Software Requirements

Specification

for

Bounce

Version 1.0

Prepared by Bounce

Dev. Team

Nov. 14th, 2014

# Table of Contents

1. Introduction 3
   1. Purpose 3
   2. Scope of System Specified 3
   3. Definitions, Acronyms, and Abbreviations 3
   4. References and Supporting Documents 5
   5. Overview of the rest of the SRS. 5
2. General Description 7
   1. Product Perspective 7
   2. Product Functions 8
   3. User Characteristics 9
   4. General Constraints 9
   5. Assumptions and Dependencies 10
3. Functional Requirements 11
   1. User Requirements 11
   2. System Requirements 11
4. Nonfunctional Requirements 13
   1. Interface Requirements 13
   2. Software Requirements 13
   3. User Requirements 13
5. System Architecture and Models 14
   1. Use Case Diagram 14
   2. Class Diagrams 15
   3. Sequence Diagram 20
6. Appendices 21

# 1. Introduction

## 1.1 Purpose

The purpose of this document is to provide a detailed description of the Bounce To-Do list mobile app. This document will completely explain all system functionalities, constraints, interfaces, and interactions with all users. This document will serve as both an illustrative guide to developers working on Bounce, as well as an explanation of the system for app users.

## 1.2 Scope of System Specified

Bounce is a mobile To-Do List application developed for Google’s Android mobile OS. As a to-do list, Bounce will contain features considered ‘standard’ in to-do list applications (task creation, list organization, etc.), as well an extensive complement of features either already found across a wide range of existing applications or tailored for this app.

Bounce operates in a Layered Architecture, where the mobile front end serves as an interface between the user and data stored in a cloud database. Functionality will eventually be expanded so this backend data will be accessible from multiple platforms.

## 1.3 Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| **Terms** | **Definitions** |
| App | Application, typically on a Mobile platform. |
| Boolean | A value which is either true or false. |
| Google | Technological giant which provides online search, email, document editing, and many other services. |
| Google Account | An account with universal access to all Google brand products. |
| Google Play Services | A platform with several APIs which manage external application’s access to Google accounts. |
| Google Plus | A Google product designed for social interaction between Google Account owners. |
| Item | Abbreviation for “To-Do Item”. |
| Material Design | Google’s Android standardized app style. |
| Mobile | Abbreviation for Mobile Cellular Device (typically a phone). |
| OS | Operative System. |
| Task | A goal that the user wants to accomplish. |
| To-Do List | A method of organizing tasks. |
| To-Do Item | A data-bearing entity representing a task or other goal. |
| UI | User Interface |
| TCP | Transfer Control Protocol.  A networking protocol where a connection is established between the two parties. Once a connection is established, continuous data transfer can begin. |
| IP | Internet Protocol.  The principal communications protocol in the Internet protocol suite for relaying datagrams across network boundaries. |
| Encryption | The process of encoding messages or information in such a way that only authorized parties can read it. |
| Decryption | The reverse process of encryption, in other words, moving from unintelligible ciphertext back to plaintext. |
| Cipher | An algorithm for performing encryption or decryption - a series of well-defined steps that can be followed as a procedure. |
| Ciphertext | The result of encryption performed on plaintext using an algorithm, called a cipher. |
| Plaintext | Information a sender wishes to transmit to a receiver. |
| Fail2Ban | An intrusion prevention software framework which protects computer servers from brute-force attacks. |
| iptables | A user-space application program that allows a system administrator to configure the tables provided by the Linux kernel firewall and the chains and rules it stores. |
| ClamAV | A free and open-source, cross-platform tool-kit able to detect many types of malicious software, including viruses. |
| Apache | The world’s most widely-used web server software. |
| MySQL | The world’s second most widely used open-source RDBMS. |
| RDBMS | Relational Database Management System. |
| Kerberos | A computer network authentication protocol which works on the basis of ‘tickets’ to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. |
| BLOWFISH | A symmetric-key block cipher which provides a good encryption rate in software and no effective cryptanalysis of it has been found to date. |
| Ethernet | A family of computer networking technologies for LAN and larger networks. |
| Datagram | A basic transfer unit associated with a packet-switched network. |
| Packet | A formatted unit of data carried by a packet-switched network. |
| CentOS | Community Enterprise Operating System |
| SELinux | A Linux kernel security modules that provides a mechanism for supporting access control security policies, including mandatory access controls. |
| Routing | The process of selecting best paths in a network. |
| Forwarding | The relaying of packets from one network segment to another by nodes in a computer network. |
| DNS | Domain Name System |

## 1.4 References to Supporting Documents

1. IEEE Std 830-1998. IEEE Recommended Practice for Software Requirements Specifications.
2. Bruegge, Bernd, and Allen H. Dutoit. Object-oriented Software Engineering: Using UML, Patterns, and Java. 3rd ed. Boston: Prentice Hall, 2010. Print.
3. Krause, Peter. Personal interview. 5 December 2014.
4. Material Design Specifications, Google Inc.

