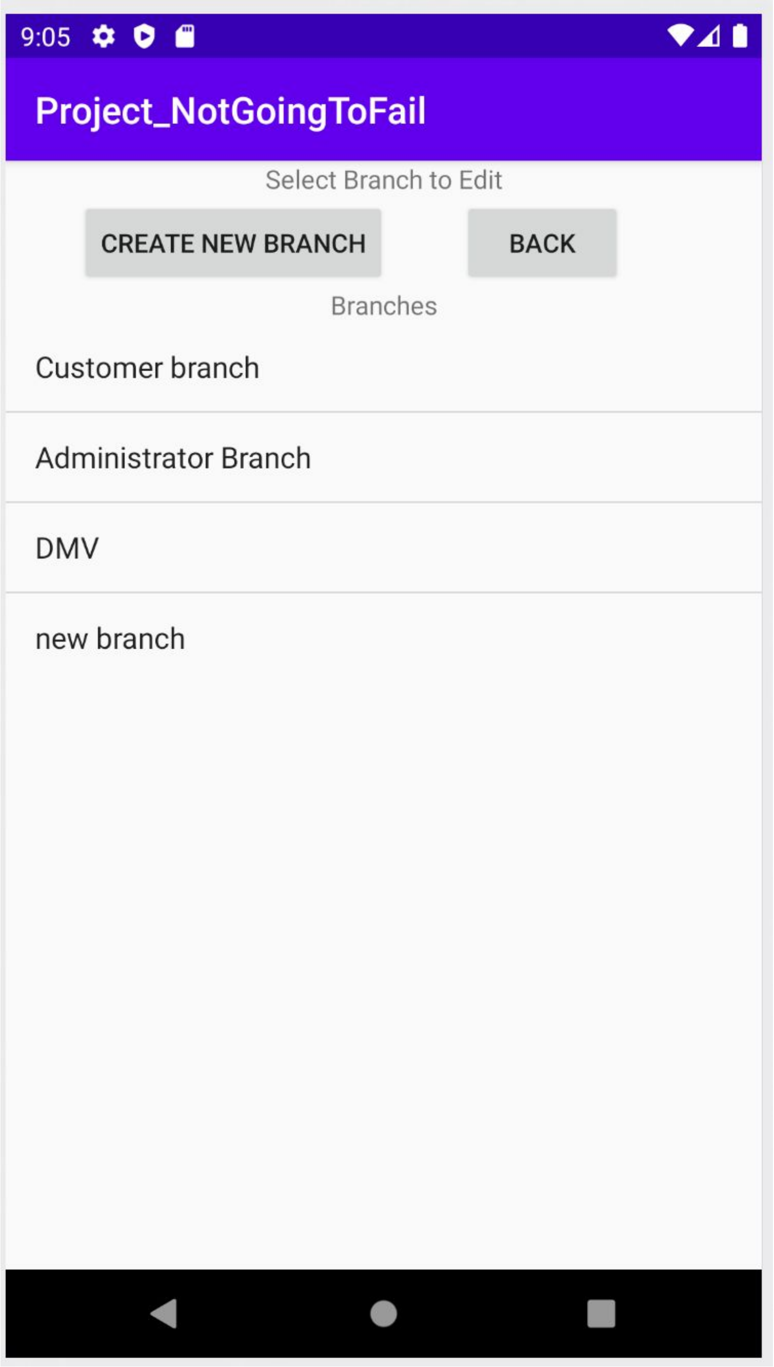
Admin Edit User	 9:04 Project_NotGoingToFail Select User to Edit BACK Users admin branch user

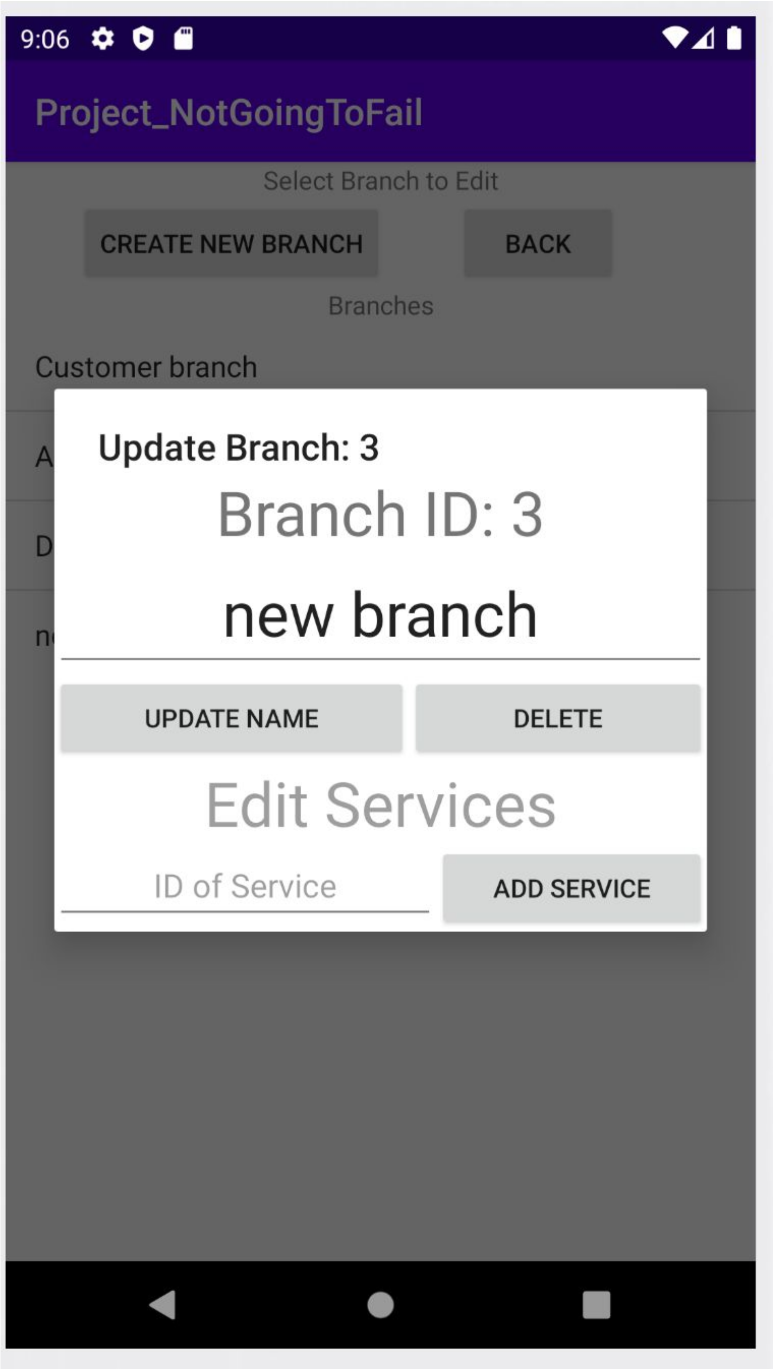
Admin Edit User Detailed	 9:04 Project_NotGoingToFail Select User to Edit BACK Users admin Editing user: user Username: user Password: name First name: user Last name: name User type: 0

