Paprika Financial App Software Design Document

Team Paprika
Financial App
Andrew Feltham, Russell Morgan
March 31, 2018
Version 0.4

TABLE CONTENTS

Revision Sheet

Revision Number	Date	Brief summary of changes
0.4	31-03-2018	Updated diagrams to current design.
0.3	09-02-2018	Updated diagram and design based on agile development
0.2	19-02-2018	Project specific design information added
0.1	15-02-2018	Baseline draft document

1 Introduction

1.1 Purpose

The purpose of this document is to identify and describe the main structural components of the Paprika Financial App, and the relationships between these components. This document is intended to be used by the client and developers of the app. This document will serve as an overview of the application architecture for the client, and as a design reference for the developers of the app. This document covers the entire scope of the application.

1.2 Scope

The Paprika Financial App project consists of a single Ionic codebase, that is transpiled into both iOS and Android compatible code. The design of the project methodology is AGILE, using 2-week sprints and daily scrums.
The app will allow the user to import .csv files that have been exported from a bank's software/website, and graph the data in these files. The software will guide the user through basic financial and data representation concepts to maximize comprehension of the data. The software will not facilitate transactions involving real money, or allow access to any bank accounts.
The product must indirectly interact with the user's banking software or website by way of importing .csv files exported from them. This document may evolve to include the apps interactions with other file types or data input methods.

1.3 Definitions, Acronyms, and Abbreviations

app: an application designed to run on mobile devices.

.csv - A file type consisting of records in comma separated value format.

1.4 References

2 Design Overview

2.1 Background Information

The client for this project is the professor, Youry Khmelevsky. The developers of this app are Team Paprika.

The assumptions made in this design are as follows: the user has a way to convert their desired financial information into valid .csv file format, and the user has a device that supports the Ionic/Cordova framework.

The business processes that the system will model are tracking and organizing financial transactions, creating and displaying graphs using transaction data, scheduling bill reminders, and providing extensive tutorials for all of these processes.

2.2 Alternatives

A mobile friendly website design was considered, but the client vetoed this design choice in favour of a native app.

Coding the Android and iOS versions separately using Java and Swift respectively was considered, but was deemed to be too ambitious given the limited timetable.

3 User Characteristics

Potential users for this software include anyone who makes frequent use of a bank account or similar financial service capable of tracking transaction history. The integrated tutorial design will allow the system to easily meet the requirements of users with limited prior knowledge concerning finances and data analysis.

4 Requirements and Constraints

4.1 Performance Requirements

There will be a testing suite to test the core functionalities of the software (proper data importing, exporting, presentation, etc.) Performance on both iOS and Android devices will be tested separately to ensure performance requirements are met on both platforms.

4.2 Security Requirements

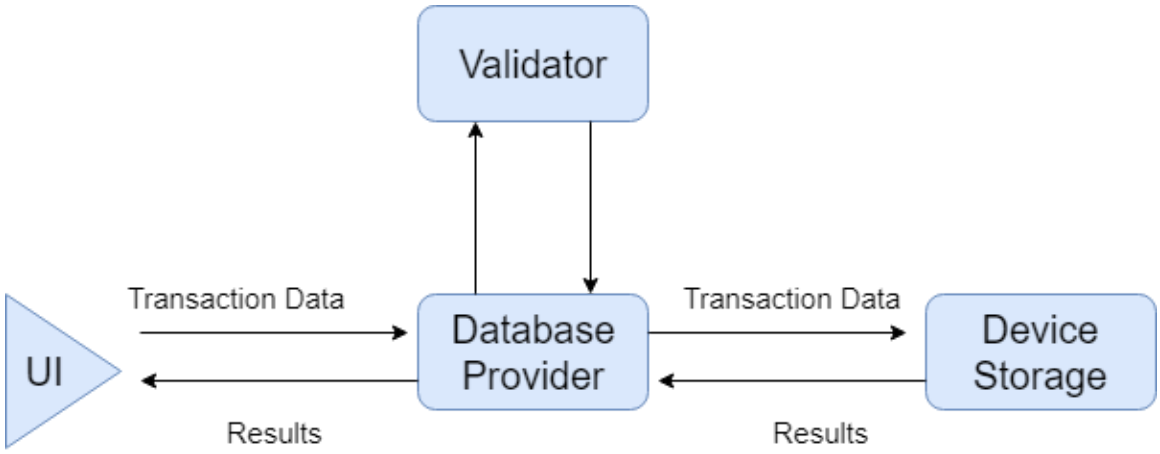
Security requirements will be met by encrypting the client side database, and implementing safeguards to ensure that the database is always re-encrypted when the user has finished accessing it. The user's data will also be protected by the requirement to log in to a protected account to access their information.

