SRS Specification
Retro Rabbits Claim System

Tlou Lebelo

August 23, 2019

Contents

1 Introduction

Retro Rabbits current claims process is somewhat messy and a tedious process for the all the employees working at Retro Rabbit. At the moment, the employees are required to keep their slips from the past month (up to three months) which they are then allowed to claim for on the first Friday of the month. This administration process is difficult to manage and if an employee loses a slip, they aren't able to claim for it. In addition to this, it is difficult to make accurate claims for fuel used driving to clients (in the event that the employee needs to take a detour that hasn't been accounted for on the map software being used).

The goal of this project is to create an android application that will improve this claims process, making it easier and more efficient for all the employees at Retro Rabbit. There are two main groups of claims, that being fuel claims and other(eg. food, data) claims. The android application therefore needs to make use of image analysis software which allows an employee to take a picture of the slip, this will then take the necessary data from it (date, time, amount, UID) and store the claim for the employee. In the event of an employee claiming for fuel, the employee is required to take a picture before they leave the office or their home of their cars odometer, then once they arrive at the client be reminded to take a picture of their odometer. The application then preforms the required calculation and stores the data for the employee. On the first Friday of the month, the application will tell the employee if there are any pending claims and remind them to send their claims through.

On completion of this project Retro Rabbit should have an application that meets their needs and provides a much needed streamlined and efficient claims process.

2 Architectural Requirements

Peer-to-peer architecture
The system is divided into two main sub components which is the back-end and front-end. The two sub-components are arranged in a peer-to-peer architecture for continuous synchronization of data.

2.1 Front-End

Micro-services Architecture
The architectural style, structures an application as a collection of services that are:

- Highly maintainable and testable.
- Loosely coupled.
- Independently deployable.
- Organized around business capabilities.

This complements our effort to try and build a system deployable on different operating system. That is android and iOS.