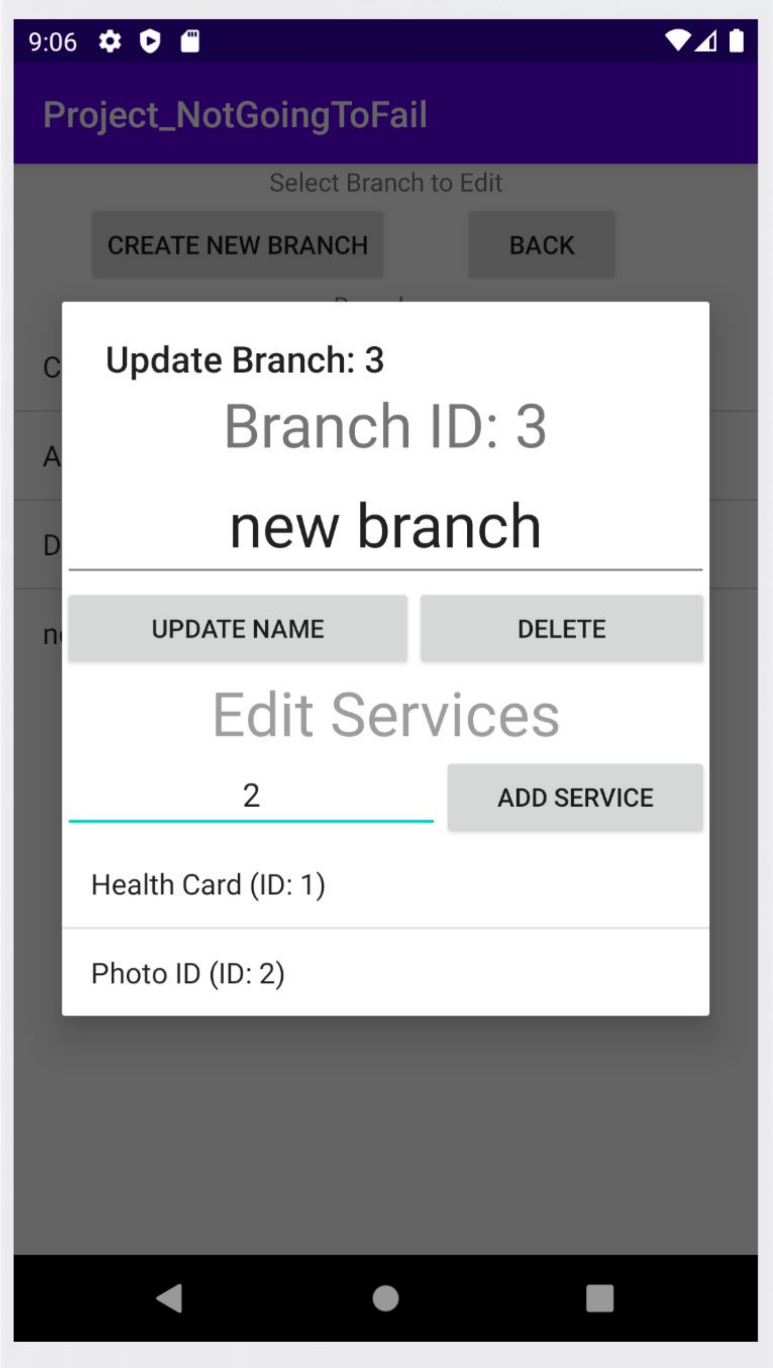
Admin Deleted User (user is gone)	 9:04 Project_NotGoingToFail Select User to Edit BACK Users admin branch

Admin Branch Options	 9:05 Project_NotGoingToFail Select Branch to Edit CREATE NEW BRANCH BACK Branches Customer branch Administrator Branch DMV

Admin Create New Branch	 9:05 Project_NotGoingToFail Select Branch to Edit CREATE NEW BRANCH BACK Branches Customer branch Administrator Branch D Create Branch: 3 CREATE NEW BRANCH BACK 3 Branch Name

Admin New Branch (new branch has been added)	 9:05 Project_NotGoingToFail Select Branch to Edit CREATE NEW BRANCH BACK Branches Customer branch Administrator Branch DMV new branch

Admin Edit Branch	 9:06 Project_NotGoingToFail Select Branch to Edit Branches Customer branch A D n Update Branch: 3 Branch ID: 3 new branch Edit Services ID of Service

Admin Edit Branch (Add Service)	

