Roopairs: Software Architecture version 1.0

Rooio
Computer Science Department
California Polytechnic State University
San Luis Obispo, CA USA

November 6, 2019

Contents

Credits

Name	Date	Role	Version
Jessica Chang	November 6, 2019	Co-author of Software Architecture	1.0
Luke Reckard	November 6, 2019	Co-author of Software Architecture	1.0
Karla Sunjara	November 6, 2019	Co-author of Software Architecture	1.0
Logan Lawson	November 6, 2019	Co-author of Software Architecture	1.0
Yusuf Bahadur	November 6, 2019	Co-author of Software Architecture	1.0

Revision History

Name	Date	Reason for Changes	Version

1 Introduction

This document describes the architecture of our solution for the Repairs Clover application and details the different components, the software that resides within them, and the relationship between said components.

The main components that comprise the solution are the Clover POS, an integrated point-of-sale system that the application will run on, and the Roopairs API, which will communicate with the application by sending and receiving data, and will contain the database for the Repairs application.

Information such as database structure, feature based flowcharts, and components within the Repairs application are included in this document.

2 Problem Description

Roopairs Technologies, Inc. currently has a fully responsive web platform that helps restaurants manage their equipment, and would like to introduce a POS integration that seamlessly connects restaurants with service providers.

The SRS document can be referred to for specifics on any features or constraints that the architecture must fulfill.

3 Solution

The solution detailed within this document provides an integration with one specific POS that allows restaurants to communicate with a large network of service providers at ease. Additionally, information on pieces of equipment and repairs for restaurants will be stored for the Repairs application to analyze.

3.1 Overview

The Repairs application will be an Android OS app deployed through the Clover App Market. It will store and retrieve data such as account information, service requests, and providers through calls to the Roopairs API. All information will be displayed on the Clover system.

3.2 Components

3.2.1 Deployment Diagram

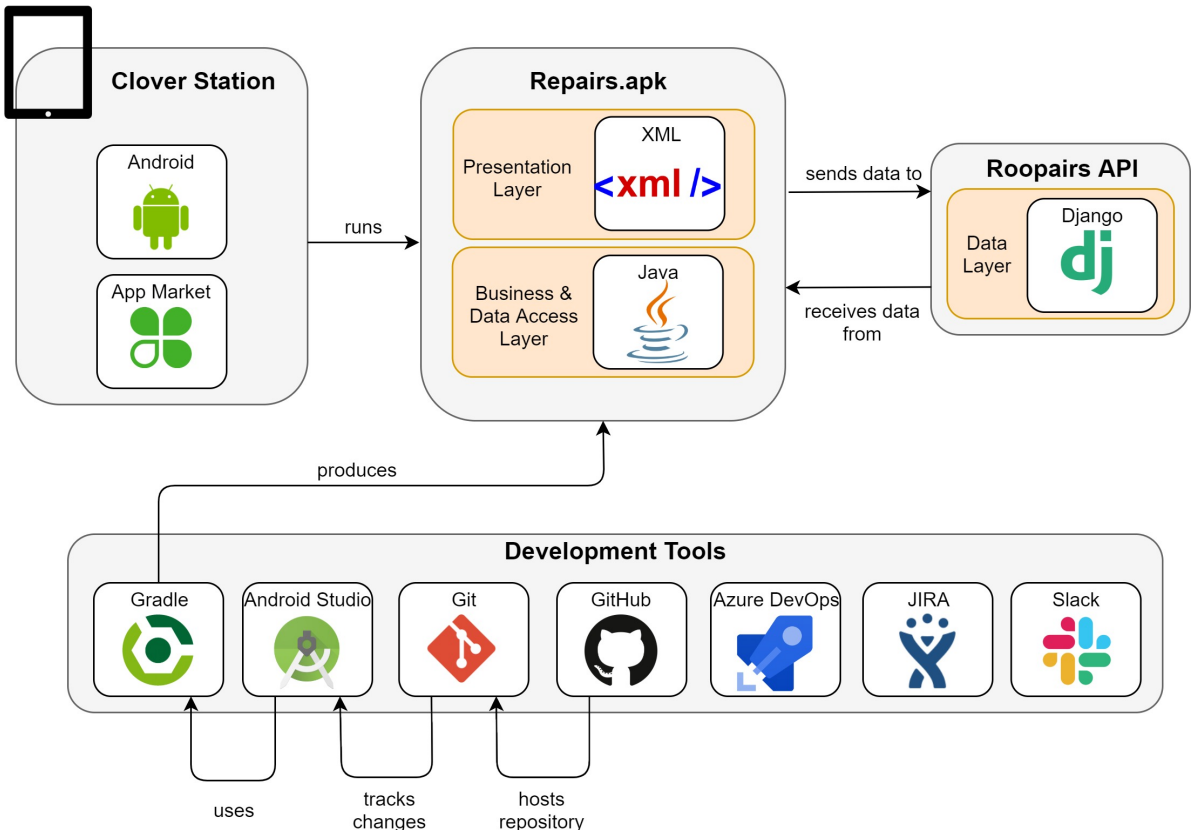

Figure 1: Deployment Diagram - Luke Reckard

The deployment diagram illustrates the fundamental components of the project. The 3 main components of the Repairs application are the Clover Station, the Repairs.apk, and the Roopairs API.