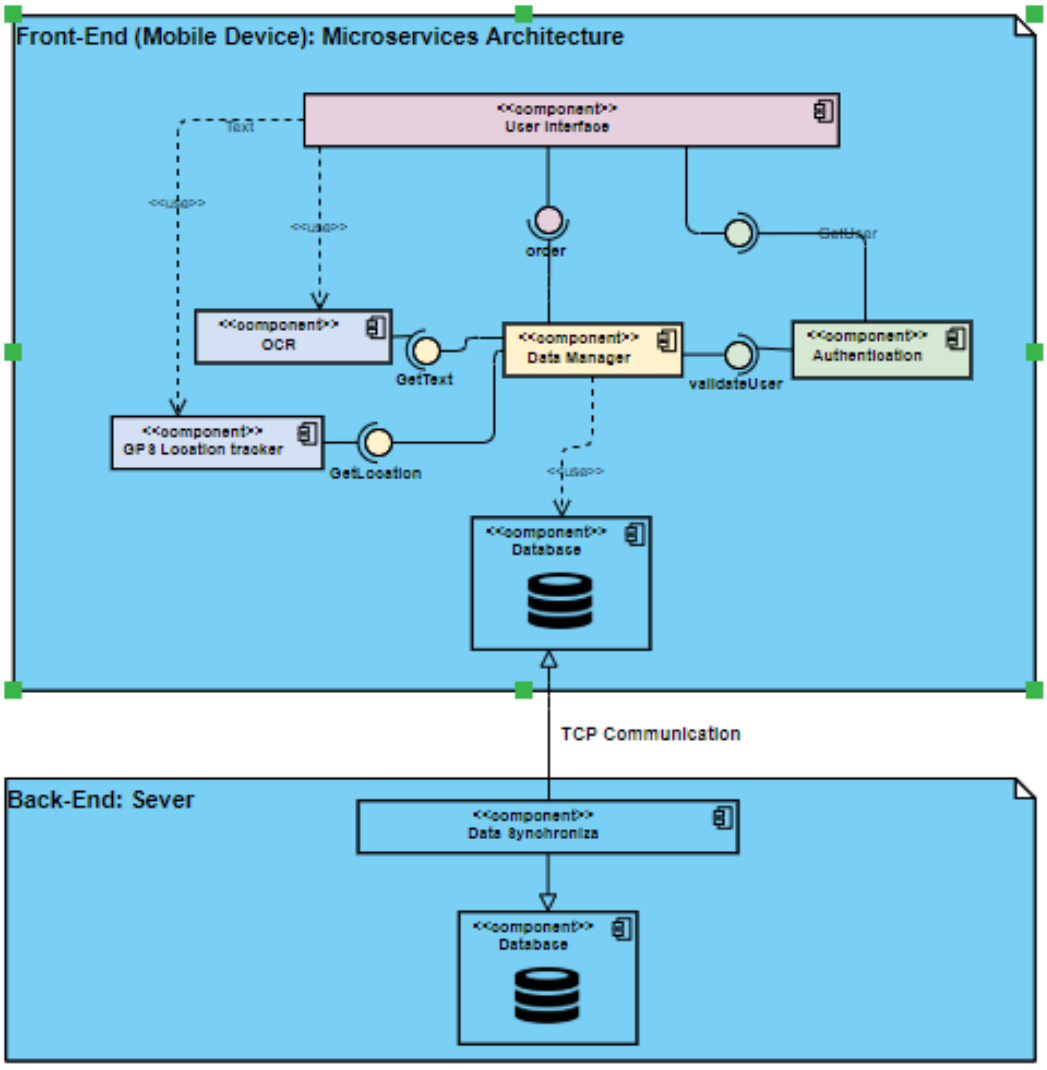
2.2 Back-End

The back-End serves as a global data infrastructure which allow continuesly synchronization using a secure virtual circuit (VC) communication. That is ...

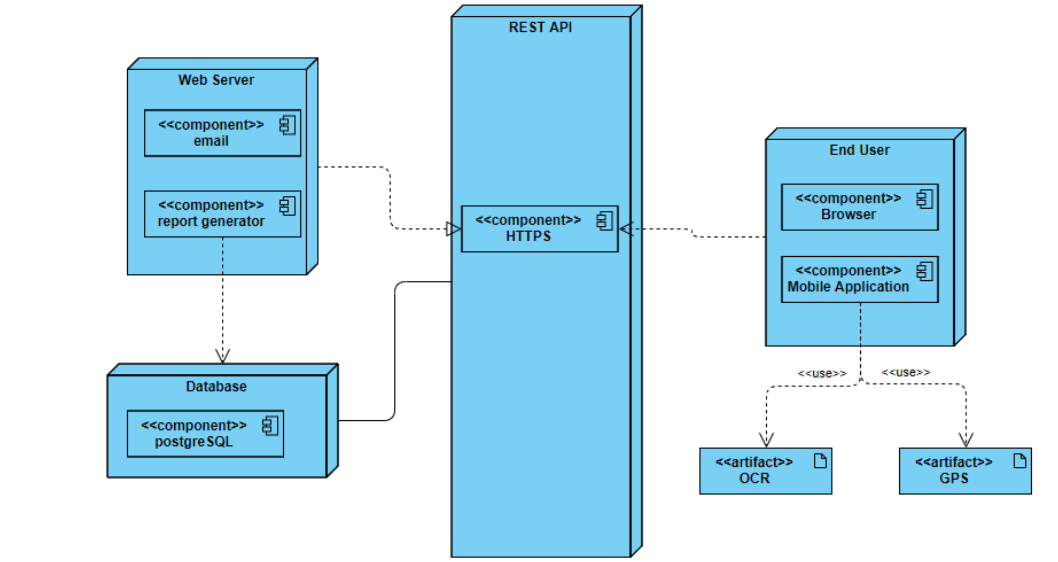
- Transmission Control Protocol (TCP).
- Stream Control Transmission Protocol (SCTP).

