Software Requirements Specification

for

Xpendit

Prepared by Aiden Lammert, Matthew Prescott, Matthew Ackley, Durgin Sweet, Joshua Schaff

February 26, 2019

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

# Introduction

## Purpose

Xpendit is a group finance managing application designed to help friends and roommates keep track of what they owe each other. The application will allow groups of people to track any fee that a group of people might share with each other, such as rent, apartment maintenance, or the cost of going out to eat. Regularly recurring costs can be set up to ensure that continual expenses such as rent, or utility bills are paid on time. Xpendit will also allow groups to create shared shopping lists to make it easy to keep track of what the group needs, how much each item costs, and how much each member of the group needs to contribute financially.

## Document Conventions

All requirements are listed within the document and with their own priority. High-level requirements are assumed to share priority level with lower level requirements.

## Intended Audience and Reading Suggestions

The Xpendit System Requirement document is intended to use by the project team. Management, developing, document writing, and testing will be divided among the five members of the team. The System Requirement Specifications document outlines the scope of the project and overview. More specifically, the document covers the design and implementation requirements for the UI and underlying system of the final product.

## Product Scope

Xpendit’s goal is provide a simple way for groups to tracks finances. Users create groups with other users to tracks expenses shared between each other extending from recurring bills to lending each other money. Users can create or join multiple groups while still tracking the total expenses to help manage their finances. Xpendit allows for flexibility in tracking any type of expenses and the amount shared between each member of a group.

## References

# Overall Description

## Product Perspective

Xpendit is a recreation of other similar roommate applications, but with more flexibility. Other applications only offered some of the features of Xpendit and were limited in certain aspects. This is not a replacement for another product, but its own self-contained product.

## Product Functions

* Individual user accounts
  + Creation of accounts
  + Managing Accounts
* Creating rooms for users
  + Tracks expenses between users of rooms
  + Provide a group shopping list
  + Provide a group bulletin board
  + Add users to room
  + Remove users from room
  + Delete room
* Track user expenses
  + Add expense
  + Remove expense
  + Show total of all expenses
  + Show who expenses are owed to
  + Allow users to dispute charges from other users
  + Allow expenses to be split among groups evenly, by percentage or specific dollar amounts
  + Manual payment confirmation of expenses
  + Prioritize expenses by age, amount due or custom priority

## User Classes and Characteristics

The intended audience, users, will have android devices, which will have access to the full app. All members of the development team will have laptop or desktops capable of app development. The Development team will need full access to all files, documents and databases.

## Operating Environment

Xpendit will operate on Android 8.0 Oreo. The majority of Android devices currently operate on this version. The app is designed to be standalone and will no interact with any other systems.

## Design and Implementation Constraints

Flutter will be used for the development of the application. Development of the application will be restricted to Android, thus will be restricted to the hardware limitations of Android devices. The application will keep user’s data secure as well as not track user location.

## User Documentation

The app will be released without any external documentation.

## Assumptions and Dependencies

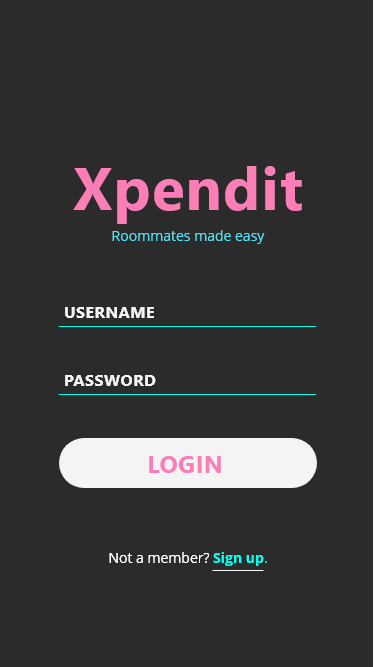
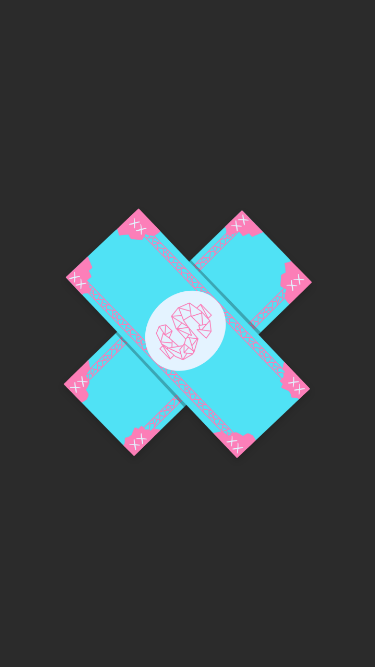
All requirements are expectation are based on having the required hardware to host the application from and