4.3 Design Constraints

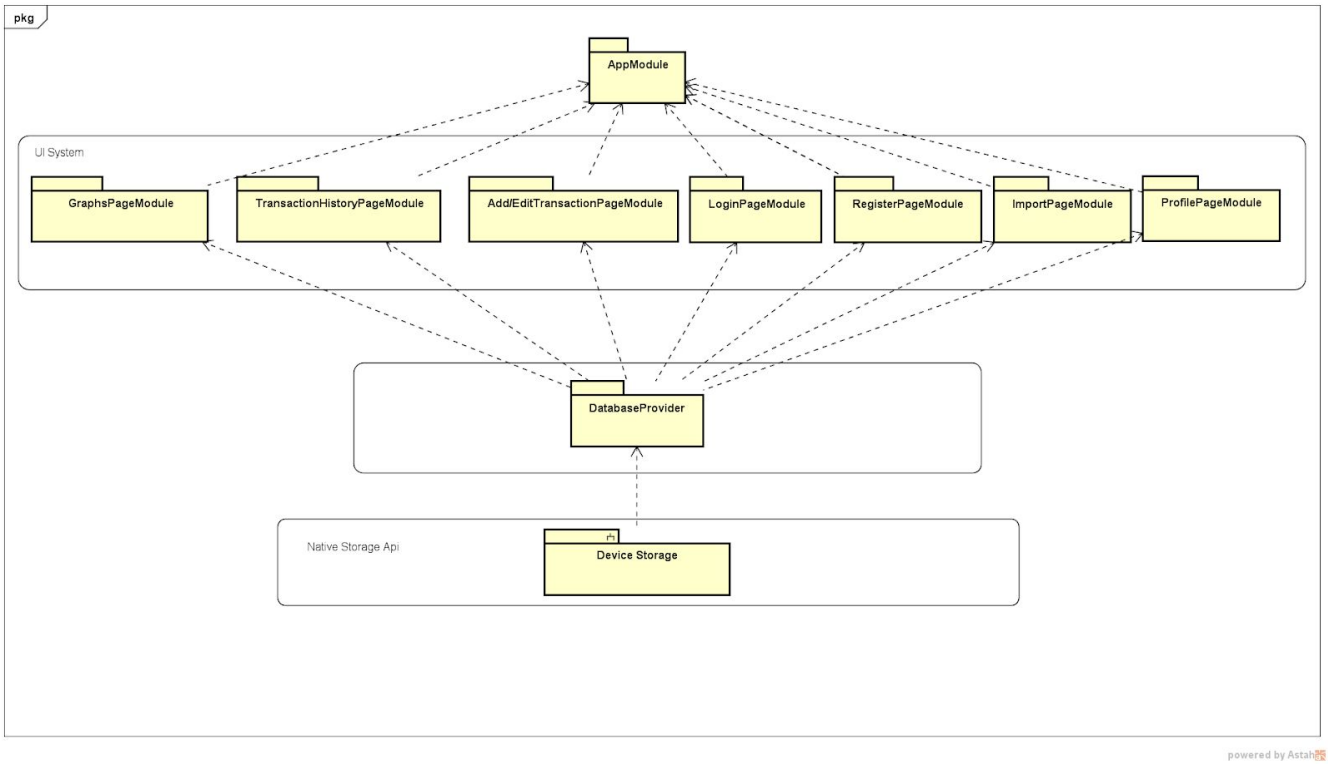
The project is under fairly strict time constraints (must be completed and deployed in April 2018,) and very strict budget constraints. The design is also constrained by the developer's lack of ability to carry out real financial transactions or access private bank

