Software Requirements Specification

for

Project: ManageMe (Group 2)

An All-in-One Self-Management System

Prepared by
Ankur Sharma (2015CS50278)
Sudeep Agrawal (2015CS50295)
Lovish Madaan (2015CS50286)
Siddharth Khera (2015MT60567)

Supervised by
Prof. S.C. Gupta

COL 740 - Software Engineering
Indian Institute of Technology, Delhi
Hauz Khas, New Delhi

Contents

Chapter 1

Introduction

1.1 Purpose

The purpose of this android application is to provide an easy and user friendly way to keep track of your expenses/food calories/important notes at one place. This application is motivated by the fact that you have to use different applications for each of these use cases. Sometimes, these use cases are inter-dependent, for example, you might want to note down certain items to buy depending upon your current expenses. We also provide category wise tags to organize the items in the app for easy access.

Our project aims to create a standalone platform for self management through which one can manage all their notes, memos, expenses, food intake / calories using just one application.

1.2 Scope

Scope of this project is building a complete standalone android application which will have different functionalities, all specific to different self-management tasks. In this project we will build the following:

- **Authentication module**: User login and Signup is handled by this module.
- **Fitness module**: Manage the fitness section of the user datewise with customized tags.
- **Expenses module**: Manages the expenses section of the user on various items of manually entered cateegories.
- **Notes module**: Keeps track of personalised notes of the user in different scenarios.
- **Tagging module**: Creates and manages custom tags for notes, expenses, food items, etc.
- **Filtering module**: Allows the user to filter expenses, calories, etc. between a user-selected start and end date

1.3 Definitions, Acronyms and Abbreviations

- **Customer/User**: Here the customer and user are both the same as we are developing this application for the end user directly. Users will be those people who want to personalise and manage their own experience in their own way.

- **Supplier**: We will be the sole suppliers of the application. We will build the app and publish it to the web where users will be able to download and use it.
- **Item**: An item is a data entry which could be of any category i.e. an item could be a note, an expense entry, workout entry etc.

1.4 References

A lot of guidance has been taken from these given IEEE standards to produce this document:

- IEEE Std 1016-1998, IEEE Recommended Practice for Software Design Descriptions.
- IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications.

1.5 Overview

This report contains all the information needed to understand the application. Section 2 provides the overall description of the application including product perspective, product functions, user characteristics, constraints, assumptions and dependencies. Section 3 provides the details about the specific requirements of the application.

Chapter 2

Overall Description

2.1 Product Perspective

ManageMe is supposed to be a simple self management app. It has the following main functionalities -

- **Expense Management:** A user can manage all the daily expenses by keeping track of each expenses with specific tags. We also provide a filter function through which reports can be generated for any date range selected by the user.
- **Notes:** A user can add to-do lists, simple notes with specific tags. Tag wise search would ensure all the items under a particular tag can be viewed.
- **Calories Record:** One can maintain a record of daily food intake by adding food items and their associated calories already present in our large database. It can be used for weekly / monthly analysis of total calories consumed.

2.1.1 System Interface

This app will be a standalone application, which is not a part of any bigger software. So all the interfaces defined here and after this, are based solely on the requirements to be met.

2.1.2 User interface

We intend to divide our application into different independent sections in an intuitive fashion. The home page of the app would require user to Sign Up / Login. Once the authentication is done, the main page would appear which would list the expense, notes and fitness section corresponding to the logged-in user.

Notes page would list all the notes in chronological order with a + (Plus) button to add a new note. A side panel would list all the tags and clicking on a tag would list all notes under that tag.

Expense page would list all expenses in chronological order with a + (Plus) button to add new expenses. Each expense would be a name value pair along with tags. For eg. <Uber, Rs. 100, [Transport, Travel]>. There would be a button that shows the total expenses for a given time period.