The Clover Station is an example of a point-of-sale system that will be running the Repairs Android application. The Repairs.apk is the Android application that is created through Gradle, a flexible build software. The front end of the application will be created using XML and the back end will be coded in Java. The back end will communicate with the Roopairs API to send and receive data from the database.

Repairs.apk, the Android application, will be created using Android Studio. Git will be used as the version control system, and GitHub will host the repository for the team. Azure DevOps will be used for testing and CI/CD. Jira and Slack are used for team planning and communication.

3.3 Design

3.3.1 Dialog Map

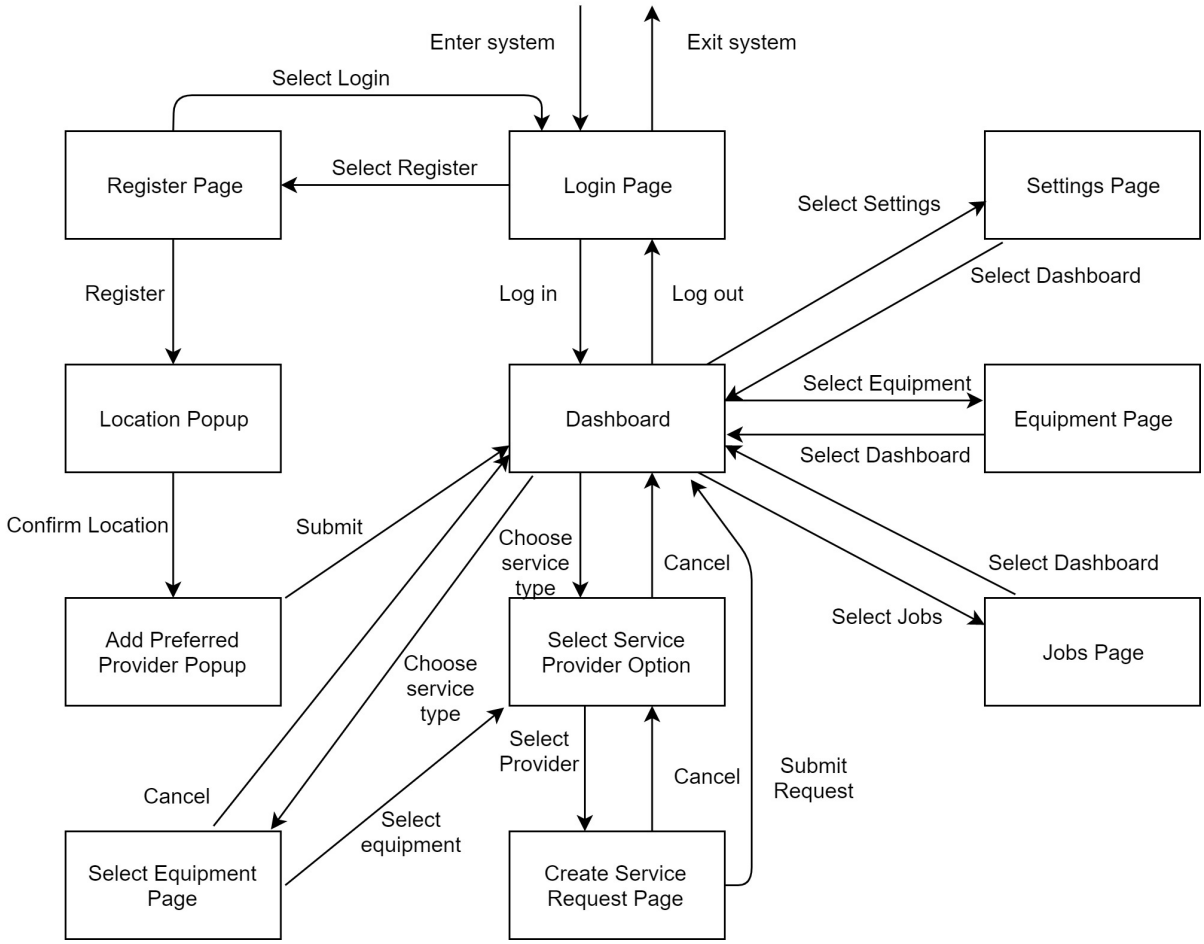

Figure 2: Dialog Map - Luke Reckard

The dialog map illustrates all of the flows through the application. The system starts at the Login Page when the application is opened. The dashboard can be accessed by logging in to a Roopairs account or going through the registration process and creating an account. From the dashboard, three main pages can be accessed: Settings, Equipment, and Jobs. Settings allows the user to edit their location or preferred providers. Equipment allows the user to track their restaurant equipment. Jobs allows the user to view pending, scheduled, in progress, and completed jobs. From the dashboard, a user can create a service request, select a piece of equipment if required, and then select a service provider option before filling out the information for the request. After submitting a request, the user is sent back to the dashboard.