## 1.5 Overview of the rest of the SRS

The second section of this document will provide an in-depth description of the application, such as the product’s features and interfaces. The third and fourth sections will provide a complete and comprehensive list of all functional and non-functional requirements, respectively. Section five will provide UML use-case, sequence, and class diagrams for the application. Finally, section six will show some screenshots from functional Android prototype.

# 

# 

# 2. General Description

## 2.1 Product Perspective

The core functionality of Bounce is to provide a feature-filled platform for managing a user’s to-do items. In doing such, Bounce relies on multiple external service providers to manage user access and content backup and synchronization.

Bounce contains a local SQLite database for storing task entries. All user transactions are immediately stored at the time of the transaction to this database. This database is synchronized to Bounce’s server database in order to both backup user data, and to allow for synchronization across multiple devices and platforms. Additionally user preferences are also stored and synchronized across a user’s devices.

User authentication is provided by Google Play Services, specifically using Google Accounts via Google Plus. As one of the leading companies in cloud technologies, Google can provide one of the most secure user authentication platforms available. The Google Plus API provides modular access to different sections of a user’s Google Accounts profile. For Bounce, the only user data being pulled from Google will be identification information, as well as a user’s profile picture and ‘cover photo’ from Google plus for interface aesthetics.

User data management is handled in a secure and concise manner. All data is encrypted using a cipher that is built for security, not speed (think BLOWFISH). Security is more important than speed. A 3rd party shall not be able to read another user’s tasks as they get transferred between the user’s address and the server’s address. The data itself is stored in a secure relational database (after getting scanned for viruses). We optimize the speed it takes to process a user’s query to make up for the time it takes to decrypt the messages between addresses. To sum up data management: it is secure and efficient where it can be.

## 2.2 Product Functions

Bounce provides users with a bounty of features found across multiple existing to-do apps in addition to a complement of brand new features specific to Bounce. Users can create To-Do items in Bounce, and organize these items into lists. Items can have many different attributes, including a Deadline date, an incremental Priority, a boolean Starred value, Reminder dates and times, an associated parent list, and a boolean completed value.

Bounce provides several dynamic organization options for viewing tasks. Tasks can be viewed individuals, or in their associated lists, or in any of the default views provided by Bounce. Defaults views include an Inbox, an Upcoming tasks list, a Completed list, and a list for All Tasks. These default views are designed to create clutter-free and efficient interfaces to view collections of tasks through.

Bounce implements several UI functions which manipulate and display the tasks in a simplified manner. First among these is the namesake ‘Bounce’ feature, which easily postpones tasks which appear in the Inbox view. Bounce also allows for linking tasks together, which will prevent clutter in list views. Linking tasks will hide tasks whose linked ‘predecessor’ tasks have not yet been completed. Location based reminders are another key feature. This will allow users to complete tasks such as ‘Get Groceries’ by notifying users when they are close to a Publix or similar. Finally, the Calendar View will present a heat-map calendar which shades days based on the number of tasks assigned to a certain day.

Bounce will synchronize its tasks to a web server and database. This is done using a most recent update scheme. Users’ who use the Bounce App on Android will have a local database that stores their tasks. Each task has a timestamp of when it was last modified. At intervals of 4-5 minutes, the App will attempt to make a request to save the user’s tasks onto the Bounce database. The backend will compare the timestamps of each corresponding task and will overwrite the task with the version with the most recent timestamp. This ensures that the users’ tasks are always the most up-to-date version across different platforms (web, Android, iOS, etc.). When users access Bounce through the website, the website interfaces directly with the database server, so that the version of the tasks accessed is always the most up-to-date (within a range of +- 4-5 minutes if users have constant access to the internet via their phone).

## 2.3 User Characteristics

The users of Bounce are of unlimited variety in nature or profession. As a to-do list, Bounce could be used by an individual of any age, ethnicity, profession, or any other specifiable characteristic so long as they meet some very general constraints.

First, users will need access to a supported device. For the initial release, this will be limited to either an Android device running Android version 4.1 (Jelly Bean) or later, as well as any device capable of accessing the internet via modern internet browser (e.g., Google Chrome or Safari).