The calories page would list daily food consumption record. For example, <Lunch, [<Idli, 20 calories>, <Sambhar, 30 calories>] >. There would be a button that shows the total calories consumed for a given time period selected by the user.

2.1.3 Hardware Interface

Our app doesn't have any specific hardware requirement except its fair share of storage, RAM and CPU.

2.1.4 Software Interface

ManageMe shall require a server to manage all the notes, expense records, calorie counts, photos and messages from all the users. We are planning to integrate AWS (Amazon Web Services) into our app so that it becomes easier to scale by putting our database on cloud.

2.1.5 Communication Interfaces

The app shall use the HTTP protocol for communication over the internet and for the intranet communication will be through TCP/IP protocol suite.

2.1.6 Memory Constraints

The device shall have enough memory to store the added items. In case of any memory shortage app shall display an error message notifying the user about the same. Also, enough memory should be available to perform filter operations on notes, fitness records and calorie calculator.

2.1.7 Operations

Operations required in the app shall be:

- Add (an item) operation, initiated by user.
- Filter operation, initiated by user.
- Tagging operation, initiated by user.
- Note storing, automatically initiated by the app.
- Data (Notes, Records, Tables, Texts, etc.) backup automatically initiated by the app whenever internet connectivity is available, otherwise data is stored locally on the phone.

2.1.8 Site Adaption Requirements

This app shall work properly in all given scenarios if all the constraints are met. As this is an android application working mostly over the internet network, no particular constraints are set by the environment this app is used in.

2.2 Product Functions

2.2.1 Context Diagram

Figure 2.1 shows the context diagram for our android application, ManageMe.