VC means of transporting data over a packet switched computer network in such a way that it appears as though there is a dedicated physical layer link between the source and destination end systems of this data

2.3 Architectural Diagram

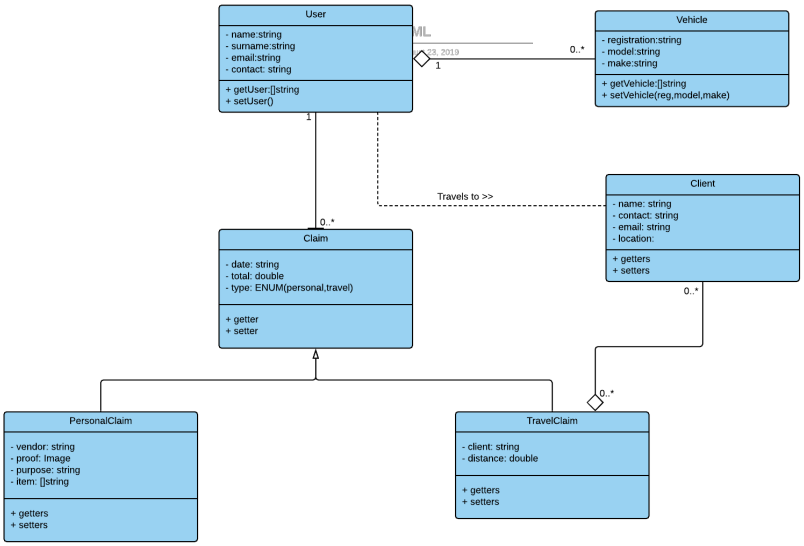


2.4 Deployment Diagram



3 Domain Model

UML.png



4 User characteristics

No technical skills or high level education required from the users. The application will prioritize usability to allow users to navigate through the application with ease.

The users of the system can broadly be categorized as follows:

- **Employee**
  These are the primary users of the system.
  The user must have an android smart-phone on which to download, install and use the application.
  The user will record their monthly expenditure which the employer will later on compensate. This will include:
  - Taking pictures of the cars' odometer before and after trips.
  - Taking pictures of receipts as proof of payment.
- **Administrator**
  This is the secondary user of the system.
  This user will use the system to receive and view claims from employees through the system.
  PDF statements containing claim information will be sent to the user every month.