Second, users will need a basic understanding of a supported language. For the initial release, U.S. English will be the only supported language. Only simple terms will be used to specify interface elements, so even a young child will be able to use this application.

There will be no formal instruction for users of this application. As with many Android applications, the Android application will provide a brief paginated tutorial upon the initial launch of the application, which will provide an overview of the functions. Additionally, there will be a ‘Help’ section available for additional aid.

## 2.4 General Constraints

Bounce is dependant on the following constraints. These constraints are divided into two sections, the mobile section and the web section.

1. Mobile Constraints
   1. Use requires access to a mobile device running a supported operating system.
   2. The initial download will require an internet connection.
   3. User authentication will require an internet connection.
   4. Task and list synchronization require an internet connection.
   5. There will be a very large but finite limit on the number of storable tasks.
2. Web Platform Constraints
   1. Internet access is required for all access and functionality.
   2. Users must understand how to connect to the internet.
   3. The web browser must support the Google Polymer framework.

## 2.5 Assumptions and Dependencies

Bounce relies on the following assumptions about the user. Without these assumptions, users will most likely find the application unusable. These assumptions are divided into mobile and web platform sections.

1. Mobile Assumptions
   1. The user must be familiar with operating their mobile device.
   2. The user must know how to install applications from the Google Play Store.
   3. The user must have adequate space to install additional applications.
   4. The user must be familiar with the basic terminology of the supported languages.
2. Web Platform Assumptions
   1. The user must be familiar with operating their device.
   2. The user must know how to connect to the internet.
   3. The user must know how to connect to a website.
   4. The user must have an internet connection speed fast enough to make the application useable.

The user must be familiar with the basic terminology of the supported languages.

# 3. Functional Requirements

## 3.1 User Requirements

1. Users must be able to make an account locally.
2. Users must be able to make an account with their existing Facebook or Google accounts.
3. Users must be able to log in using their local account, Facebook account, or Google account.
4. Users must be able to log into their account on any of the available platforms.
5. Users must be able to create a task.
6. Users must be able to modify their tasks (Name, Times, Dates, etc.)
7. Users must be able to delete individual tasks.
8. Users must be able to create a list.
9. Users must be able to modify their lists(Name, Colors, Tasks, Default info, etc.)
10. Users must be able to delete a list.
11. Users must be able to turn off location based reminder.
12. Users must be able to connect tasks to be completed in a specified order.
13. Users must be able to rearrange connected tasks.
14. Users must be able to disconnect connected tasks.
15. Users must be able to set reminders for tasks as well as have deadlines set.
16. Users must be able add individual tasks to their mobile calendars from the application.
17. Users must be able to “Bounce” or move a task which they do not wish to complete to another day quickly.
18. Users must be able to change the default amount of time that a “Bounce” will add to a deadline.

## 3.2 System Requirements

1. Tasks must have default information added depending on the list it is created in.
2. Lists must assign default information to the tasks created in those lists.
3. Users must be warned that when deleting a list, the tasks under the list will be deleted.
4. All changes made to one platform must be updated to each platform with the Bounce cloud.
5. Users must be able to look at a Calendar overview of their tasks for the month.
6. The Calendar overview must colored in a way to show users which days will be busier than others.
7. Bounce must be able to notify users if they are locationally within range to complete certain tasks.
8. Tasks that are “connected” will only be displayed if it is the first task or the previous task had been completed.
9. Data shall be encrypted when it is routed between users and the server.
10. Data shall be decrypted when inside the Bounce backend to increase speed and efficiency.
11. Every server in the backend shall run a version of CentOS Linux greater than or equal to version 6.5 (to ensure that the system is up-to-date on the latest security patches).
12. All servers will use SELinux. It shall be set to enforcing on all of the servers. All actions on each server will be monitored, logged, and stored.

# 

# 

# 4. Non-functional Requirements

## 4.1 Interface Requirements

1. The interface design must adhere to Android standardized guidelines (e.g. Material Design).
2. The interface must be simple to use (such that it does not require training).
3. The interface must not suffer UI thread delays while processing.
4. The interface must make efficient use of screen real estate.
5. The hardware of the server should support thousands of queries a minute.