Update Branch: 3

Branch ID: 3

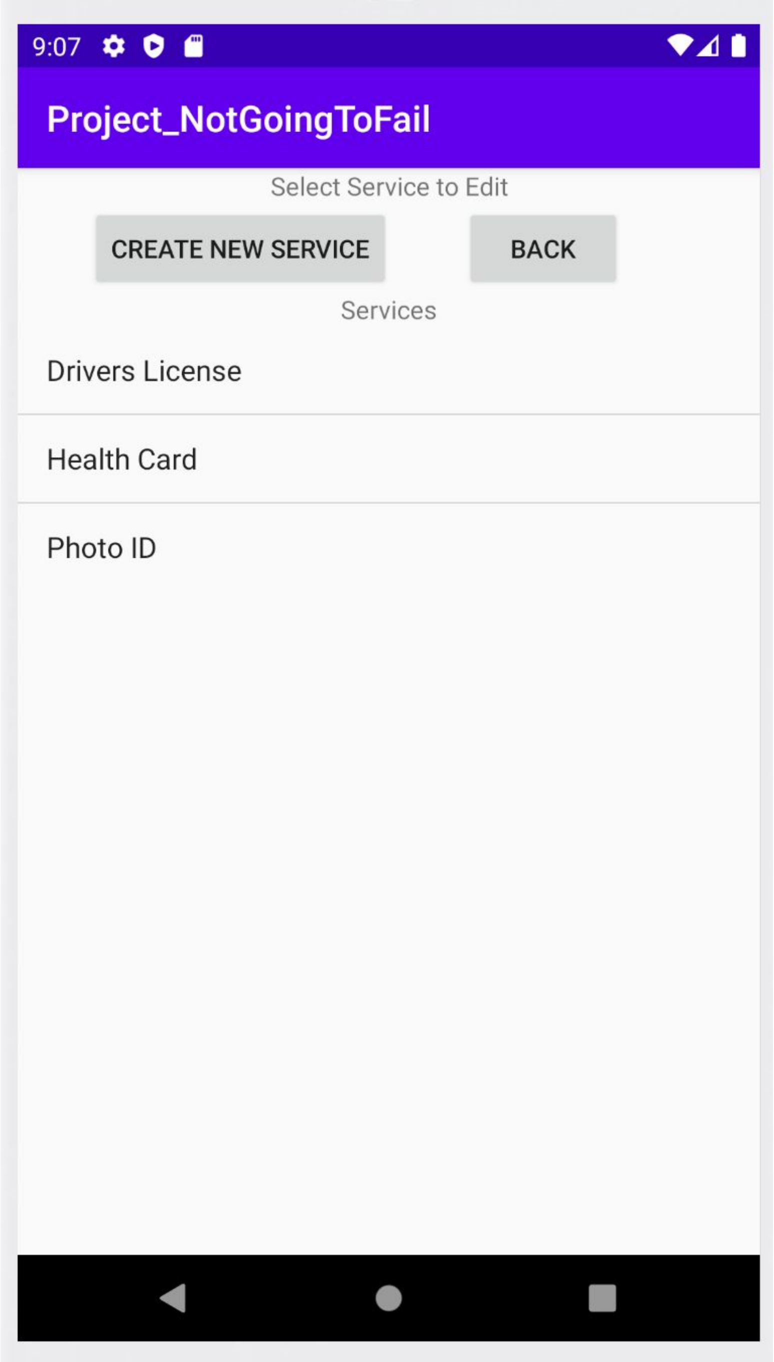
new branch

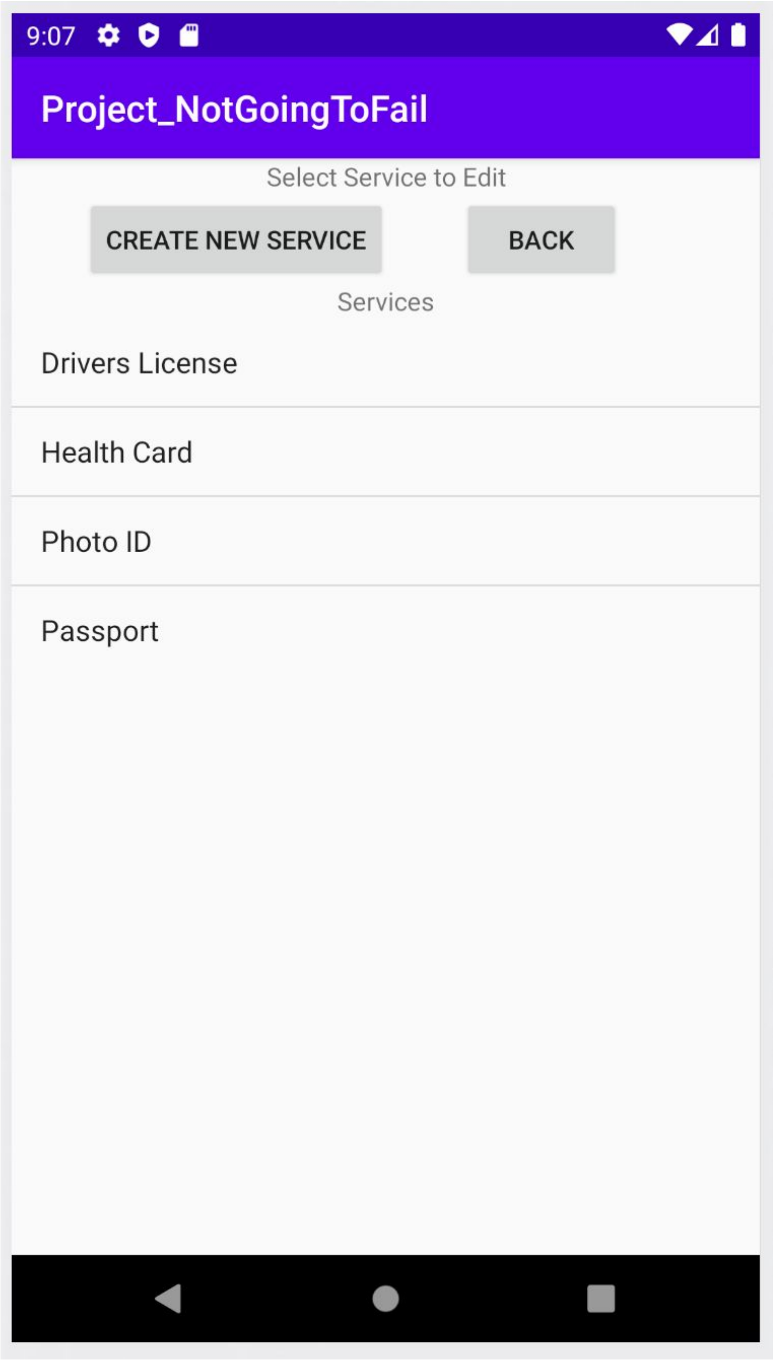