5 Functional Requirements

5.1 Use cases

Figure 1: Authentication UC1

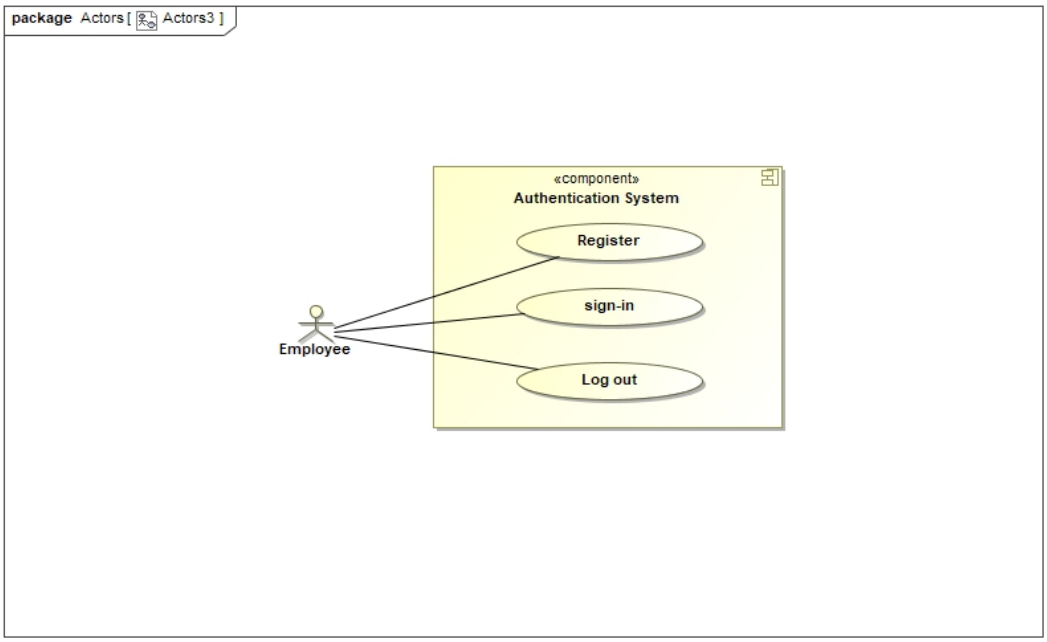


Figure 2: Other Claims UC2

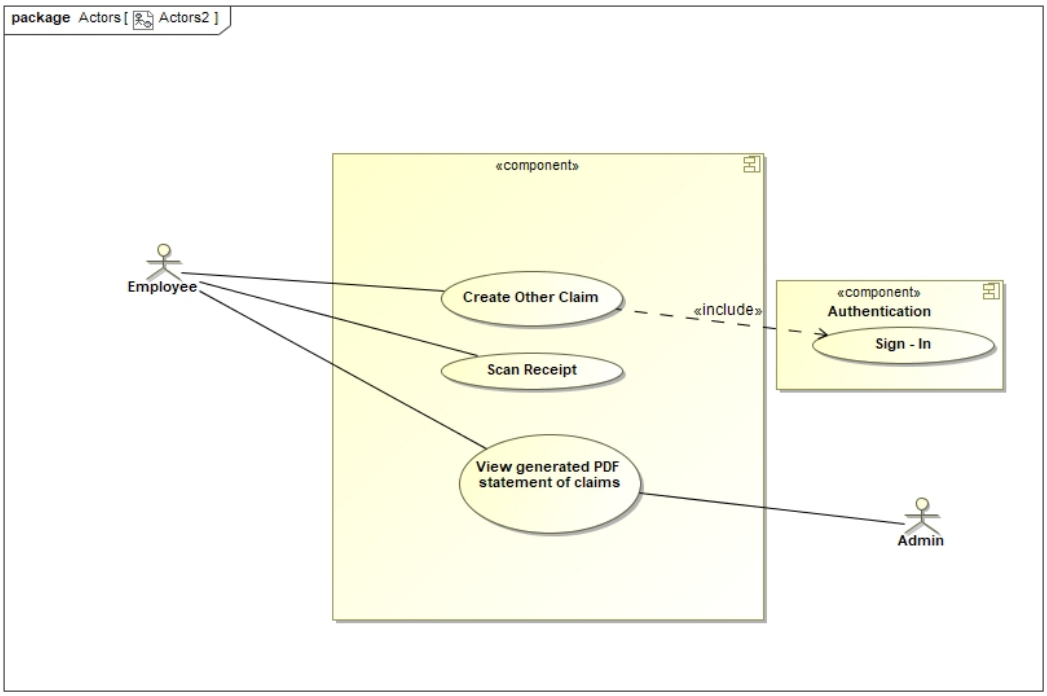


Figure 3: Unprocessed Claims UC3

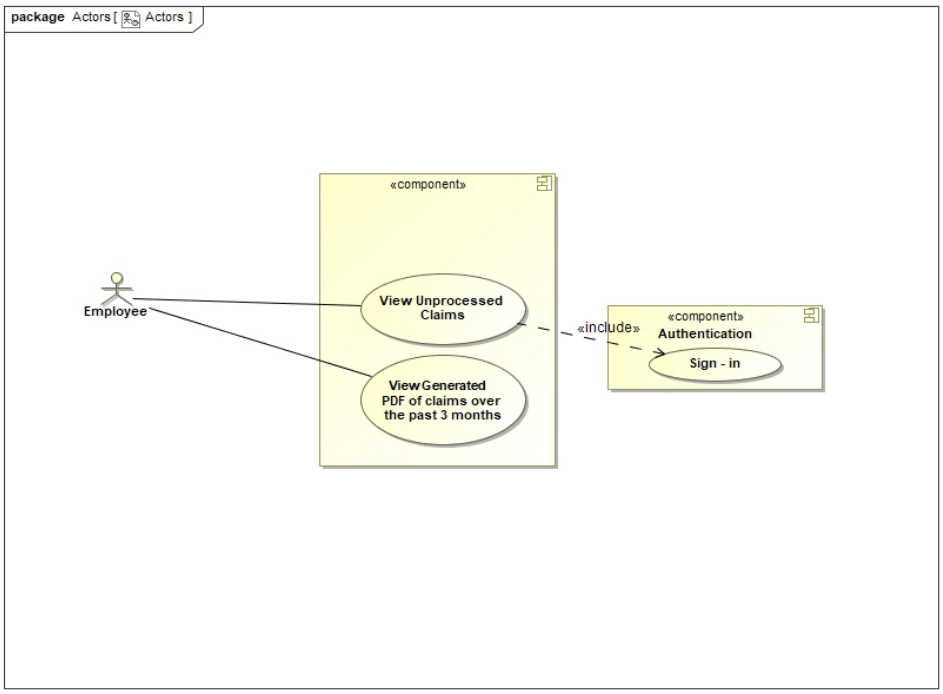
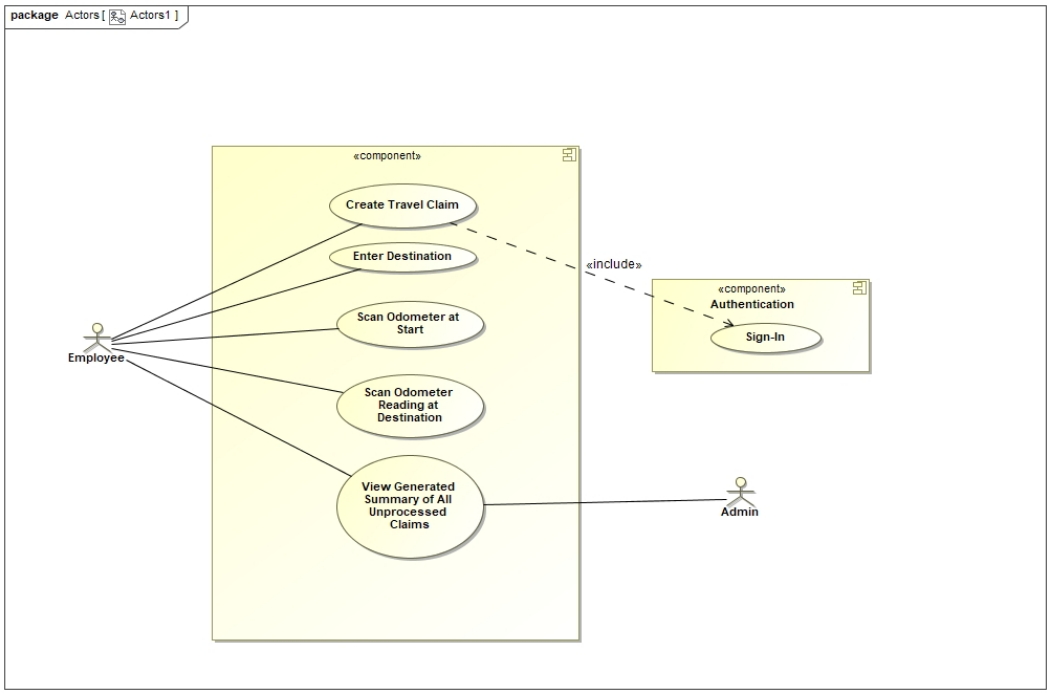


Figure 4: Travel Claims UC4



5.2 Requirements

Our Identified Subsystems are: Application, Database, GPS, Image Recognition, Log - in and Authentication and Employee Accounts

5.2.1 Application

R1.1 Provide a log in functionality for the employee.

R1.2 Allow the employee to select the type of claim they would like to log.

R1.3 Allow the employee to make use of their camera to scan the odometer or slip.

R1.4 Allow the employee to view all unclaimed processes as well as the estimated cash back that can be expected(claim form).

R1.5 Remind the employee via a push notification to log the odometer once they have arrived at their destination.

R1.6 On the first Friday of the month remind the employee to log their claims for the month.

5.2.2 Database

R1.7 Provide a NoSQL solution for data storage(compressed images and receipt/odometer data)

R1.8 CRUD functionality for the data captured.

R1.9 Statement generation using stored data.

R1.10 Provide a suitable record structure for each claim.

R1.10.1 Store time/date

R1.10.2 ID

R1.10.3 The value(cost/ distance)

5.2.3 GPS

R1.11 Get the employees current location from Google maps.

R1.12 Calculates the the distance using the cars odometer between start and destination points.

R1.13 Determine the shortest route for employee to take.

R1.14 Compare the distance covered from the odometer reading to the shortest Google map route.

5.2.4 Image Recognition

R1.15 Provide functionality to access the phone camera and take pictures.

R1.16 Scan the images and provide a digital reading of:

1. The odometers current value in kilometers.
2. Date, time, unique ID an amount on the purchase slip.

5.2.5 Log-in and Authentication

R1.17 The system should allow only valid employees to access the system.

