FeastVite

Software Requirements Specification

1.0

06/13/2019

Pradeep Kolli

Sambi Reddy Chanimella

Pranathi Mothe

Srikar Patle

Durga Susmitha Kotyada

Satyakanth Kolakani

Submitted in partial fulfilment

Of the requirements of

CSIS 44-691 Graduate Directed Project 1

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 6/13/19 | 1.0 | Chanimella Sambireddy | Initial draft of the requirements. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Document Approval**

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
| P.K | Pradeep Kolli | Team Lead | 6/13/19 |
| D.K | Durga Susmitha Kotyada | Business Analyst | 6/13/19 |
|  |  |  |  |

Table of Contents

[1. Introduction 5](bookmark://_Toc12812534#_Toc12812534)

[1.1 Purpose 5](bookmark://_Toc12812535#_Toc12812535)

[1.2. Scope 5](bookmark://_Toc12812536#_Toc12812536)

[1.3. Definitions, Acronyms, and Abbreviations 5](bookmark://_Toc12812537#_Toc12812537)

[1.4. Overview 5](bookmark://_Toc12812538#_Toc12812538)

[2. General Description 6](bookmark://_Toc12812539#_Toc12812539)

[2.1. Product Perspective: 6](bookmark://_Toc12812540#_Toc12812540)

[2.2. Product Functions: 6](bookmark://_Toc12812541#_Toc12812541)

[2.3. User Characteristics: 6](bookmark://_Toc12812542#_Toc12812542)

[2.4 General Constraints: 6](bookmark://_Toc12812543#_Toc12812543)

[2.5 Assumptions and Dependencies: 6](bookmark://_Toc12812544#_Toc12812544)

[3. Specific Requirements 7](bookmark://_Toc12812545#_Toc12812545)

[3.1 External Interface Requirements 7](bookmark://_Toc12812546#_Toc12812546)

[3.1.1 User Interfaces 7](bookmark://_Toc12812547#_Toc12812547)

[3.1.2 Hardware Interfaces 7](bookmark://_Toc12812548#_Toc12812548)

[3.1.3 Software Interfaces 7](bookmark://_Toc12812549#_Toc12812549)

[3.1.4 Communications Interface 7](bookmark://_Toc12812550#_Toc12812550)

[3.2. Functional Requirements 7](bookmark://_Toc12812551#_Toc12812551)

[3.2.1. User Login 7](bookmark://_Toc12812552#_Toc12812552)

[3.2.2 Designing an invitation card 8](bookmark://_Toc12812553#_Toc12812553)

[3.2.3. Sending an invite 8](bookmark://_Toc12812554#_Toc12812554)

[3.2.4. User Accept/Deny Invitation 8](bookmark://_Toc12812555#_Toc12812555)

[3.2.5. User Logout 8](bookmark://_Toc12812556#_Toc12812556)

[3.3 Use Cases 9](bookmark://_Toc12812557#_Toc12812557)

[3.4 Class/Objects 9](bookmark://_Toc12812558#_Toc12812558)

[3.5 Non-Functional Requirements 9](bookmark://_Toc12812559#_Toc12812559)

[3.5.1. Performance 3.5.2. Reliability 3.5.3. Availability 3.5.4. Security 3.5.5. Portability 9](bookmark://_Toc12812560#_Toc12812560)

[3.6. Inverse Requirements 9](bookmark://_Toc12812561#_Toc12812561)

[3.7. Design Constraints 9](bookmark://_Toc12812562#_Toc12812562)

[3.8. Logical Database Requirements 9](bookmark://_Toc12812563#_Toc12812563)

[3.9. Other Requirements 9](bookmark://_Toc12812564#_Toc12812564)

[3.10. Use Case Diagrams 18](bookmark://_Toc12812567#_Toc12812567)

[4. Design 18](bookmark://_Toc12812568#_Toc12812568)

[4.1. ER diagram 18](bookmark://_Toc12812569#_Toc12812569)