accounts, so these behaviours must be simulated by the system, and their effects on the accounts must be entered manually into the app.

5 System Architecture



5.1 Component Diagram



6 Detailed Design

ImportPage Component:

This component handles importing .csv files into the app, parsing the data, and storing it in the Database. This system interacts with the Database, User Account and Bank Account components, as the data file is associated with a single User and Bank Account.

TransactionHistory Component:

This component allows the user to load data from the database and modify it. It interacts with the Data Provider, as the data must be associated with one or many Bank Accounts, all of which must be associated to a single User Account.

GraphPage Component:

This component allows the user to view and export data in the form of graphs and charts. It interacts with the Data Provider component.

Login and Register Component:

This system handles User Account information (username, password, etc). This component interacts with the DatabaseProvider.

Bank Account Component:

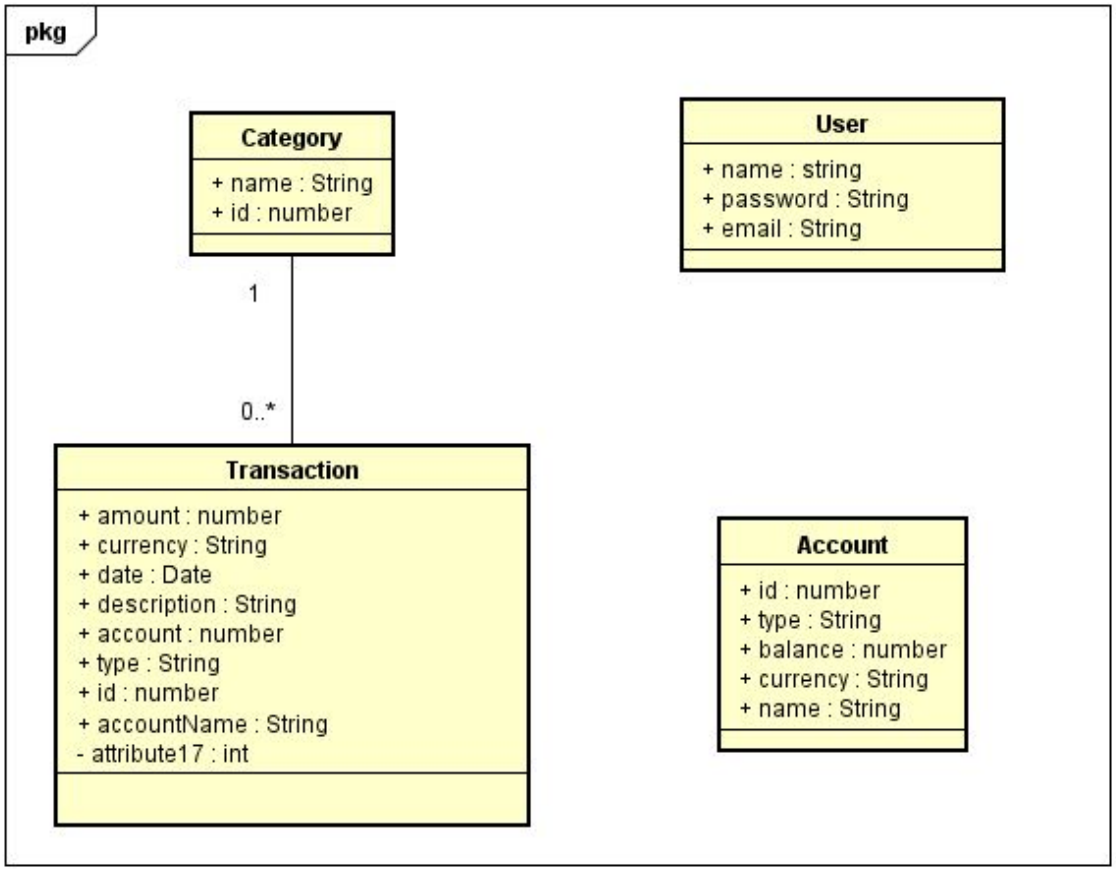
This component tracks which transactions are associated with which Bank Account, and allows users to manually enter transactions into the app. A Bank Account is associated with a single User, and the Bank Account component must also interact with the Database component.