R1.18 The system should allow an employee to log in using some n-factor based authentication.

R1.19 The system should allow an employee to log out.

R1.20 The system should allow a new employee to be registered.

5.2.6 Employee Accounts

R1.21 The system should allow employees to view details on the claim form generated.

R1.22 The system should allow an employee to edit details on the claim form generated.

R1.23 The system should allow employees to view their unprocessed claims.

6 Non Functional Requirements

6.1 Quality Attributes

Performance:

QR1.1 The system should be able to make timely(should take 2-3 minutes) claim forms that will be sent to the correct administrative powers handling the claims.

QR1.2 The image to data recognition software used by the system should take less than a minute to process the information captured.

QR1.3 Unprocessed claims within the last three months can be requested by the user at any point, this process should not take more than a minute.

QR1.4 The application should provide the user with a timely(less than a minute) notification once a set destination has been reached in order to remind the user to take an odometer reading.

QR1.5 The system should allow for multiple concurrent users to be logged in and be able to process multiple requests to the server at the same time.

Scalability

QR2.1 The database solution will have to be highly scalable to allow for the potential influx of data therefore a NoSQL solution seems necessary.

QR2.2 At least 100 concurrent users must be able to use the system at any given time.

Security:

QR3.1 The system front-end will make use of at least 1 factor authentication system however the user will have a choice of verification techniques(using a password or authenticating through google).

QR3.2 All local passwords used will be encrypted in a separate secure data storage.

QR3.3 The data captured form the front-end will be encrypted and the encrypted data will be stored in the back-end.

QR3.4 Extra security functionality will be added to the database used in order to prevent various data attacks.

QR3.4 The system will be designed to prevent as many data leaks as possible.

Intergratability:

QR4.1 All the defined subsystem must be able to communicate with each other using the API framework we will develop.

QR4.2 External library functions that will be used for the image to data recognition must work within our system.

QR4.3 The system must also be able to make use of Google services for both GPS services and potential authentication.

QR4.4 The system modules must require very little changes in order to work with new components.

Usability:

QR5.1 The system relies heavily on user interaction it is therefore imperative that the front-end sub-systems that the user interacts with is well designed

QR5.2 The sub-systems that will be in constant interaction with the user must be designed so that the controls and functionalities are easy and intuitive to use.

QR5.3 The system should allow for analytics collecting which can determine whether or not users were able to use the system efficiently.

6.2 Constraints

1. A user will only be allowed to login on one device per session.
2. The OCR must allow for capture of any text scanned.
3. The user will have a limited amount of tries to login after which they will be emailed.
4. The claims will only be sent to the administrative entities of the company on the 1st Friday of the month
5. The GPS tracking will only track the user once a route has been enabled.
6. The GPS system must track a user within 50 meters of their actual location
7. The database solution must allow the storage of claims for the last 3 months

6.3 Technology Requirement

1. The application must run on android devices
2. The OCR subsystem makes use of the free Firebase ML AI for image recognition
3. The GPS subsystem makes use of the free geolocation library from React Native for user tracking as well as determining destination coordinates.
4. A NoSQL(MongoDB) solution is required for the database
5. Google services for authentication is required

7 Traceability Matrix

	Client Accounts	Application	Database	Image Recognition	Authentication	GPS
R1.1					X	
R1.2		X				
R1.3		X				
R1.4	X	X				
R1.5		X				X
R1.6		X			X	
R1.7			X			
R1.8			X			
R1.9		X		X		
R1.10			X			
R1.11						X
R1.12		X		X		
R1.13						X
R1.14				X		X
R1.15		X				
R1.16				X		
R1.17					X	
R1.18					X	
R1.19					X	
R1.20					X	
R1.21	X	X				
R1.22	X	X				
R1.23	X	X				
QR1.1		X				
QR1.2				X		
QR1.3	X	X	X			
QR1.4		X				X
QR1.5	X	X				
QR2.1		X				
QR3.1					X	
QR3.2			X		X	
QR3.3			X	X		
QR3.4			X			
QR4.1		X				
QR4.2		X				
QR4.3		X				
QR5.1		X				
QR5.2		X				