## 4.2 Software Requirements

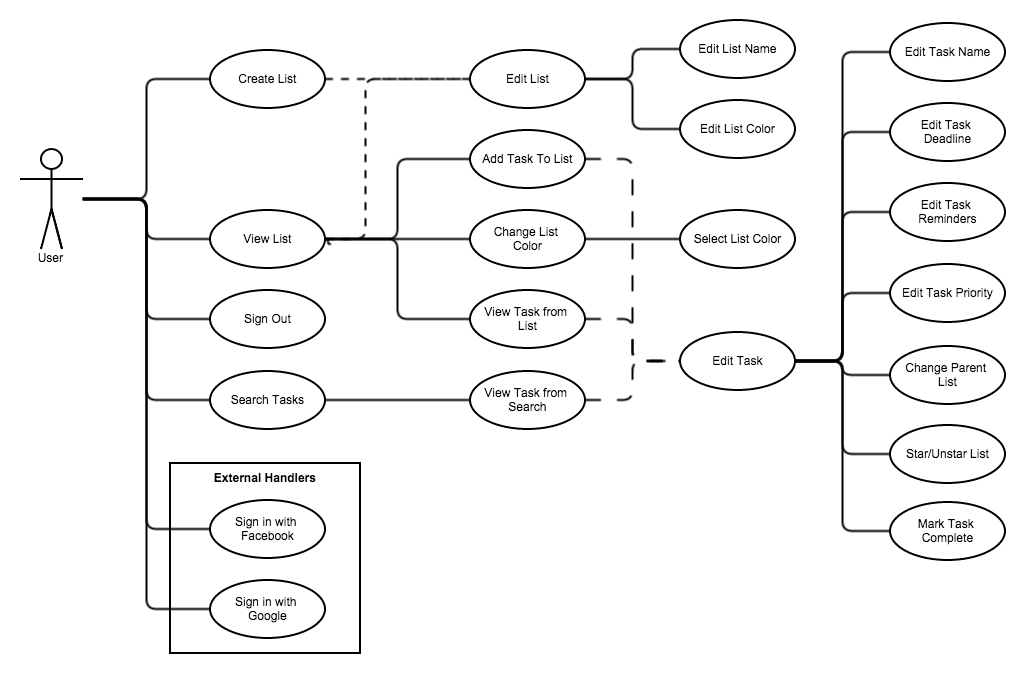
1. The software must be stable (e.g., not crash as result of operation).
2. The software must suppress any bugs or glitches in a manner that the user is not aware they occurred.
3. The software must be closed source.
4. The software must use methods and algorithms that will scale well for the number of lists, tasks, and users.
5. The software must operate in a manner that will not require maintenance.
6. The software must be compatible with the current versions of it’s platform, and have reasonable backwards compatibility (e.g., Android app must be compatible with Android 5.0, and back as far as Android 4.1).

## 4.3 User Requirements

1. The user must be familiar with standard Android interfaces.
2. The user must be familiar with simple terminology of supported language.
3. The user must be familiar with basic internet usage and terminology.