3.3.2 Class Diagram

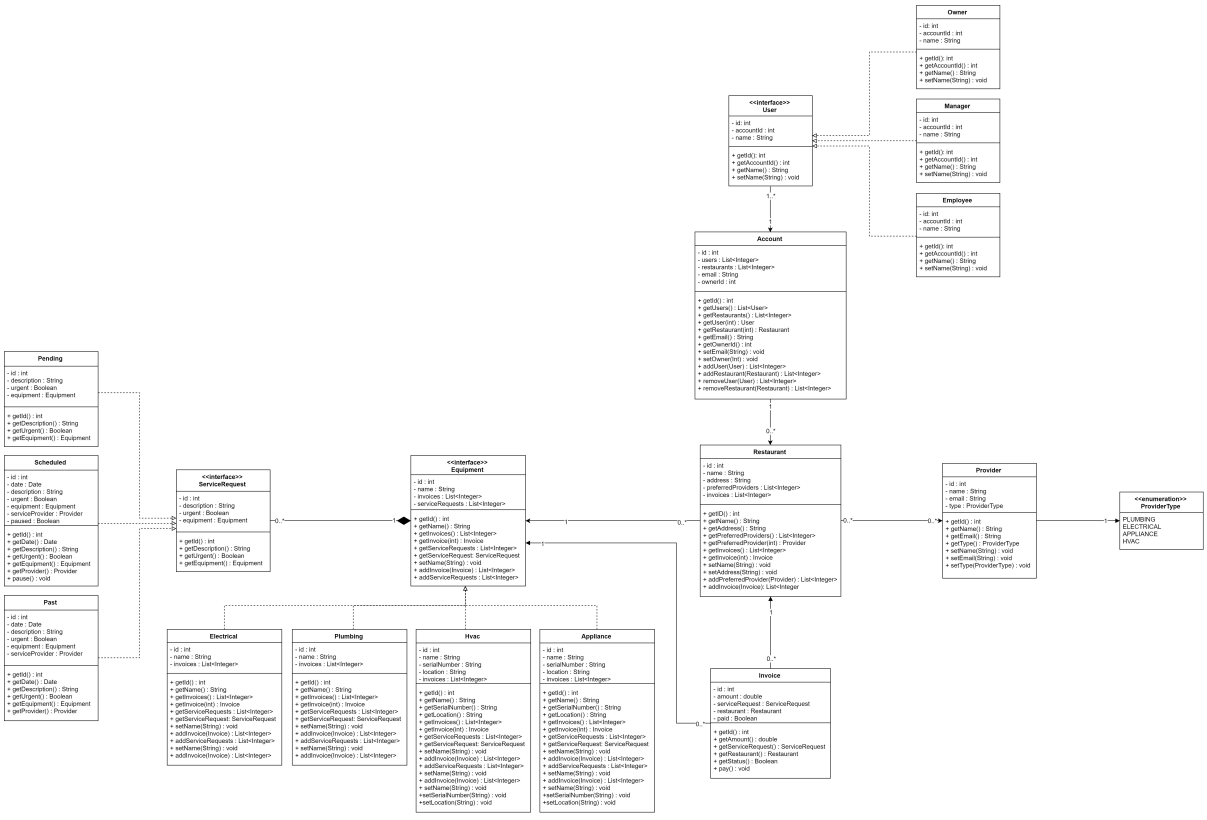

Figure 3: Class Diagram - Jessica Chang

Figure 3 above is a class diagram that represents the tentative classes in our system's model. We have a User interface, which is implemented by Owner, Manager, and Employee that will handle the different permissions for each User. User objects will be associated with an Account, and an Account will be associated with one to many Restaurant objects. Restaurant objects will have zero to many Provider objects and each Provider will be one of the types in the ProviderType enumeration. An Invoice will be associated with one Restaurant object and one Equipment object. The Equipment interface is implemented by Electrical, Plumbing, HVAC, and Appliance. HVAC and Appliance have more information stored because there will be multiple instances of them in each Restaurant. The Equipment has zero to many ServiceRequest objects and each ServiceRequest has one Equipment object. The ServiceRequest interface is implemented by Pending, Scheduled, and Past, which represent the status of a ServiceRequest.