Database Component:

This component updates or inserts data into the database by interacting with the Data Import system. It also interacts with the User Account component to query the database to access data associated with the relevant User. This component also re-encrypts the database when the User logs out.

7 Data Architecture

A client side encrypted relational database will be used to store user information, bank account information, and transaction histories. The data schema is shown below.



7.1 Data Analysis

The app will use basic data analysis tools (averaging, regression, normalization) to present the financial data in the form of simple graphs and charts.

7.2 Output Specifications

The app will support exporting data back into .csv files, and will also support exporting graphs in image format (file format not yet specified).

7.3 Logical Database Model

See ER diagram above.

7.4 Data Conversion

Data will be migrated into the app as .csv files from an external filesystem, after the .csv files are validated. The data will then be stored in an encrypted client side database. After

the user has modified their transaction history in the app, it can then be exported back into .csv format, with the changes made being properly reflected in the exported file.

8 Interface Requirements

8.1 Required Interfaces

The app is written using Ionic framework, and will therefore utilize it's software interface to interact with the host operating system. This includes methods to access the local databases and notification systems on the device. The design leaves most interface issues to be handled by the Ionic framework.

8.2 External System Dependencies

Support for different devices/software is limited to operating systems which support Cordova, and the app only supports tracking transaction histories for accounts which can be converted to .csv format.

9 User Interface

The software is a mobile application for iOS and Android, so the primary user interface will be a touchscreen compatible GUI in accordance with the product family style guides for iOS and Android. The primary reference used for GUI standards will be Google Material design. Most data is outputted to the screen, but the app will support exporting data to an external file system.

Refer to the Software Users Manual for further interface documentation.