# 5. System Architecture & Models

## 5.1 Use Case Diagram

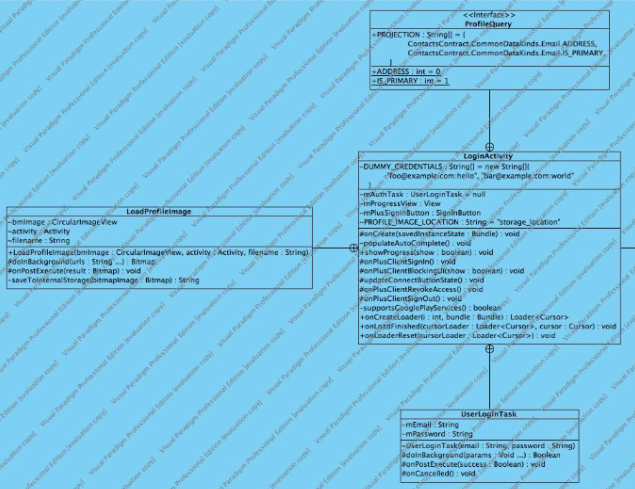
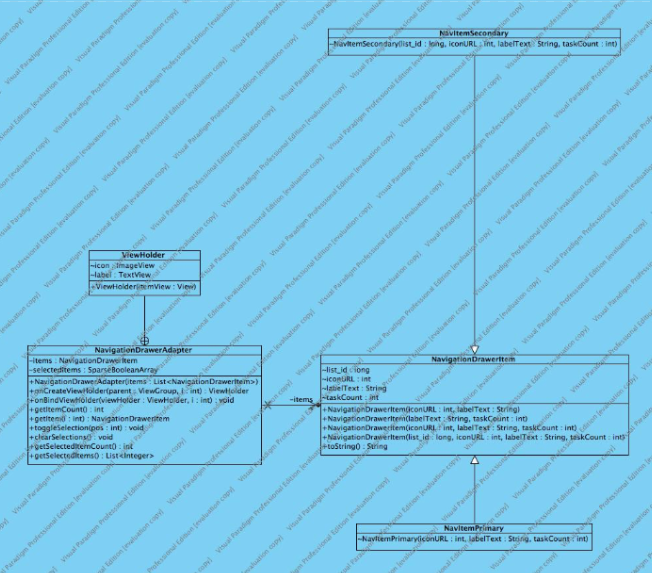
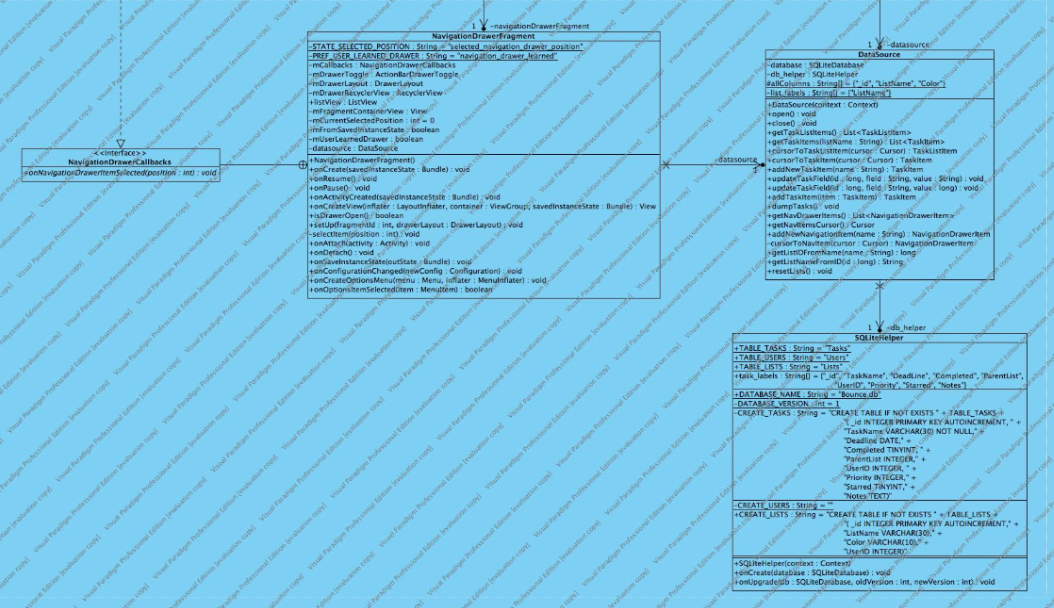
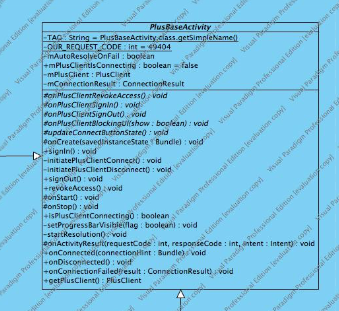
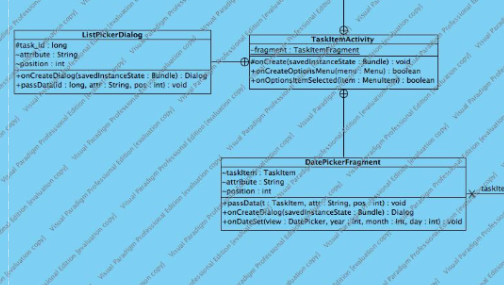
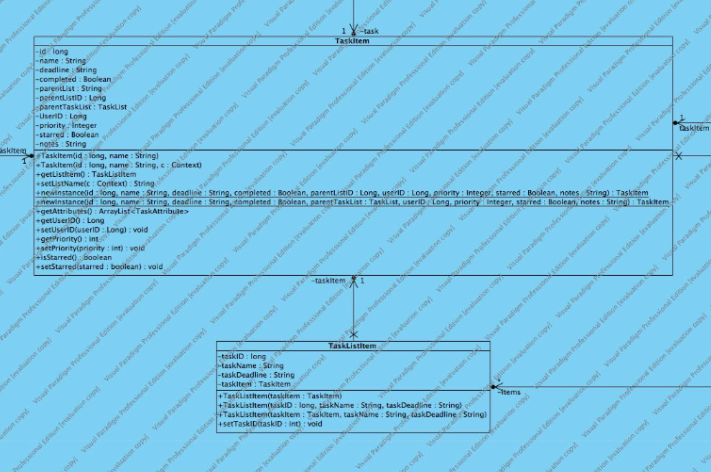
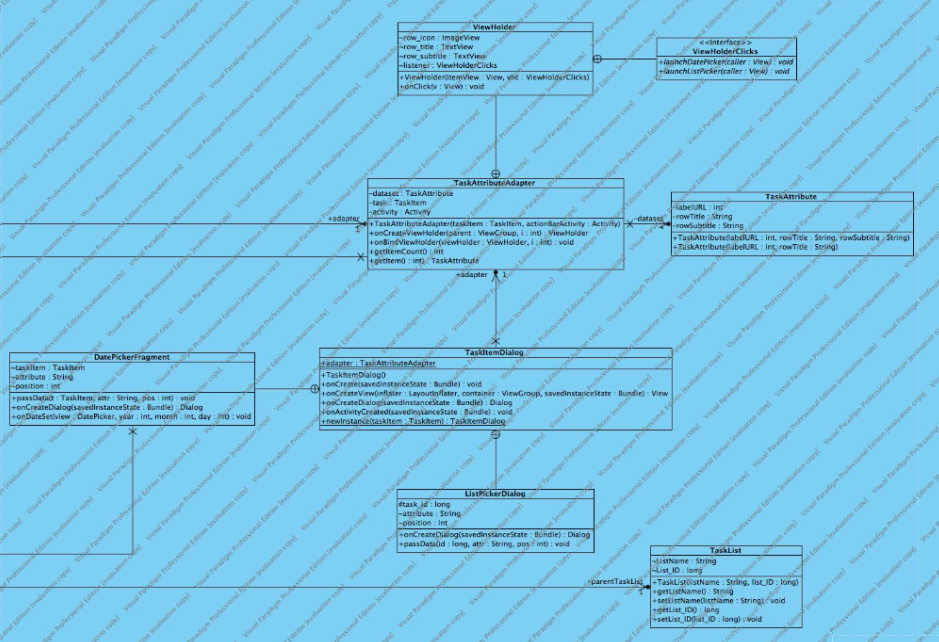