3.3.3 Use Case Diagram

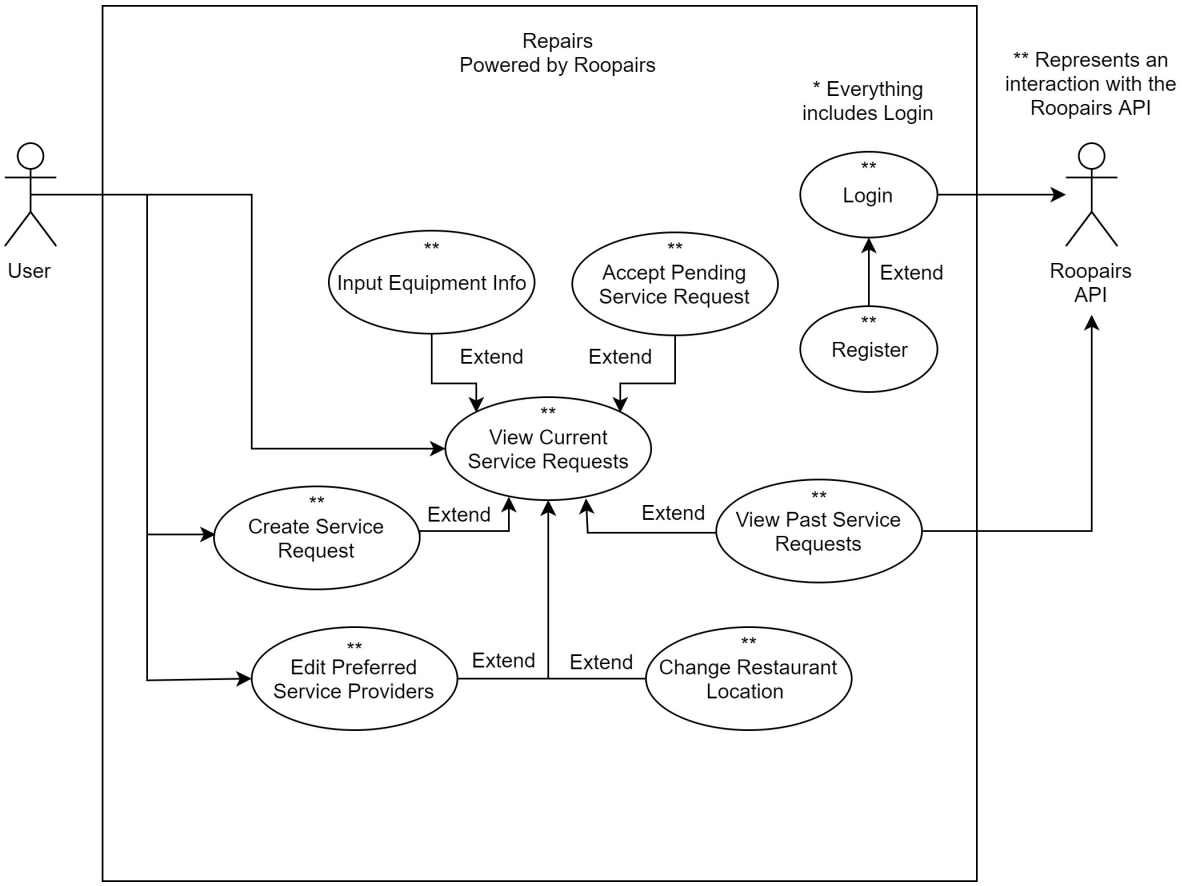

Figure 4: Use Case Diagram - Logan Lawson

The use case diagram above illustrates all the use cases of the application. The three use cases our team felt that the user would primarily open the application for were creating a service request, viewing current service requests, and editing preferred service providers. The other use cases we considered to extend off of viewing current service requests on the main dashboard. Through the dashboard you could input equipment info, accept pending service requests, view past service requests, and change restaurant location. All these use cases interact with the Roopairs API to validate, query and update data.