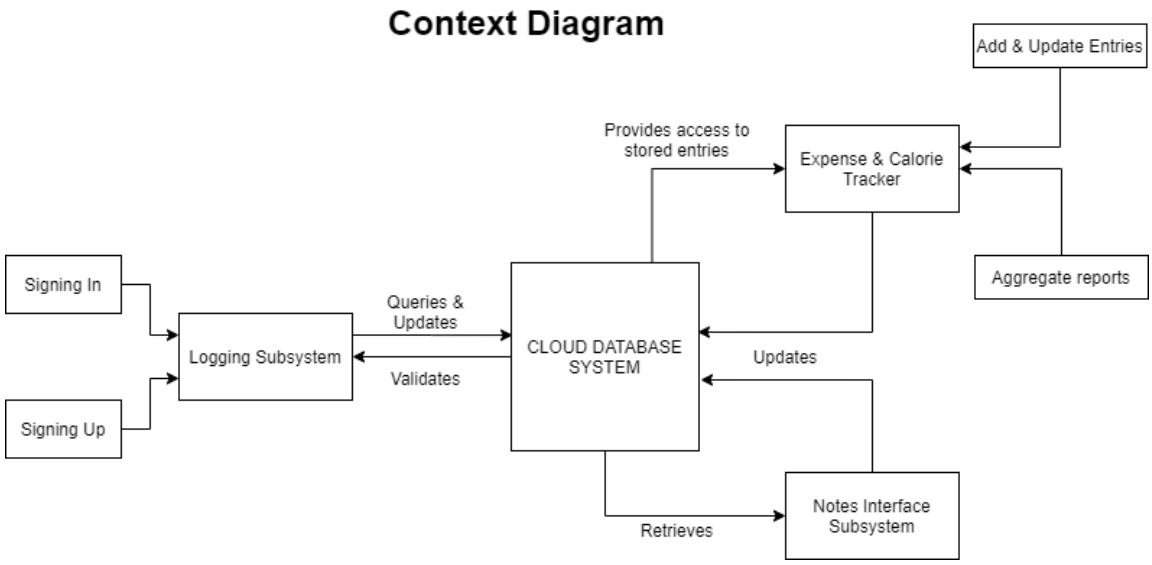

Figure 2.1: Context Diagram for ManageMe

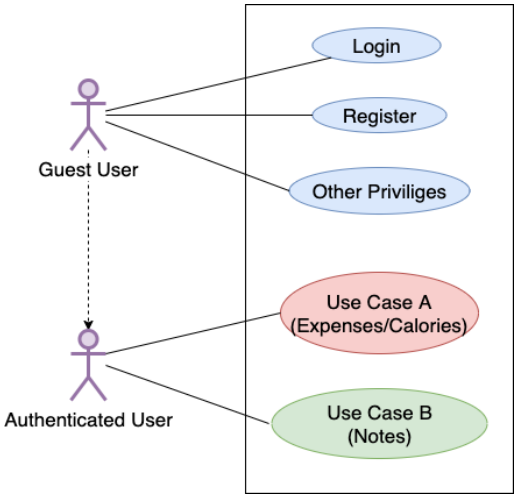

Figure 2.2: Use Case Diagram for Login/Signup Activity

2.2.2 Use Case Diagram

In this section, we describe our use case diagrams. We break our complete functionality into 3 use case diagrams. Figure 2.2 shows the use case diagrom for login/signup.

Figure 2.3 shows the use case diagram for the calorie tracker and expense tracker modules.

Figure 2.4 shows the use case diagram for the note taking module.

2.2.3 Use Case Description

Here we define the use case requirements of the app which clarifies a function of a system and its components in the form of diagrams.