Edit Services

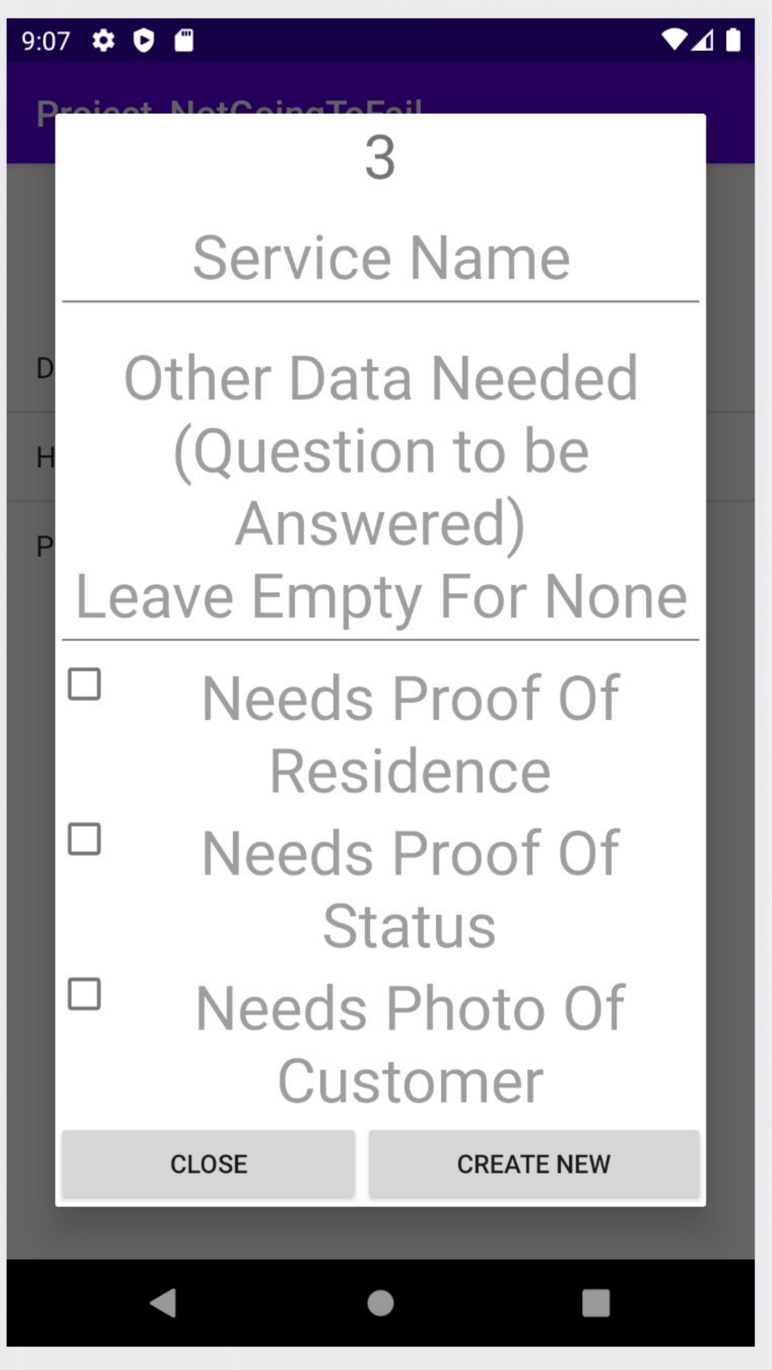
2

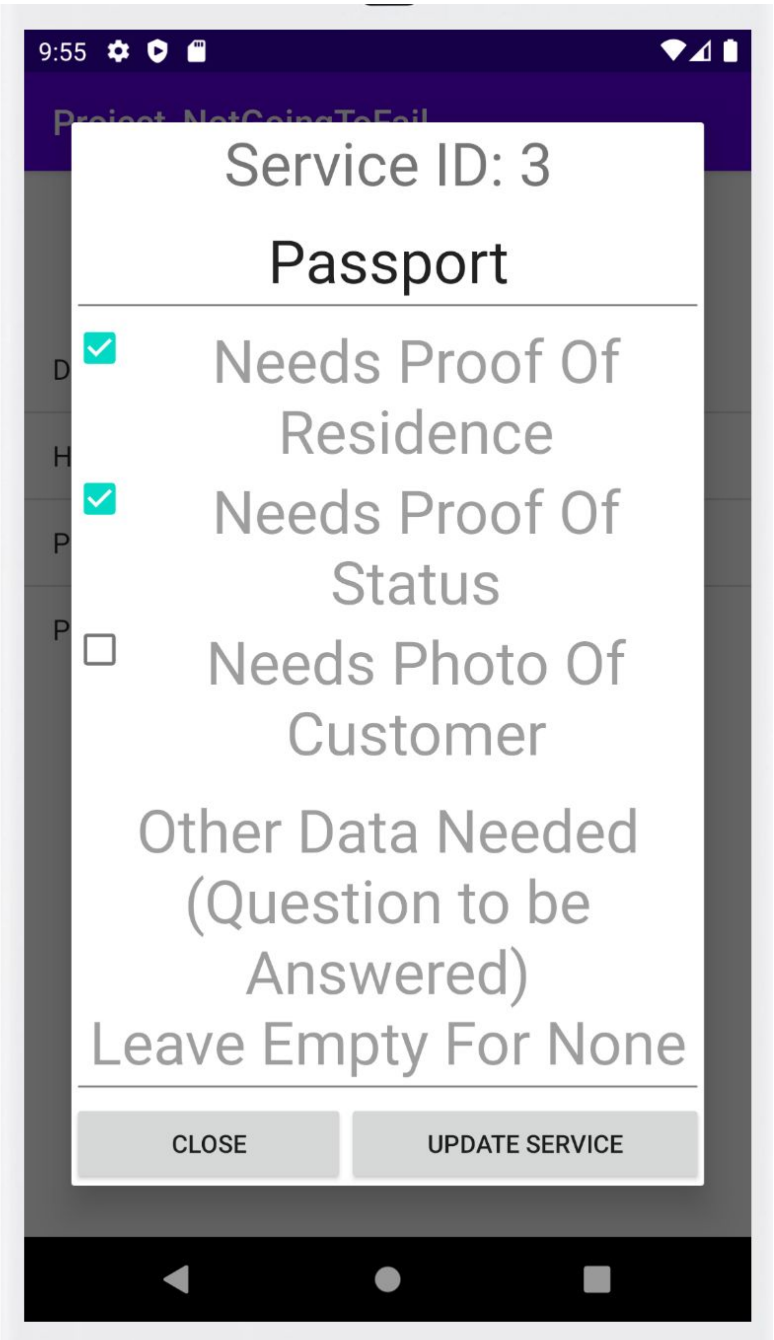
Health Card (ID: 1)