4.2. GUI

[4.2.1. Prototypes (for complete project) 9](bookmark://_Toc12812565#_Toc12812565)

[5. Analysis Models 19](bookmark://_Toc12812571#_Toc12812571)

[5.1. Data Flow Diagram 19](bookmark://_Toc12812572#_Toc12812572)

[5.2. Sequence Diagram 19](bookmark://_Toc12812573#_Toc12812573)

[6. Technical Manual 19](bookmark://_Toc12812574#_Toc12812574)

# 1. Introduction

## 1.1 Purpose

FeastVite brings people together to celebrate your most important life moments, from kids’ birthdays to happy hours. We make coming together face-to-face effortless and even more memorable for you and your loved ones every day.

## 1.2. Scope

Scope is the part of project planning that involves determining and documenting a list of specific project goals, deliverables, features, functions, tasks, deadlines, and ultimately costs. In other words, it is what needs to be achieved and the work that must be done to deliver a project.

## 1.3. Definitions, Acronyms, and Abbreviations

* + Tenant (or client): A business entity using the services and accepting some sort of service level agreement.
  + Consumer (or customer or end user): Someone who consumes the services offered by the tenant. Also accepts the service level agreement through the tenant or has one of his own with the tenant.
  + Service: A means of delivering value to customers by facilitating outcomes customers want to achieve without the ownership of specific costs and risks.
  + User: Someone who interacts with the mobile phone application
  + IOS: iphone operating system.

## 1.4. Overview

FeastVite is focused on online invitations and ecards. FeastVite allows event planners to create invitations, include gift, registry, and charitable options. When an invitation is sent, the users can accept or decline the invitation, therefore we maintain a count of people who are attending the event.

# 2. General Description

## 2.1. Product Perspective:

This system will consist of mobile application. The mobile application will be used to find invitations and view information about them. Since this is a data-centric product it will need somewhere to store the data. For that, a database will be used. The mobile application use the database to get data.

## 2.2. Product Functions:

The members will be able to register online with a login name and password. This information is verified with the information in the database and the member is appended.

The members will be able to add, modify or delete the information provided by them when they register.

## 2.3. User Characteristics:

The User can login and register with login name and password. They can view the invitations and send the invitations online.The user interface will be friendly enough to guide the user.

## 2.4 General Constraints:

SomeGeneral Constraints that a client expected from the developers are:

Results, Time frames, Resources, Activity performance.

## 2.5 Assumptions and Dependencies:

One assumption about the product is that it will always be used on mobile phones that have enough performance. Assumptions bring possibilities and some of them are: Project member’s availability, Project member’s performance, Project member’s skills, Budget limitations, Accuracy of the project schedule dates.

Dependencies:

A task dependency is a relationship between two tasks in which one task depends on the finish of another task in order to begin. Dependencies can be created between two or more tasks, tasks and tasks groups or between two or more task groups.

# 3. Specific Requirements

## 3.1 External Interface Requirements

The member has to register using a form provided on the app. The user can input data with the help of the keyboard wherever necessary. The package provides pull down menus from which the user can select and links and icons to navigate among the app.

### 3.1.1 User Interfaces

A first-time user of the mobile application should see the log-in page when he/she opens the application, If the user has not registered, he/she should be able to do that on the log-in page. If the user is not a first-time user, he/she should be able to see the search page directly when the application is opened, Here the user chooses the type of search he/she wants to conduct. Every user should have a profile page where they can edit their e-mail address, phone number and password. Once the user logs in he/she can use the preview invitation or can create, edit or upload in his own invitations.

### 3.1.2 Hardware Interfaces

Since the mobile application don’t have any designated hardware, it does not have any direct hardware interfaces.

### 3.1.3 Software Interfaces

The communication between the database and the mobile application consists of only reading operations.

### 3.1.4 Communications Interface

The communication between the different parts of the system is important since they depend on each other. However, in what way the communication is achieved is not important for the system and is therefore handled by the underlying operating system for the mobile application.

## 3.2. Functional Requirements

### 3.2.1. User Login

User needs to Login into the application to use it.