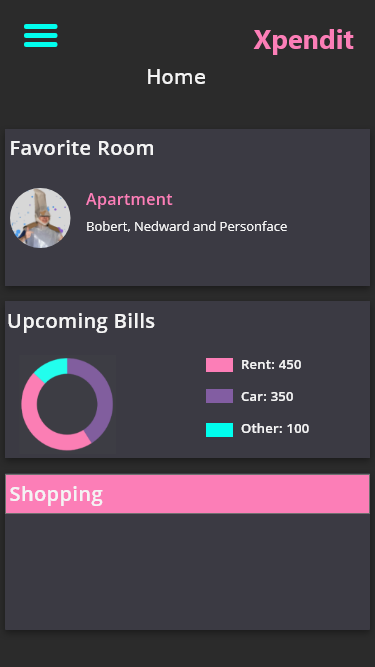
# External Interface Requirements

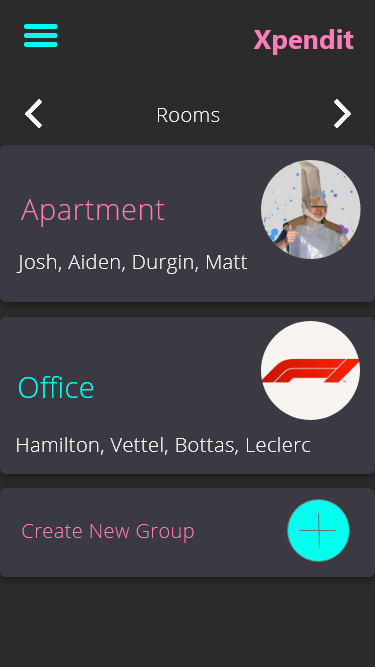
## User Interfaces

Xpendit is designed to use with a touch screen device. All interaction between the user the application will be through on-screen buttons and keyboards.

**Mock-up UI:**







## Hardware Interfaces

The application is built for Android device. It is meant to be standalone, except for having to interact with the account database.

## Software Interfaces

The software will need to connect to a database in order to store information about groups and users; however, the specifics of the database and the communication are still to be determined.

## Communications Interfaces

Xpendit applications will communicate with the user and group database server.

# System Features

## Debt Management

4.1.1 Description and Priority

This is the system’s ability to track the debts of a user, including whether owed to a specific user or to the room as a whole. This is the primary feature of the system, so this is high priority.

4.1.2 Stimulus/Response Sequences

The user will be able to input a debt owed to the requester by a second user or multiple other users within a room. The system will then check to see if the room is valid and both users are within the room, before sending a notification to the second user(s) that the debt has been charged. The second user(s) can then choose to accept or dispute the charges. If the charges are accepted, they are applied to the second user(s) balance, and if the charges are disputed, then the charges are not applied. In either case, the system notifies the requesting user of the status of the charge.

The user should be able to see the debts they owe, itemized by each individual charge. The charges should include the amount of the charge, the person that the charge is owed to, and the amount of time passed since the charge was issued. The user should also be able to organize these charges based on any of the three categories, as well as by user-defined sorting methods.

The user should be able to pay off debts physically and log in the system that the det has been payed off, requiring the person that the money was given to agree that the transaction happened.

4.1.3 Functional Requirements

REQ-1: The system must allow a user to charge debt to one or more users.

REQ-2: The system must be able to alert a user if a charge is file against them.

REQ-3: The system must allow the user to accept or dispute a debt charged to them.

REQ-4: The system must alert the user if a charge they have made is accepted or disputed by the other party.

REQ-5: The system must b able to keep track of a user’s debt.

REQ-6: The system must be able to add to or subtract from a user’s debt.

REQ-7: The system must be able to display a user’s debts, along with who the debt is owed to and the time since the debt was applied.

REQ-8: The system must be able to sort the debts by amount owed, person the debt is owed to, the time since the debt was applied, and by any user defined priorities.

REQ-9: The system must allow the users to define their own priorities to be used when sorting debts.

REQ-10: The system must allow users to mark debts as paid with the acceptance of the other person.