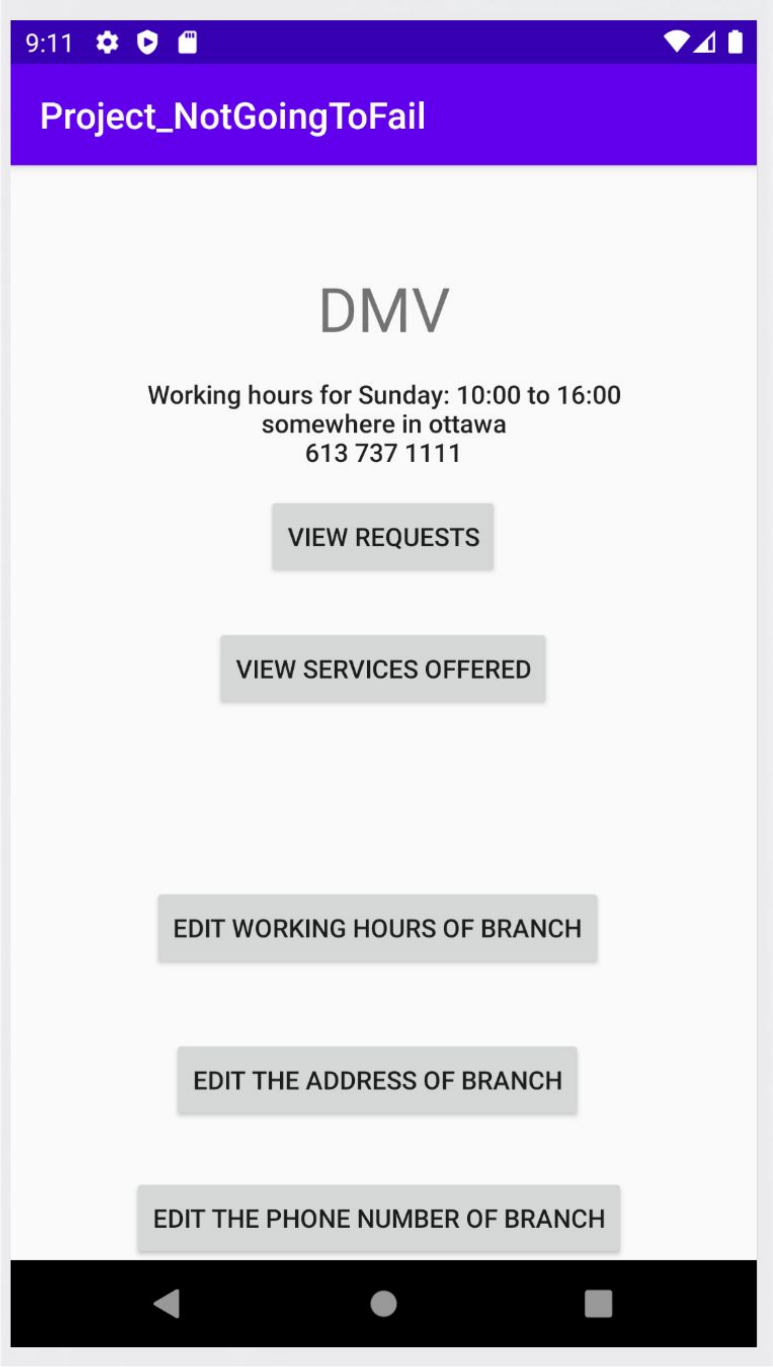
Photo ID (ID: 2)

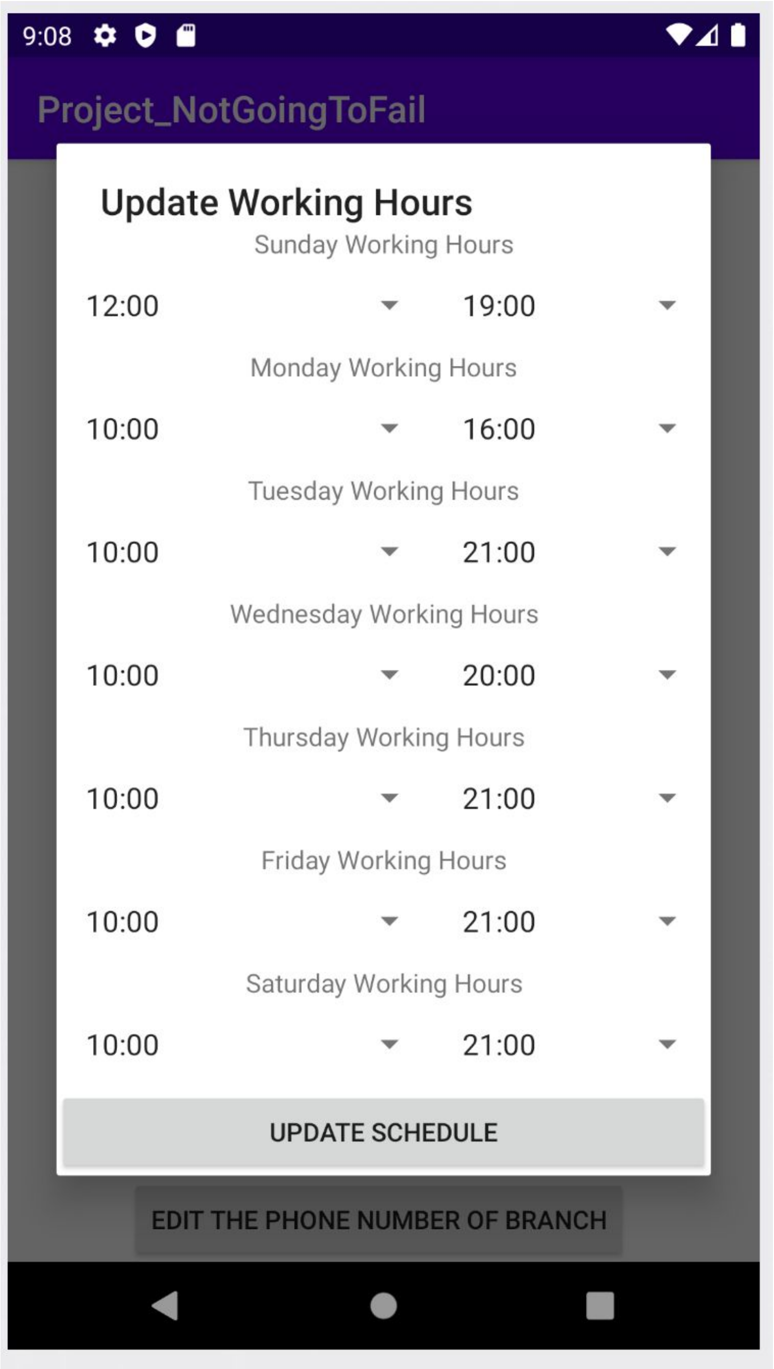
Admin Services Option	 9:07 Project_NotGoingToFail CREATE NEW SERVICE BACK Select Service to Edit Services Drivers License Health Card Photo ID