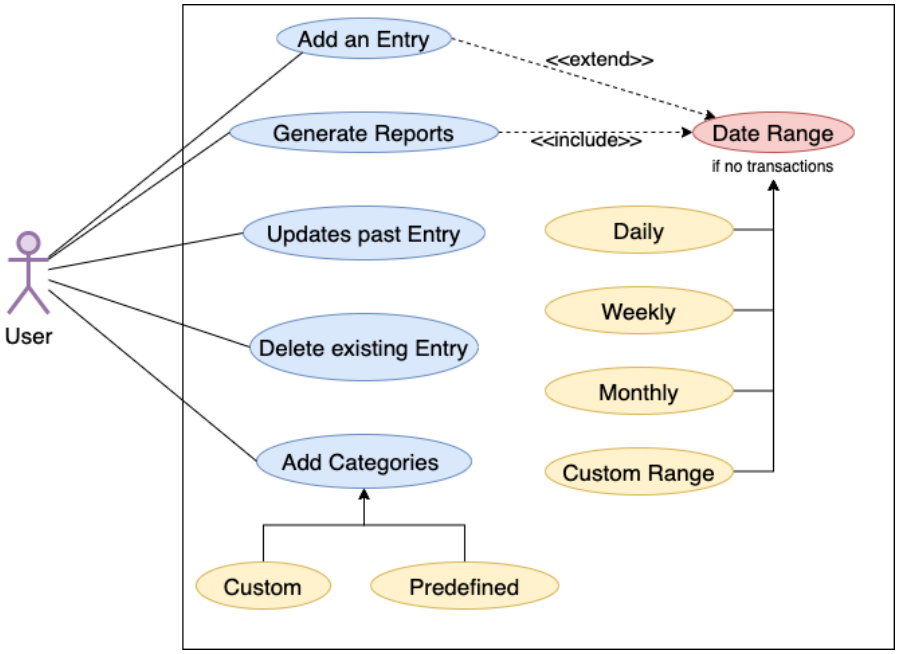

Figure 2.3: Use Case A corresponding to Expense Tracer & Calorie Tracking

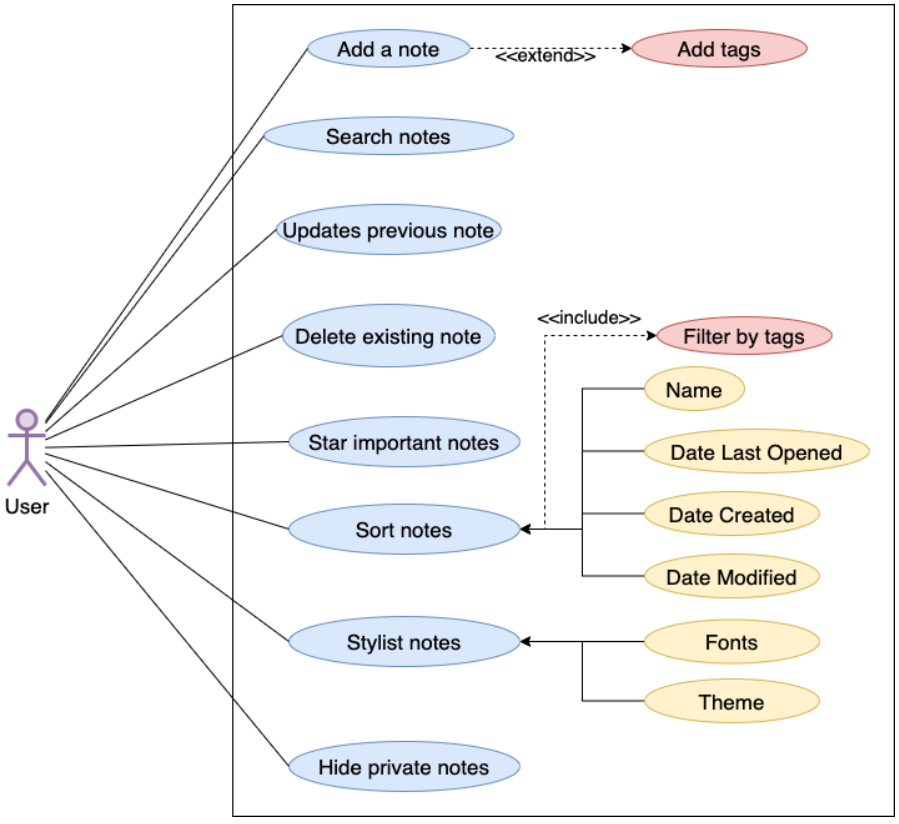

Figure 2.4: Use Case B corresponding to Note taking

The ManageMe application primarily solves 3 main use cases :

- Making Notes as detailed in Figure 2.5

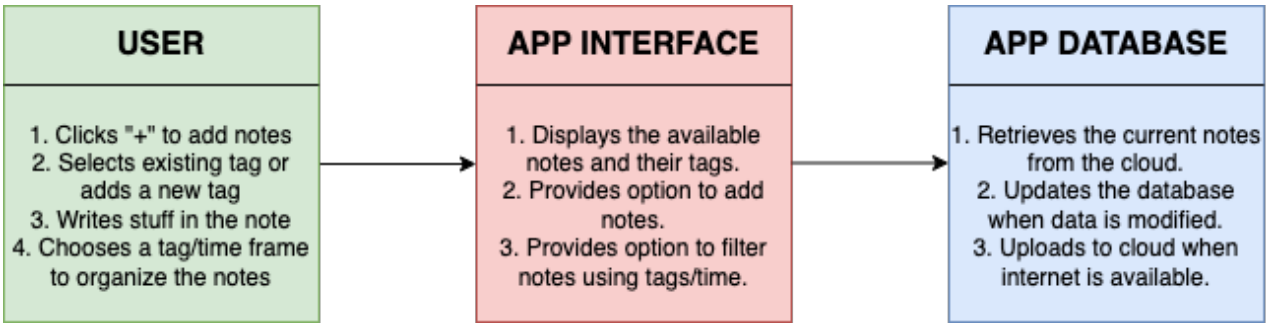

Figure 2.5: Use Case : Notes

- Tracking user's diet and calories as detailed in Figure 2.6

Figure 2.6: Use Case : Calorie Tracker

- Aggregating user expenses as detailed in Figure 2.7.