## 

## 

## 5.2 Class Diagram

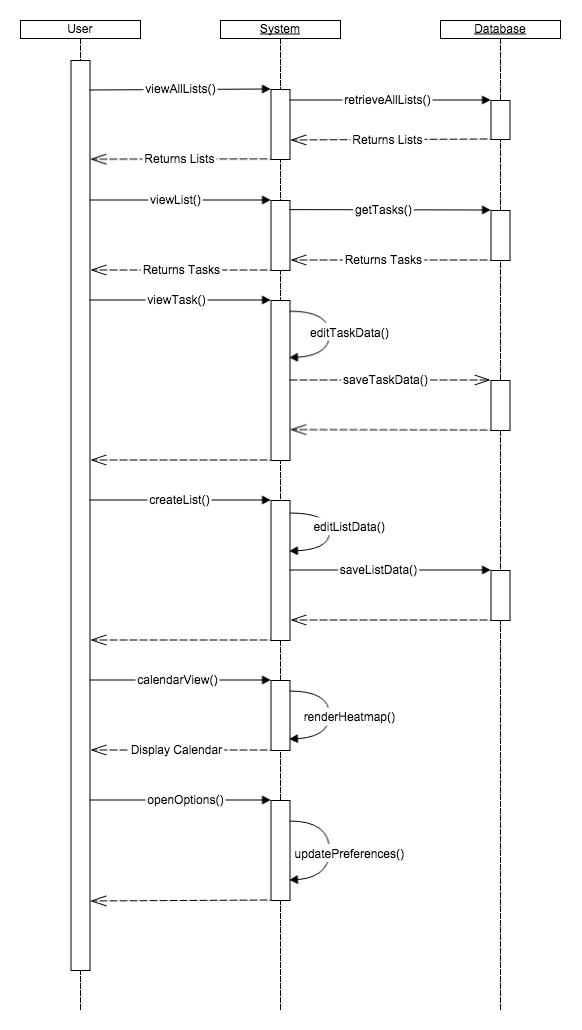
The following diagrams are from the Android mobile platform.



## 

## 

## 5.3 Sequence Diagram

The following diagram is a heavily abstracted representation of the system. 

# 6. Appendices

The following are screenshots of views from the application.