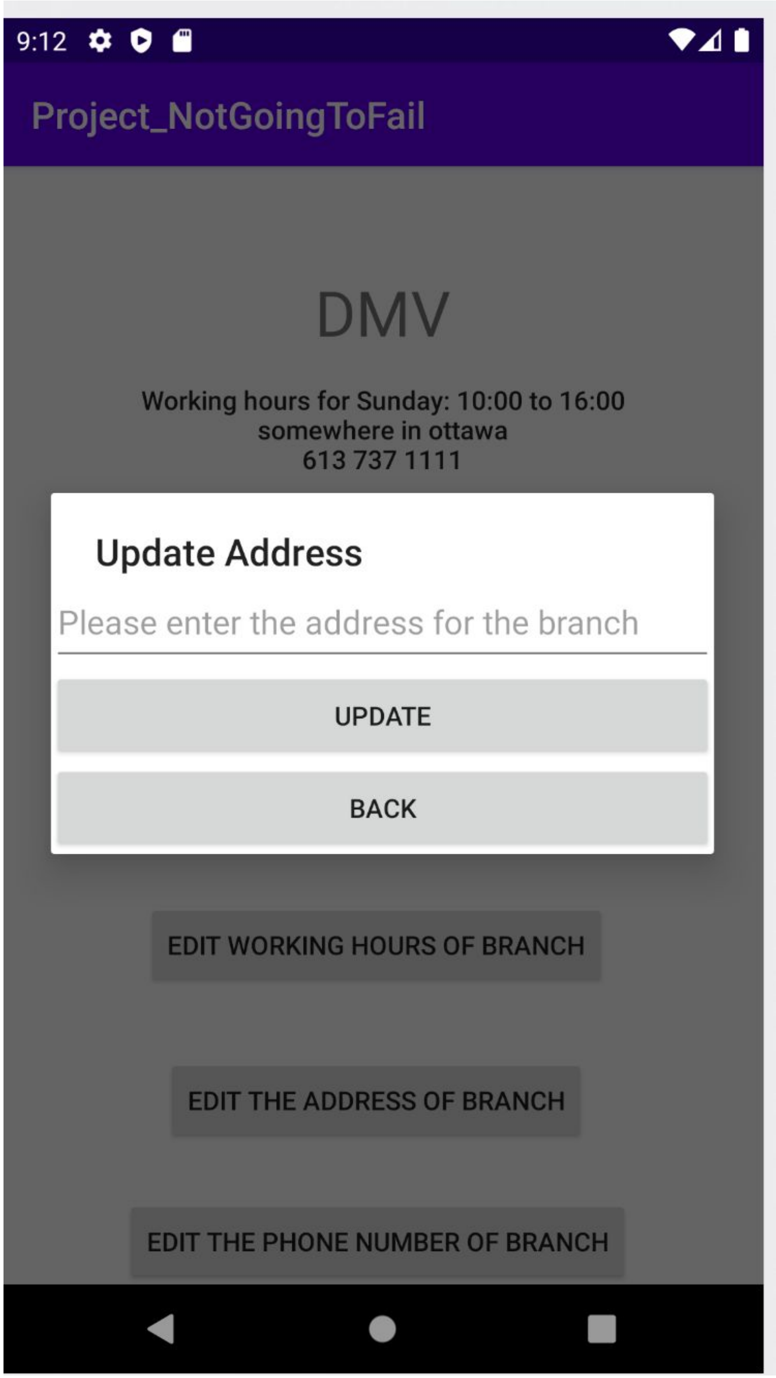
Admin New Service	 Project_NotGoingToFail Select Service to Edit Services Drivers License Health Card Photo ID Passport

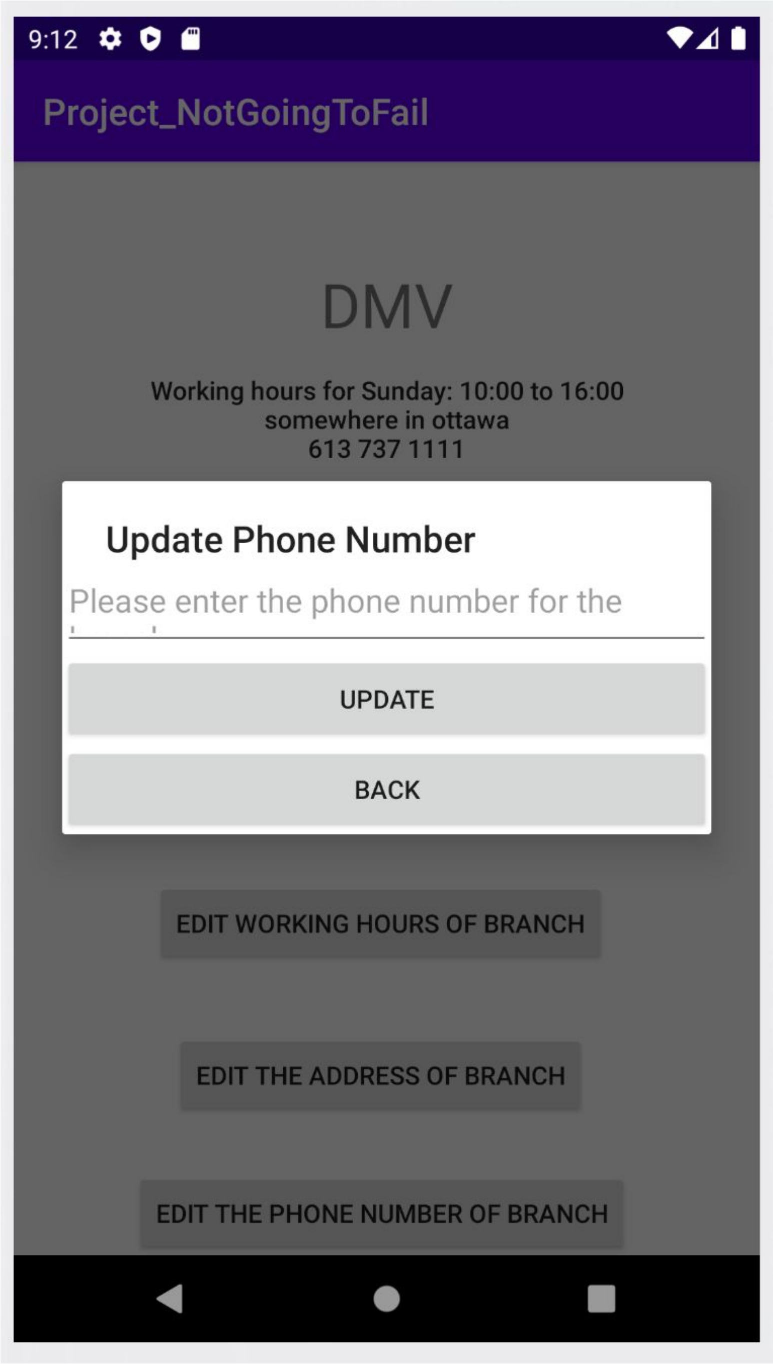
Admin New Service Menu	 9:07 3 Service Name Other Data Needed (Question to be Answered) Leave Empty For None □ Needs Proof Of Residence □ Needs Proof Of Status □ Needs Photo Of Customer CLOSE CREATE NEW