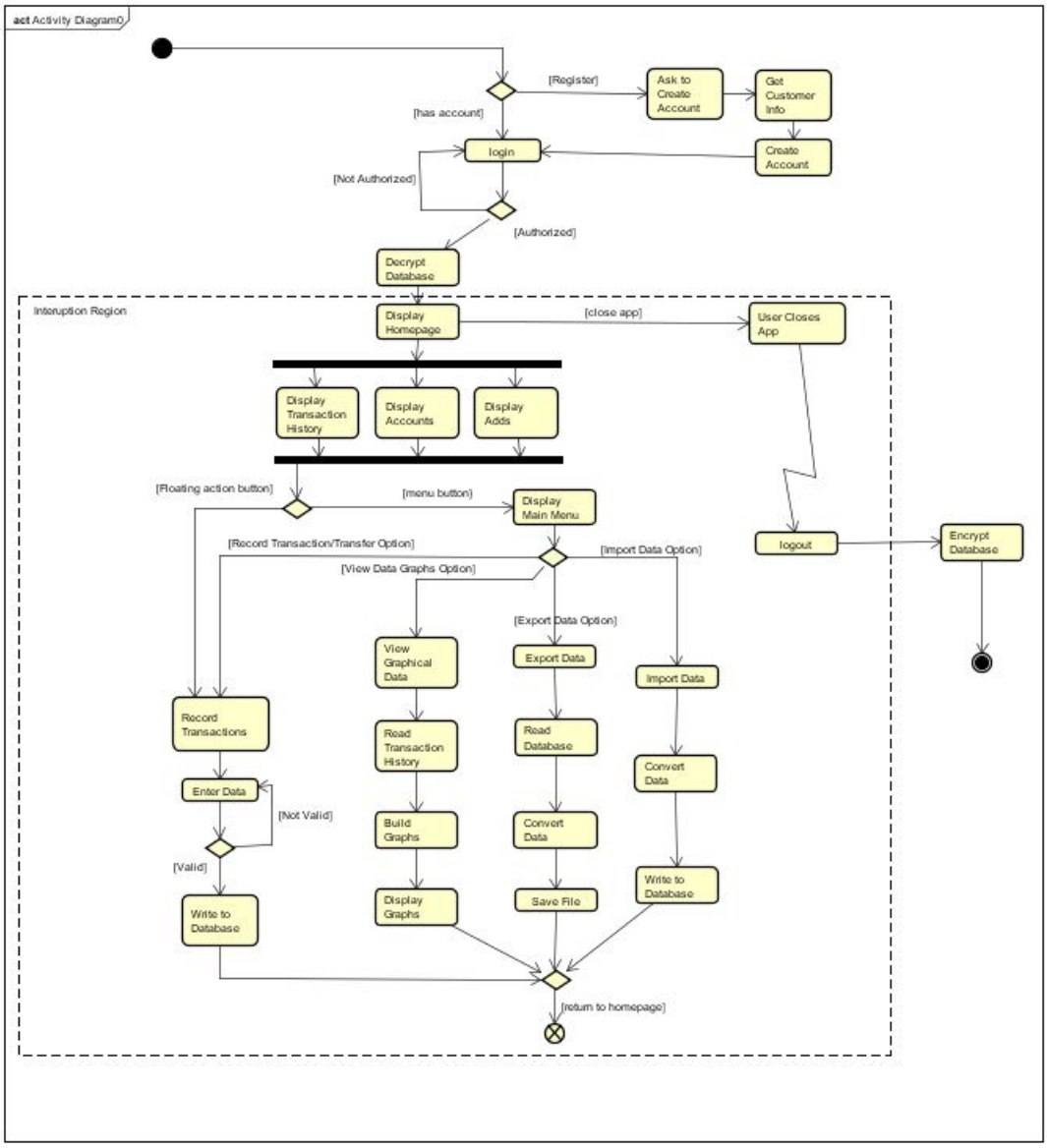
10 Non-Functional Requirements

The Paprika Financial App will be available in English only. There is no licensing for any content in the application.