#### a. Registration / Signup

User needs to register if he/she is using the system for the first time.

User will have to use a valid Email Id to register into the application.

User will have to use a valid password to register into the application.

Password should be 10 characters long.

Password should contain uppercase letters.

Password should contain lowercase letters.

Password should contain alphanumeric.

Password should contain special characters.

#### b. Login

User will be allowed to login only if the user has a valid combination of email ID and password.

#### c. Forgot Password

User will be allowed to reset the password by sending the reset link.

### 3.2.2 Designing an invitation card

The user will be able to choose from a variety of templates.

The user can add details about the venue, time, and other miscellaneous details.

The user will be able to preview the invitation card.

The user will be allowed to edit the invitation card at any time.

The user can save the invitation card as a draft for future use.

The user can upload the picture as well in invitation card.

### 3.2.3. Sending an invite

The host will be provided with provision to send the invite to other users.

The invitation will be sent to the other users based on the email Id’s we provide in an input field.

### 3.2.4. User Accept/Deny Invitation

The invitation sent by the host will send a mail with a unique link to the attendee. On click of the link the user will be redirected to the app if he is a user and he will be able to accept or deny the invitation.

We expect the user to be logged in and have the application installed on the phone.

If the attendee is not a user of the application, he will be sent an invite to download the app from the Appstore.

### 3.2.5. User Logout

The user will be able to logout from the application.

## 3.3 Use Cases

## 3.4 Class/Objects

## 3.5 Non-Functional Requirements

### 3.5.1. Performance 3.5.2. Reliability 3.5.3. Availability 3.5.4. Security 3.5.5. Portability

## 3.6. Inverse Requirements

## 3.7. Design Constraints

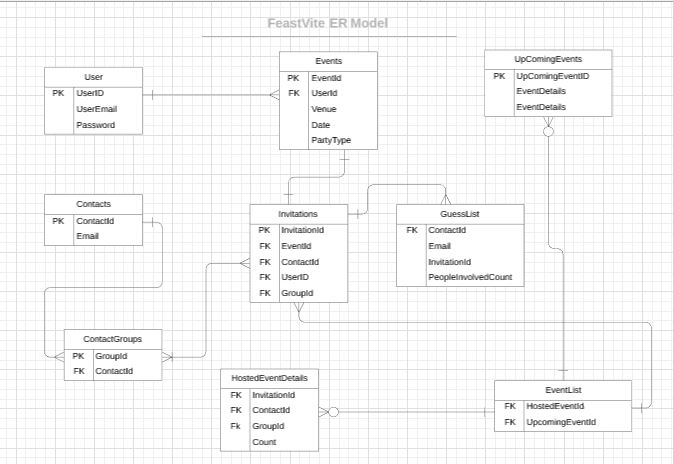
## 3.8. Logical Database Requirements

## 3.9. Other Requirements

**3.10. Use Case Diagrams**

## 4. Design

## 4.1. ER diagram



As per the ER diagram, initially user needs to login with his/her credentials by giving the email id and password, at that time unique user id will be created which serves as a primary key.

Once the user has login, he can have any number of events which means a single user can create multiple events.

For an event, the user can create only one invitation. Once the invitation is created the user can invite multiple number of guests, therefore the Guestlist maintains the count of guest attending the event.

We also maintain the contacts and contact groups of the people which contains email ids, in order to send the invitations.

Hosted events are nothing but the events which are conducted earlier, therefore a user can have zero number of hosted events or multiple number of hosted events, similarly we also maintain the list of upcoming events where a user can have zero number of upcoming events or multiple number of upcoming events.