3.3.4 Activity Diagrams

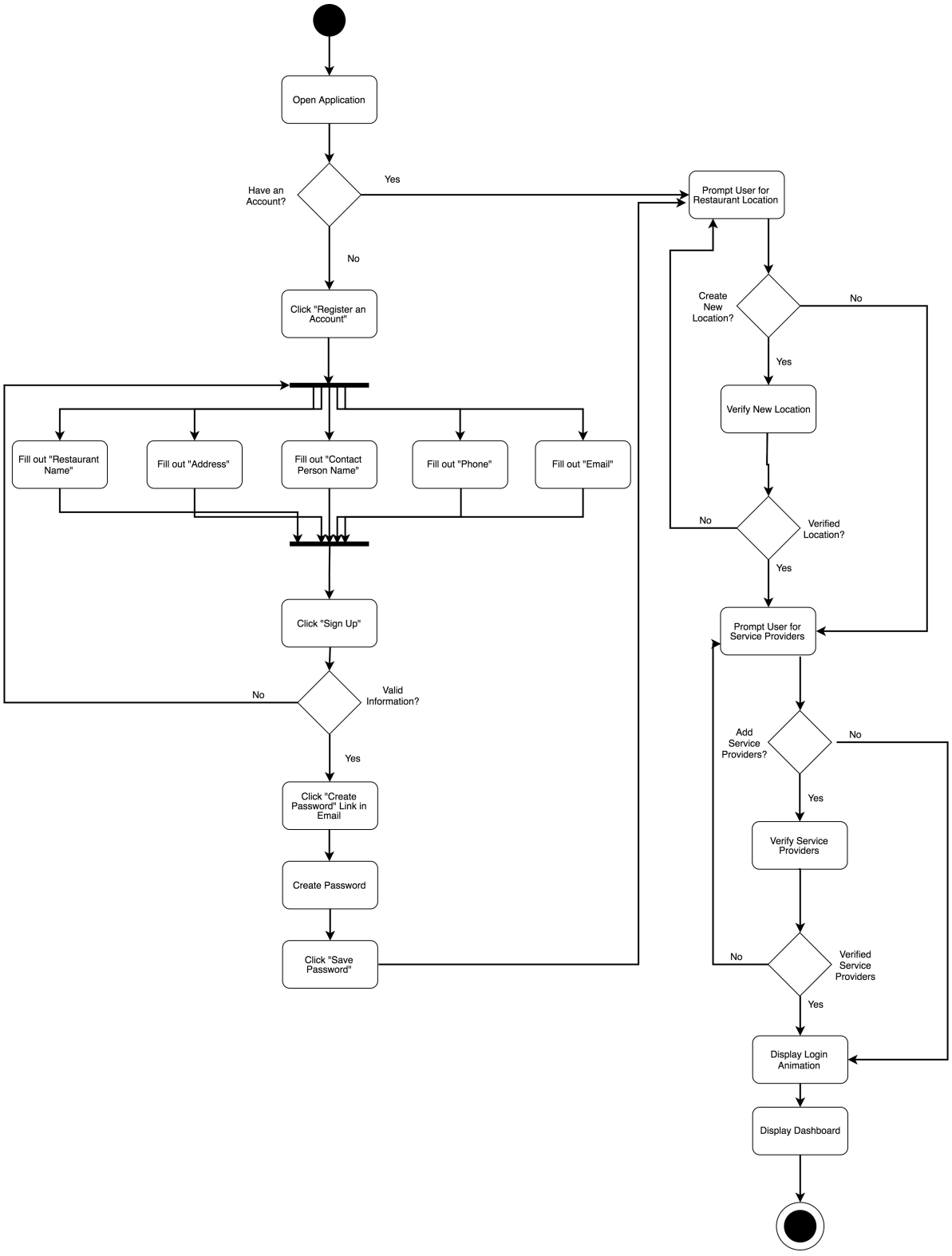

Figure 5: Activity Diagram: Login - Karla Sunjara

This diagram shows the possible flows a user can take when logging into the application. After opening the application the user will be able to register for an account or log into their existing account. If a user is registering a new account, they will be prompted to enter fields pertaining to their account information. Once a user clicks the "Sign Up" button, their information will be validated. If any information is invalid, the user will be prompted to input the information again. Otherwise, the user will be sent an account verification email with a link to creating a password.

In either case of registering for an account or logging into an existing account, the user will be prompted for a restaurant location. The user will be able to add a new restaurant location or choose an existing location. Once the restaurant location has been validated and added, the user will then be prompted to choose whether they want to add any preferred service providers. Afterwards, the user will be displayed a login animation and exited to the main dashboard screen.