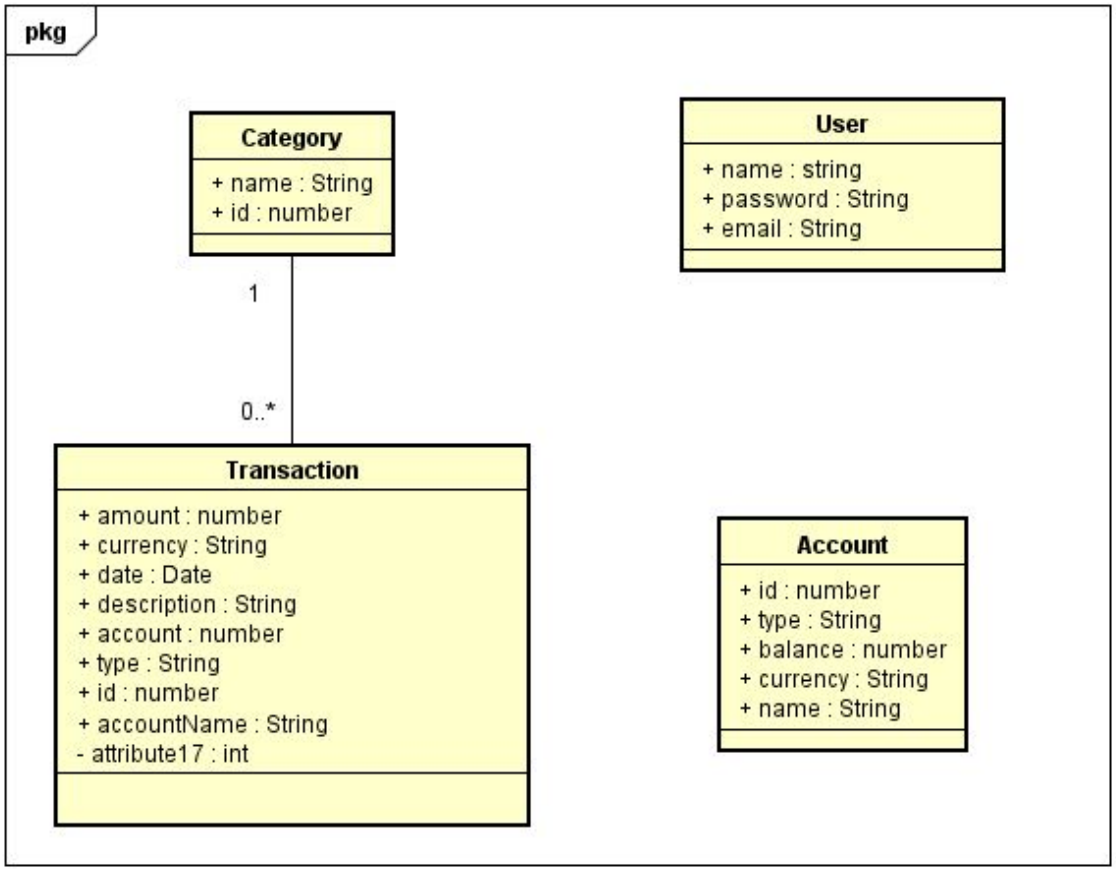
11 Software Design Diagrams

11.1 Activity Diagram

The activity diagram shows the process that the user will go through with each function of the app. Guards are displayed as follows: [condition].



11.2 Class Diagram



The Class diagram shows what objects are being used in the software.
As shown in the class diagram, most objects are stored in the database separately, only "category" is an attribute of transaction, as each category can be assigned to zero or more transactions.