## 4.2 User Account Management

4.1.1 Description and Priority

*This is the system that covers the creation and management of user accounts. Users should be able to create accounts as well as log in or out as needed. This is also necessary for our system to work, so it is also high priority.*

4.1.2 Stimulus/Response Sequences

Users should be able to choose to create an account if they don’t already have one, requiring the creation of a username and password. The system must make sure that the username is unique.

The user should also be able to log in as a user with the correct username and password and be able to log out afterwards.

4.1.3 Functional Requirements

REQ-11: The system must allow the user to attempt create an account with a username or password.

REQ-12: The system must be able to check if a proposed username is unique or not.

REQ-13: The system must only accept the account if the proposed username is unique

REQ-14: The system must allow the user to log into an account if they are not currently logged in to an account.

REQ-15: The system must allow the user to log out of an account if currently logged in.

## 4.3 Room management system.

4.1.1 Description and Priority

This feature covers the creation and maintenance of “rooms” with multiple users, including creation, destruction and the addition/removal of users to the room. This is the final cornerstone of our system, so it is also considered high priority.

4.1.2 Stimulus/Response Sequences

The user should be able to create a room, choosing the name of the room and adding at least one other person to the room by searching for their username. If the username is not in the servers, then the system should give an error message and refuse to try to add them to the room.

After the room is created, users should be able to add more users to the room at any time by searching for their username. If the username is not in the servers, then the system should give an error message and refuse to try to add them to the room.

Once the room is created, users should also be able to try and leave the room. If the user leaving the room owes debts to other people in the room, the system should prevent them from leaving until the debts are repaid. Once the user has no debts left, they can leave whenever they want.

Users should also be able to delete a room, so long as nobody in the room owes any money and all current members of the room agree to delete it. The room should also delete itself if all members leave.

4.1.3 Functional Requirements

REQ-16: The system must allow users to create rooms containing themselves and other users

REQ-17: The system must allow users in the room to add other users not currently in the room.

REQ-18: The system must allow users without outstanding debts to remove themselves from a room

REQ-19: The system must allow a room to be deleted if no outstanding debts are owed between any of the members in the room.

## Miscellaneous Room Features

4.1.1 Description and Priority

These features covet the group bulletin boards in the rooms created by the users that allow users to leave notes for other members of the group, as well as the shopping list that the users can use to request items that other users can buy and charge immediately to the requesting user. These features aren’t as central to our app, so they can be considered medium priority.

4.1.2 Stimulus/Response Sequences

The user should be able to open a “bulletin board” tab when viewing the room information, which should allow the user to select from other posts that have been made by users in the bulletin board or to create their own topic. If the user views a topic, they should be able to reply to the post as well.

Their should also be a shopping list tab within the room viewer, which should allow the user to see requests that members of the group have made for purchasing. The user should be able to mark any item that has been put on the list as bought, removing it from the board and going immediately to the apply charge screen if they are buying something for a different user. The user could also add things to the shopping list, and users should also be able to remove something they added to the shopping list without charging anyone.

4.1.3 Functional Requirements

REQ-20: The system must allow users in a group to view the bulletin board for any group they are in.

REQ-21: The system must allow the user to create posts in the bulletin board

REQ-22: The system must allow the user to comment on posts that have been created in the bulletin board.

REQ-23: The system must allow users in a group to see the shared shopping list in a group.

REQ-24: The system must allow users to mark that they have bought an item in the shopping list.

REQ-25: The system must apply the charges to the person corresponding to the item in the shopping list that is bought.

REQ-26: The system must allow the user to delete items from the shopping list that they added themselves.

REQ-28: The system must automatically remove ought items form the shared shopping list.

# Other Nonfunctional Requirements

## Performance Requirements

Xpendit should be able to run anywhere were it can access the internet to communicate with the backend that keeps track of the data. Beyond that, Xpendit should run in a timely manner, with no waiting times between actions greater than 20 seconds in 90% of cases.

## Security Requirements

Xpendit will require a username and password to log into an account, and when creating the account, will require that the password be of good quality, such as enforcing 10+ characters, multiple character types, and checking the password against common passwords to ensure that it is not easily cracked.

## Software Quality Attributes

Xpendit will require moderately scalable software; users will be able to be part of many groups and the groups can each potentially have many people as well, so debt tracking and cost splitting could potentially grow quite complex. The software also has to be perfectly correct in keeping track of the debts owed so that people do not end up loosing money because of incorrect debt collection.

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

The exact method of user and group data storage still needs to be determined.