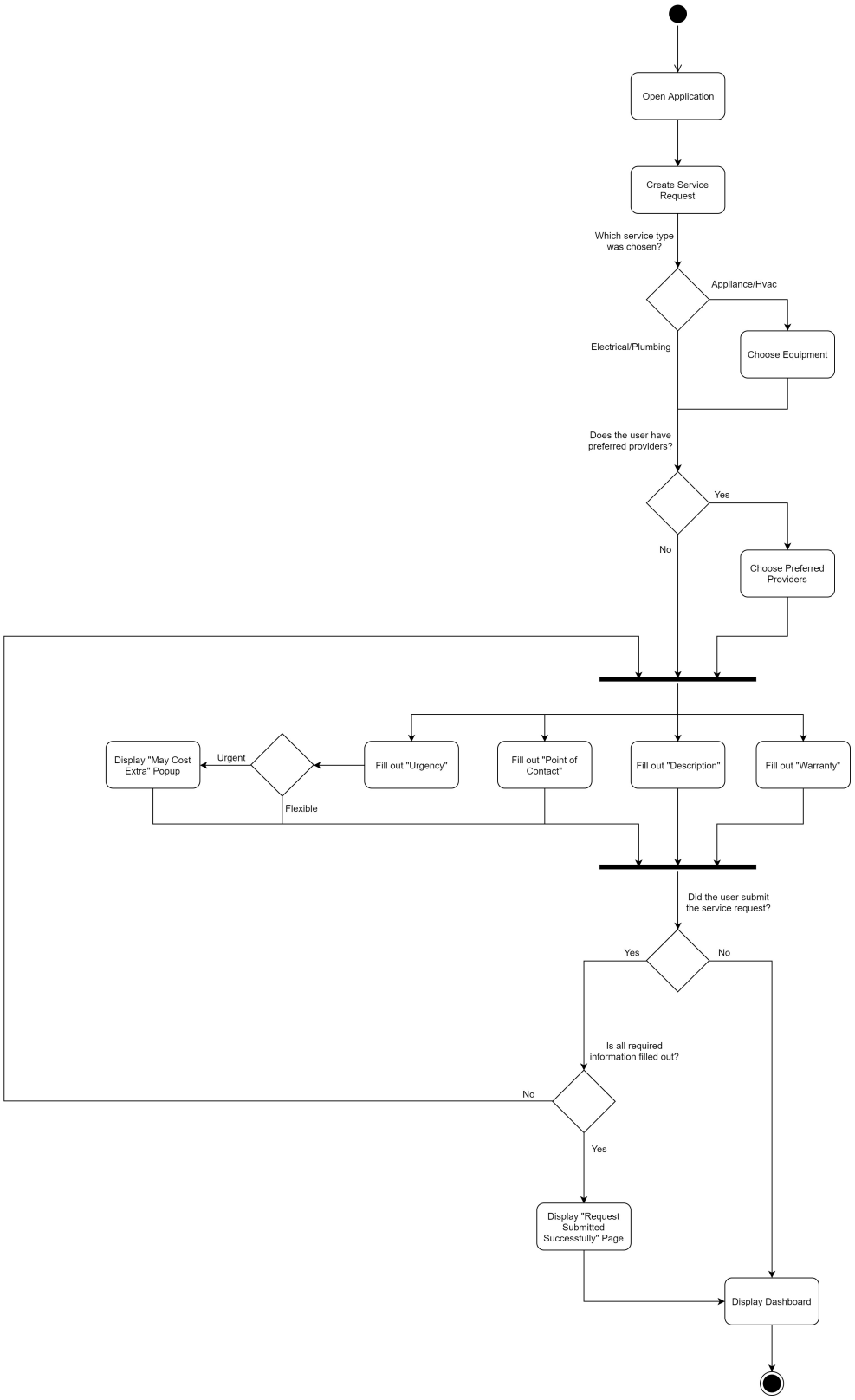

Figure 6: Activity Diagram: Create Service Request - Jessica Chang

Figure 6 illustrates the logical flow through the creation of a service request. First, the user opens the application and chooses the type of service request they want to create. If it's a service request for an appliance or heating, ventilation, and air conditioning (HVAC), the user is prompted to choose the piece of equipment that needs service. Then, if the user has preferred service providers, they are prompted to choose a provider to send the service request out to.

Otherwise, the user is prompted with an information screen, in which they fill out fields such as "Warranty", "Description", "Point of Contact", and "Urgency". Then, the user can either submit or cancel the service request. If the request is cancelled, the user is brought back to the dashboard. If the user submits the service request, the program validates that all the required information is filled out, displays to the user that the request was submitted successfully, and then returns to the dashboard.

3.3.5 Sequence Diagrams

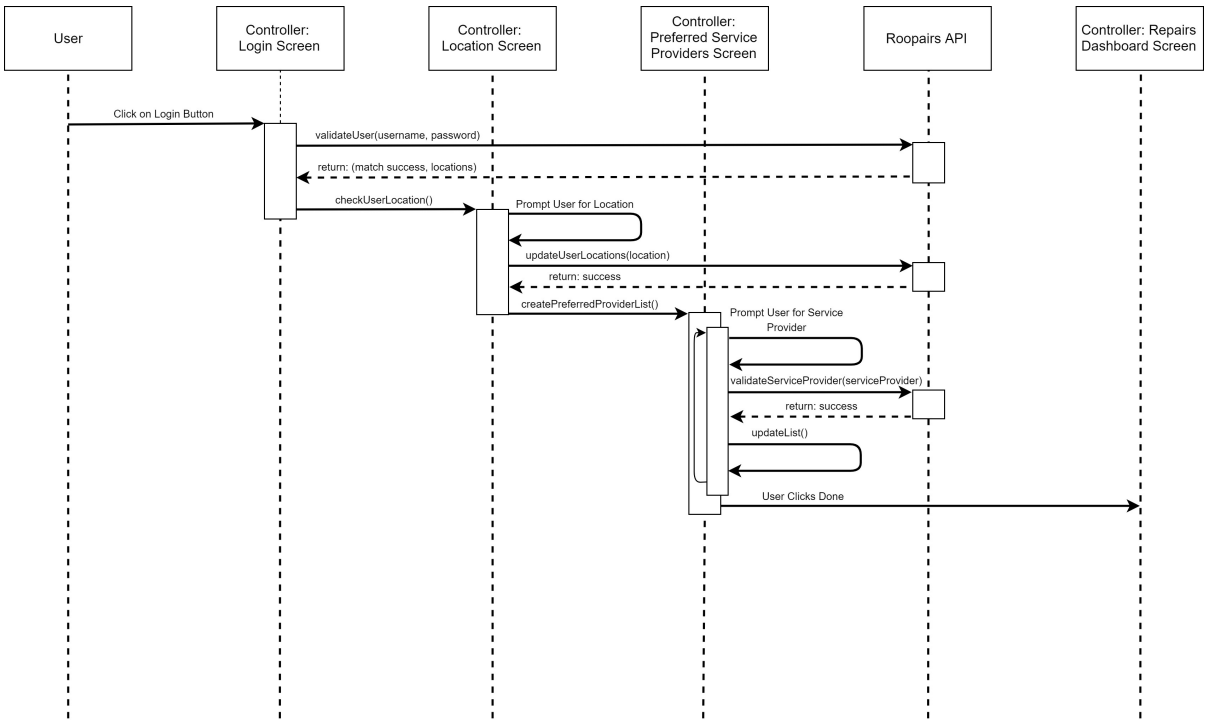

Figure 7: Sequence Diagram: Login - Logan Lawson

The sequence diagram above is built for the Login use case. The diagram starts with the user going through the login page in to the application. Upon clicking the login button, the system validates the users information with the Roopairs API and then moves to the Location screen. The system prompts the user to enter the device's restaurant location

and then updates the saved location settings. After the location check screen the system prompts the user to update or add preferred service providers. These providers are then added to the saved list when the Roopairs API validates the providers existence on the platform. Finally, the user is then exited to the main dashboard screen.