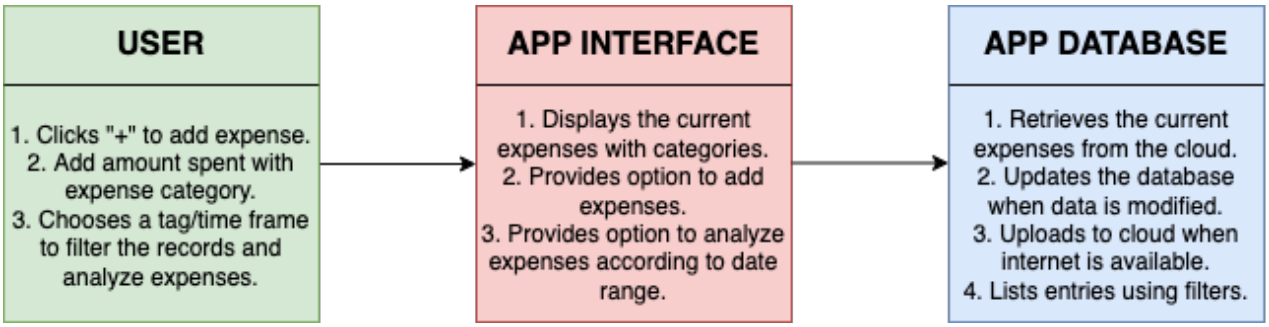

Figure 2.7: Use Case : Expense Tracker

2.3 User Characteristics

User must be of age 8 and above, because users can starting managing themselves at this age. Apart from that, no other user restrictions are put in place with respect to the use of this application.

2.4 Constraints

- **Memory constraint** Each user would be allotted a fixed storage capacity in cloud database.
- **Tag constraint** We are limiting the tags to 100 to avoid redundant tags.
- **Fast synchronization** The local data may take time to sync with cloud storage. Hence local changes won't be instantly synced with cloud storage.

2.5 Assumptions and Dependencies

- **Android Environment**: This app is developed only for android and thus is dependent on android platform services. It will work on all android devices with support of API Level 23 or more.

2.6 Apportioning of Requirements

- **Incorporating visual media in the notes app**: This is a very useful feature that can be added in later versions that user be able to add the images in the context of the notes. This will heavily enhance usability of the app.
- **Fetching health data from a Fitness app**: We can use pre-existing optimized applications like Samsung Health, Google fit, etc. to populate the records of our Calories section.
- **Sharing of personal data**: We can allow the users to share their personal data (expenses records, food consumption, etc.) with other people if they want to.
- **Filtering by Expenses/Calories tags**: We can allow the users to filter their expenses or food categories as per various categories.

Chapter 3

Specific Requirements

3.1 External interface

- **Touch Input**: We're taking screen touches as our input for controlling and using the app. We get this input using *onTouchListener* function in Android API. We get both the touch events and the coordinates of touch to control the app, which is very standard in any android app.
- **Expense Reports**: We can export our expenses in some suitable format (e.g. CSV, JSON, XML, etc.) for each category from our application directly so that users can have a hard copy of their data too. The same expense reports will also be synced with the cloud database for greater access.
- **Calories Reports**: Similarly, we can dump the food consumption data in terms of calorie intake for any given date range via this export option.

3.2 Functional Requirements

We aim to add the following functionality in our project:

- The software shall provide an interface to allow users to sign-up or login using their email-IDs. Once logged in, the user stays logged in until he chooses to log out or delete the app.
- Filtering the expenses or calorie information with several date ranges and categories. One can filter it on a weekly, daily, monthly or can select any custom dates.
- Text/drop-down facility for entering expenses and food details. Expenses and calories information can be updated and deleted at any point of time from the database.
- Tagging facility to add multiple tags or categories associated with notes, expenses, food items, etc.
- Feature for creating user-defined customised tags for expenses and calories.
- Exporting our expenses or calories report in some suitable format so that it can be shared across platforms and users.
- Save items to local storage and to the AWS server along with the timestamp of entry or update.
- Specific features for notes module:
  - Searching notes based on topic/text matching.
  - Hiding all notes marked as private.
  - Starring our most favorite notes, and they will appear on the top of the list view for all notes.