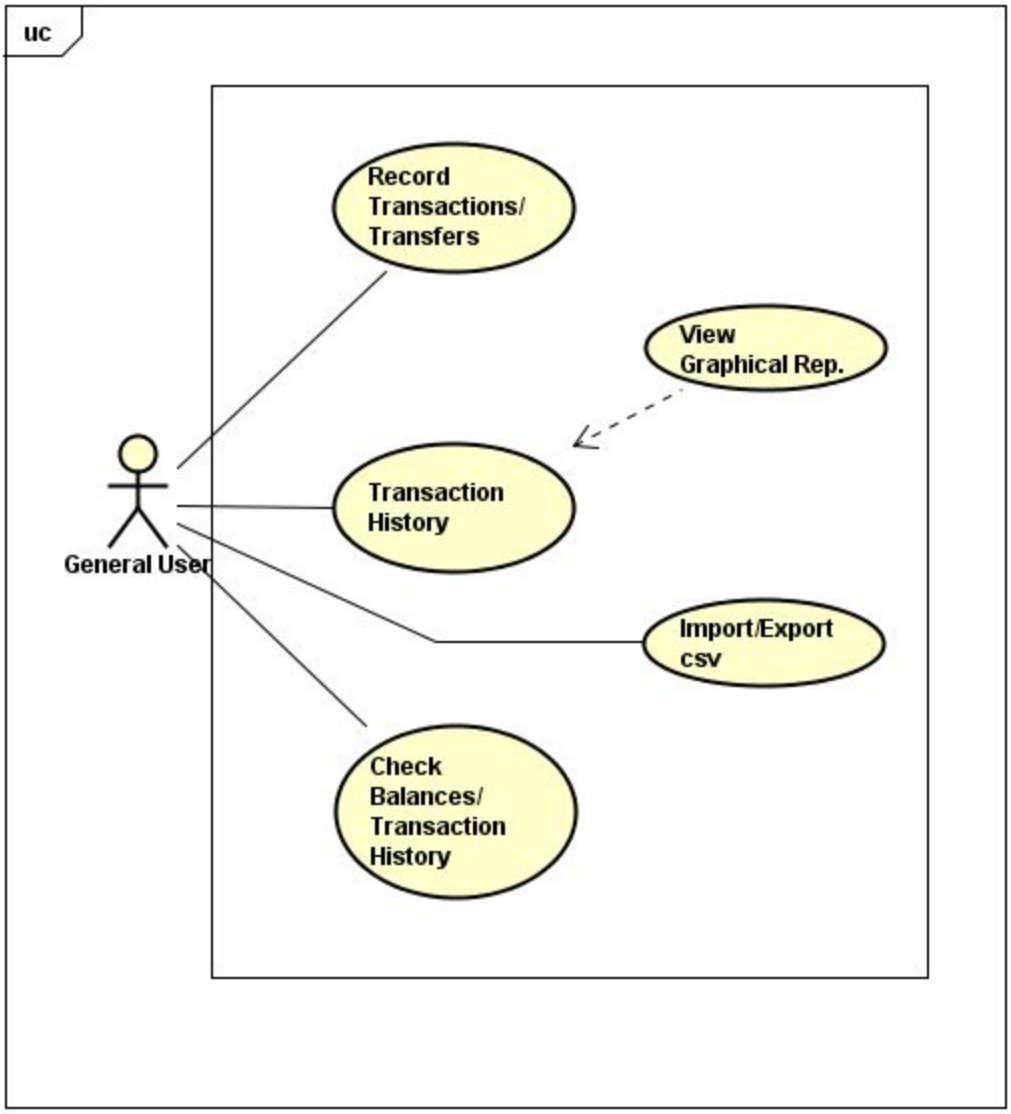
11.3 Sequence Diagram

See "sequence diagram.png" that is located with this document.

The sequence diagram shows how different components of the software interact as time goes on, from top to bottom.

11.4 Use Case Diagram

This diagram shows what uses a customer might have for the app, or what they will be using the app for.



APPENDIX A// REQUIREMENTS TRACEABILITY MATRIX

[This can be annotated on the Requirements Traceability Matrix, in the Design Review document, or in the Configuration Management System.]

[Organization/Project]
Requirements Traceability Matrix
(RTM)

Requirement Name	Priority	Risk	SRS Ref.	SDD Ref.	Validation Method(s) *	Formal Test Paragraph
Input Personnel Data	H	L	3.1.1	6.2.3	A	6.3.14
Store Personnel Data	M	H	3.1.2	6.2.2	A,I	6.3.17

*** Validation Method(s)**

D - Demonstration
T - Test
A - Analysis
I - Inspection

Field Descriptions

1.	**Requirement Name**	A short description of the requirement to be satisfied
2.	**Priority**	High (H), Medium (M), Low (L); to be negotiated with customer and remains **fixed** throughout software development lifecycle
3.	**Risk**	High (H), Medium (M), Low (L); determined by technical staff and will change thoughout software development lifecycle
4.	**SRS Ref.**	Paragraph number from section 4 of the SRS.
5.	**SDD Ref.**	Paragraph number from section 6 of the SDD.
6.	**Validation Method(s)**	Method to be used to validate that the requirement has been satisfied.
7.	**Formal Test Paragraph**	This column will provide a linkage between the SRS and the software Test Plan. This will indicate the test to be executed

to satisfy the requirement. (This column will be further defined in later EPG plans.)