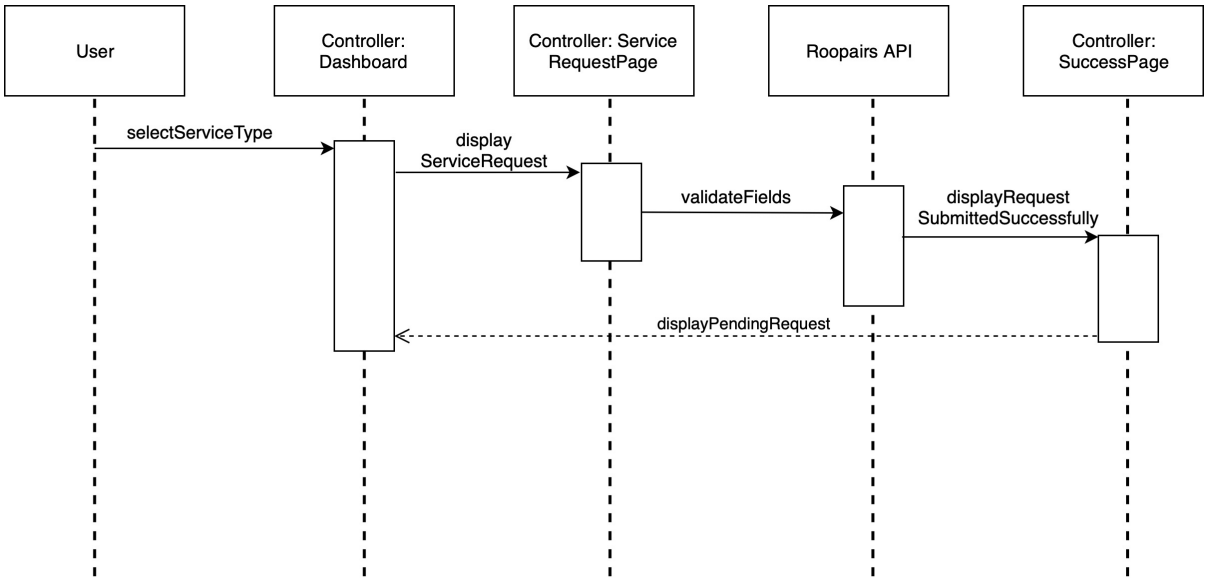

Figure 8: Sequence Diagram: Create Service Request - Karla Sunjara

The sequence diagram above is built for the Create Service Request use case. The diagram begins with the user selecting the type of service request they would like to create. Based on the service request type, the system will display to the user a service request page with fields that need to be filled in. After the user provides the necessary information and clicks submit, the system will validate the information with the Roopairs API. Once the information is approved, a success page will pop up, stating that the request was submitted successfully. Finally, the user will be exited to the main dashboard screen where their pending request will be displayed.

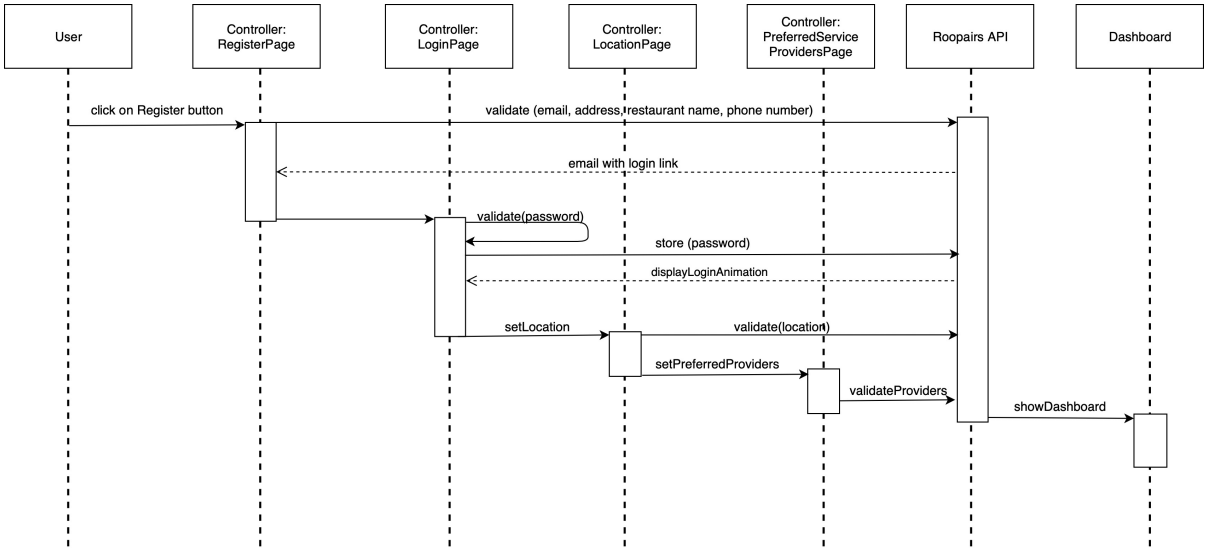

Figure 9: Sequence Diagram: Registration - Yusuf Bahadur

The sequence diagram above is built for the User Registration use case. The diagram begins with the user clicking on the "Register" button which brings them to the Register Page. The user then enters in their information. Upon clicking "Submit" the users information is validated against the Roopairs API to ensure the account does not already exist. After a successful validation, a email link is sent to the user, which routes the user back to a Login Page. Here the user creates a Password which is validated upon submission to ensure the users password contains numbers and letters. In the next step, the password is sent to the Roopairs API to store in the authentication database. The system then prompts the user to enter a restaurant location which is validated by the Roopairs API upon submission. Next, the system prompts the user for their preferred service providers which is also validated upon submission. Lastly, the system has completed its registration of the user and brings the user to the Roopairs Dashboard.