- Altering theme and styling our notes as we like as per various color/theme options.
- A text view for adding notes with a topic name for each note. It will offer the facility to save, update, add or delete any notes.
- Sorting notes based on the date last opened, modified, created, tags, etc.

3.3 Performance Requirements

3.3.1 Consistency

We aren't looking for strict consistency, we want eventual consistency. An item once modified / added / deleted must be modified / added / deleted in cloud storage as well.

3.3.2 Storage

The total data required by a user can not exceed the total local storage on the device, unless the user decides to delete some of the data.

3.3.3 Upload Time

We are using AWS as our backend for database and storing photos. AWS ensures 99.99% uptime for S3 storage and 99.95% for RDS relational database. AWS uses very good network connections, so the upload time is only dependent on the user's device and internet connection.

3.4 Logical Database Requirements

We shall use AWS RDS for maintaining our database about users, records and pictures. Each of above mentioned entity shall have a unique identifier. We shall also have relational tables to specify the relations between these entities. We shall have the following relational tables:

- **Users**: `<user-id, email-id, username, password>`
- **Expenses**: `<user-id, expense-id, name, value, (tag1, tag2, ..), date>`
- **Calories consumed**: `<user-id, calorie-id, name, calories-consumed, (tag1, tag2, ..), date>`
- **Notes**:
  `<user-id, note-id, title, text-content, (tag1, tag2, ..)>`,
  `<user-id, note-id, font-id, hide-note, color-theme, star>`,
  `<user-id, note-id, date-created, date-modified, date-last-opened>`

3.5 Design Constraints

- The design has to be according to the database which will be used. We have to keep in mind the tabular structure of the SQL database.
- Whatever the design of software is, it should be in accordance with the relationships in SQL database.
- The app will only work for android phones with a camera, so the phone should have a camera to share photos.
- The app has to work on android mobile phones, so the app should not use a lot of RAM.
- Also, the design has to be implemented in an android application, so it has to designed accordingly.

3.6 Software System Attributes

3.6.1 Reliability

- The App should not crash when a lot of notes / expenses / food items etc. are clicked.
- The user should be able to add offline notes / expenses / food items entry even when there is no internet connection.
- Any offline changes should be reliably synced with cloud including tags.

3.6.2 Availability

- When the app crashes, it has to be restarted, on starting it syncs with the cloud and is able to get all the user data.
- If for some reason, items stored locally are lost, the user will still be able to get items uploaded previously, from the cloud.

3.6.3 Security

- A user cannot access any item of another user from both local and cloud storage, not even for reading purposes.
- The local app user data shall not be shared to any other app.

3.6.4 Maintainability

- Modularity should be present in the design and implementation so that it is easy to maintain code in future.
- Different modules have been specified above and the modular structure should be followed

3.6.5 Portability

- This software will be highly portable and can be used on any Android phone.
- There might be a minimum required version of the Android operating system, but the app will work on all phones with version same as or more than that.

3.7 Organising Specific Requirements

Requirements in this software can be organised based on the feature to which it is related. This has been done in section 3.2.

3.8 Additional Comments

- The app should be easy to use and user-friendly.
- User should be able to see all items of a category in one place in chronological order.
- The look of the app should be pleasing to the eye
- UI should be well spaced and it should not be clumsy

Chapter 4

Supporting Information

4.1 Appendixes

This app solves the painful problem of using multiple different applications for catering the user's personal needs. All the data fed to this application gets automatically uploaded to the AWS server and can be accessed anywhere. The data is kept in a suitable, safe and accessible format and is easily synchronized with the application.