Admin Edit Service	 Service ID: 3 Passport ☑ Needs Proof Of Residence ☑ Needs Proof Of Status ☐ Needs Photo Of Customer Other Data Needed (Question to be Answered) Leave Empty For None

Branch Options	 Project_NotGoingToFail DMV Working hours for Sunday: 10:00 to 16:00 somewhere in ottawa 613 737 1111 VIEW REQUESTS VIEW SERVICES OFFERED EDIT WORKING HOURS OF BRANCH EDIT THE ADDRESS OF BRANCH EDIT THE PHONE NUMBER OF BRANCH

Branch Edit Working Hours	 9:08 Project_NotGoingToFail Update Working Hours Sunday Working Hours 12:00 ▾ 19:00 ▾ Monday Working Hours 10:00 ▾ 16:00 ▾ Tuesday Working Hours 10:00 ▾ 21:00 ▾ Wednesday Working Hours 10:00 ▾ 20:00 ▾ Thursday Working Hours 10:00 ▾ 21:00 ▾ Friday Working Hours 10:00 ▾ 21:00 ▾ Saturday Working Hours 10:00 ▾ 21:00 ▾ UPDATE SCHEDULE EDIT THE PHONE NUMBER OF BRANCH

Branch Edit Address	 9:12 Project_NotGoingToFail DMV Working hours for Sunday: 10:00 to 16:00 somewhere in ottawa 613 737 1111 Update Address Please enter the address for the branch UPDATE BACK EDIT WORKING HOURS OF BRANCH EDIT THE ADDRESS OF BRANCH EDIT THE PHONE NUMBER OF BRANCH