3.4 Database Diagram / Entity Relationship Diagram

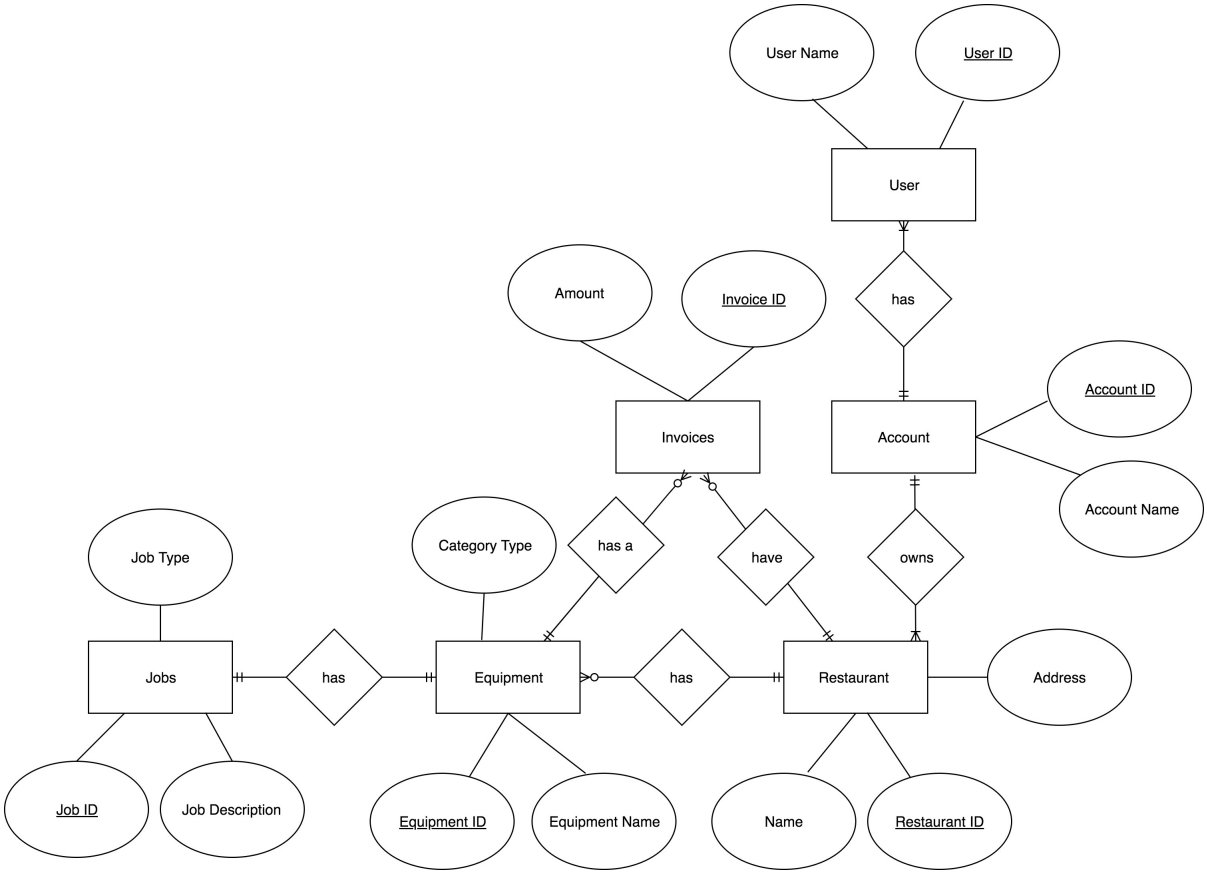

Figure 10: Entity Relation Diagram - Yusuf Bahadur

The entity-relation diagram is a high level conceptual model diagram that helps visualize the relational database. Entities are represented as square boxes and the lines between the boxes symbolize the two entities have something in common. A diamond box on top of the lines shows the type of relationship between two entities. The arrows represent cardinality to show how many instances of an entity correspond to one instance of another entity. The attributes are represented by the ovals connected to each entity and the ovals with underlined names are primary keys. .

The figure contains all the information to represent the Repairs application within Clover. The main actor is the user which interacts with the entire system through the application. Multiple users are able to be a part of one instance of an account. Each instance of an account can have multiple instances of restaurants. Each instance of a restaurant is related to zero or multiple relations of both invoices and equipment. Each instance of equipment has one instance of a job. Jobs include in pending jobs, scheduled jobs, and past jobs which is defined by the attribute job type.

4 Test

Azure DevOps will be used for continuous integration and continuous development through the use of pipelines. The rest of testing is TBD.

A Glossary

HVAC: Heating, ventilation, and air conditioning.
POS: Point-of-sale system.