Branch Edit Phone Number	

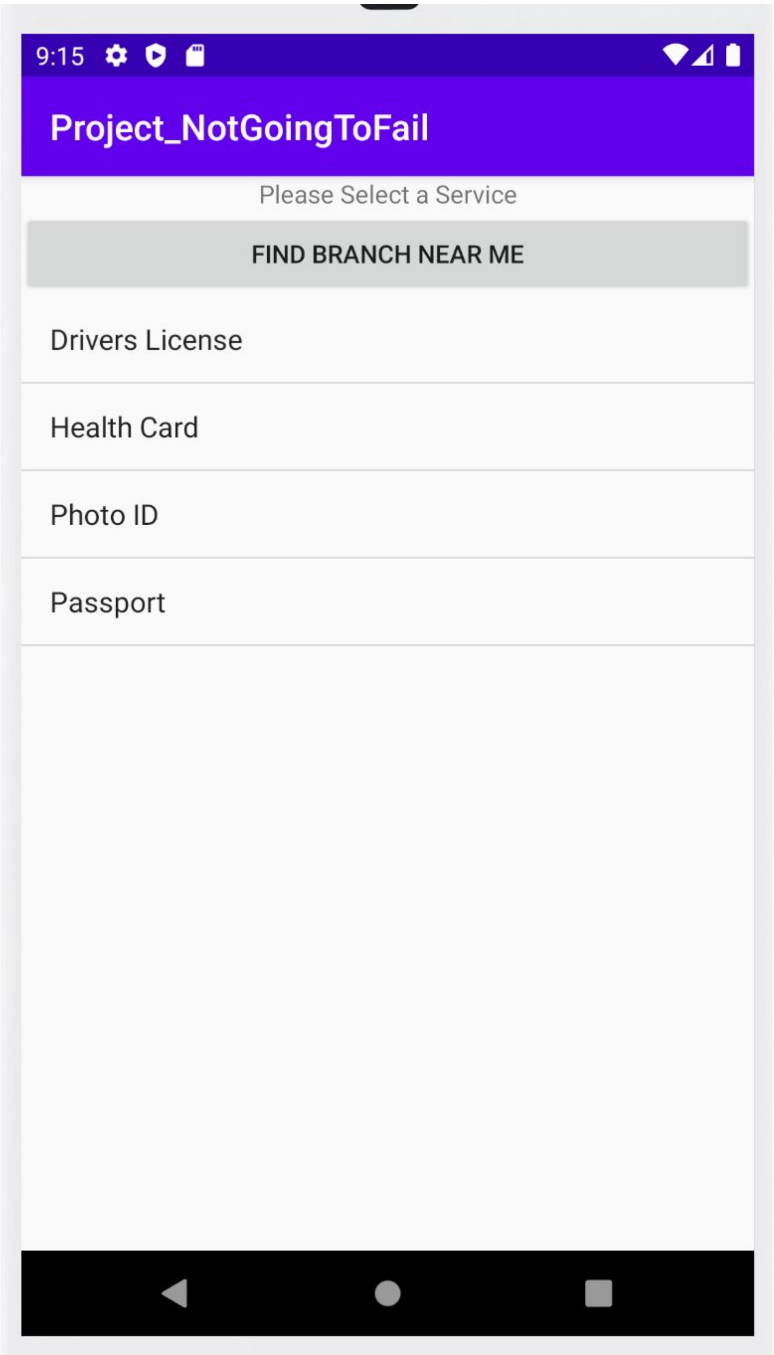
DMV

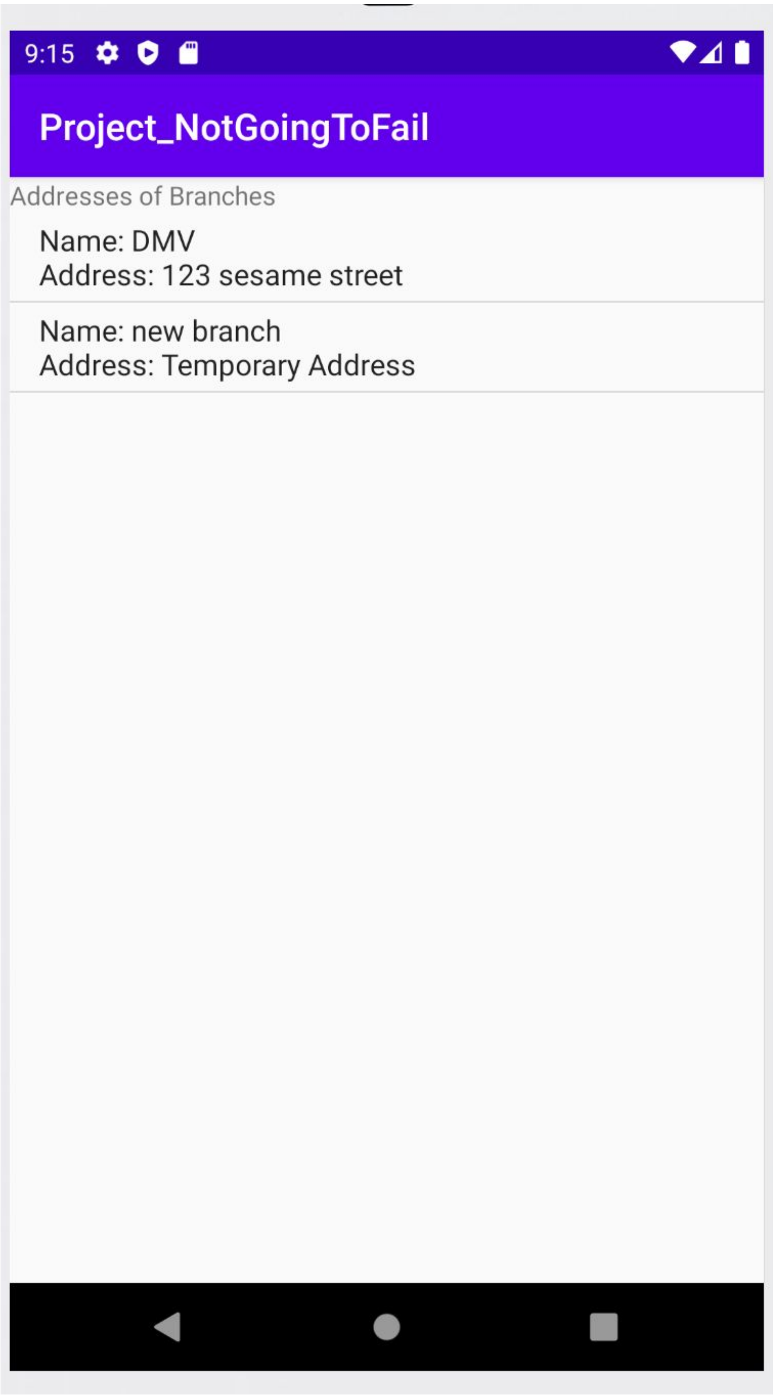
Working hours for Sunday: 10:00 to 16:00
somewhere in ottawa
613 737 1111

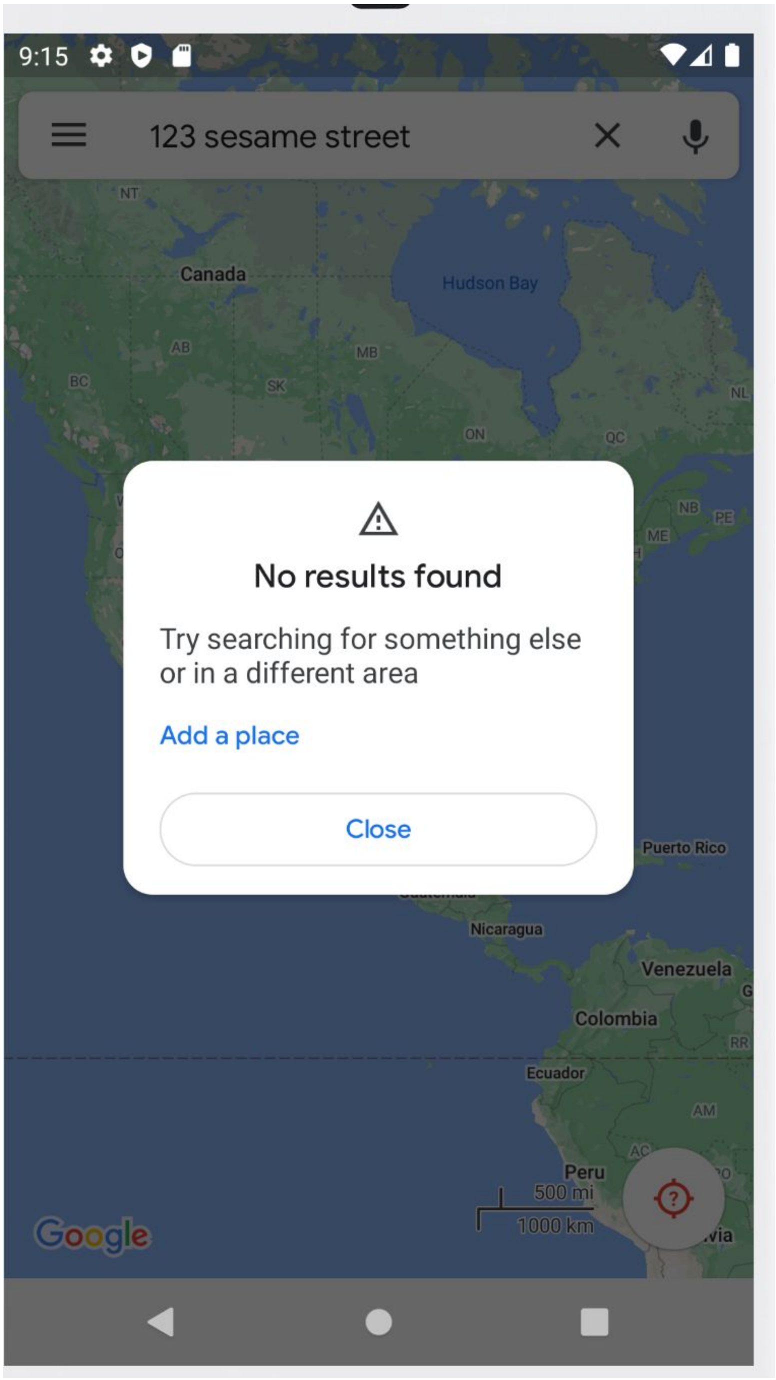
Update Phone Number

Please enter the phone number for the

9:12
Project_NotGoingToFail
UPDATE
BACK
EDIT WORKING HOURS OF BRANCH
EDIT THE ADDRESS OF BRANCH
EDIT THE PHONE NUMBER OF BRANCH

Customer Login	 9:15 Project_NotGoingToFail Please Select a Service FIND BRANCH NEAR ME Drivers License Health Card Photo ID Passport

Customer Find Branch Near Me	 Addresses of Branches Name: DMV Address: 123 sesame street Name: new branch Address: Temporary Address

Customer Open Map	 9:15 123 sesame street No results found Try searching for something else or in a different area Add a place Close NT Canada Hudson Bay AB MB BC SK NL ON QC NB PE ME Puerto Rico Nicaragua Venezuela Colombia RR Ecuador AM AC Peru 500 mi 1000 km Google