We created dimension model, is a database structure that is optimized for online queries and increases performance, it is compromised fact tables and dimension tables, invitations entity act as a fact table and remaining tables are dimension table

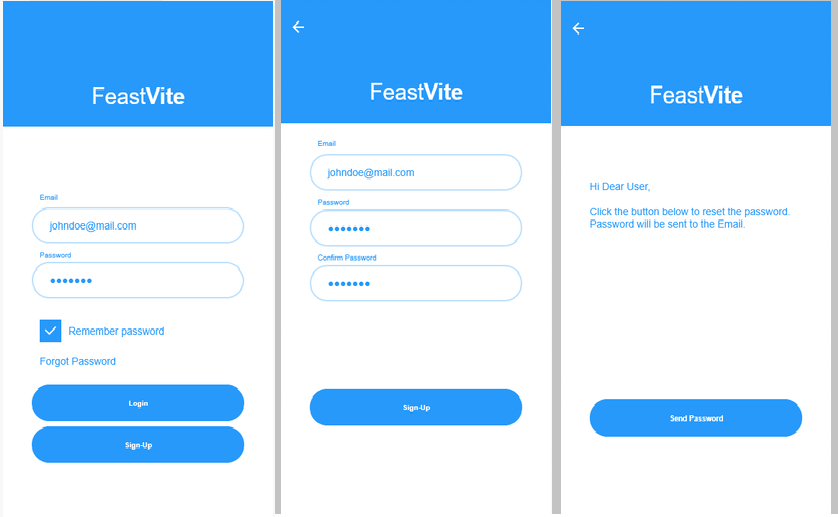
Fact table, contains numerical values, and dimension tables are used to describe dimensions they contain dimension keys, values and attributes.

## 4.2. GUI

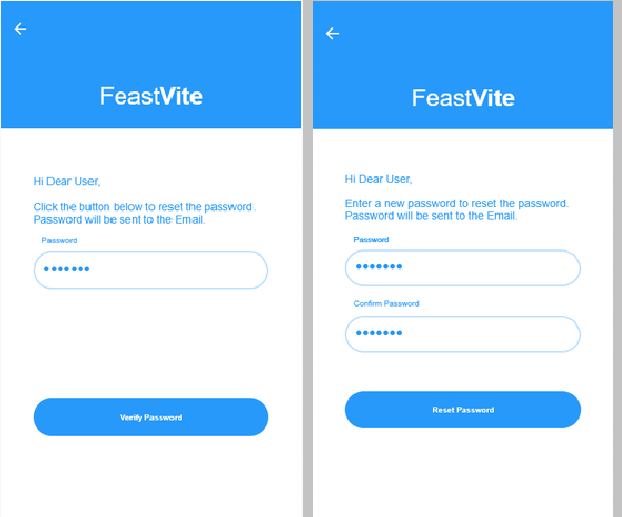
## 4.2.1 Prototypes (for complete project)

1. **Login Functionality:**

Login Sign Up Send Temporary Password

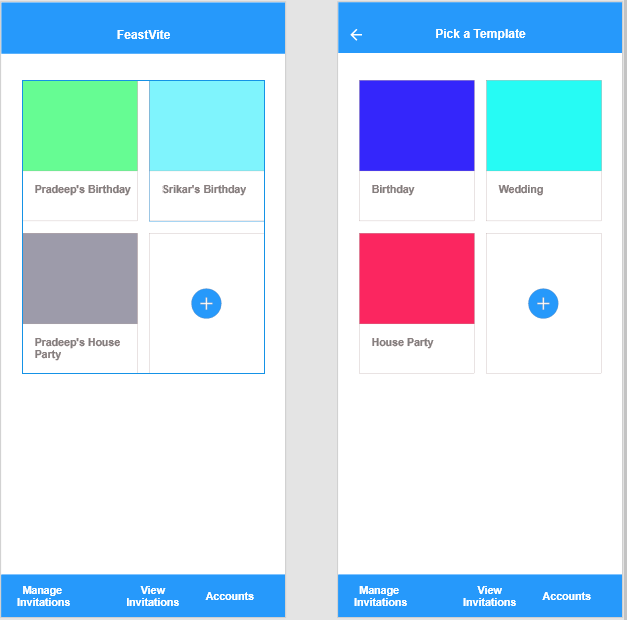


Confirm Password Reset Password



1. **Manage Templates:**

Home (view existing hosted invitations) Pick from existing templates

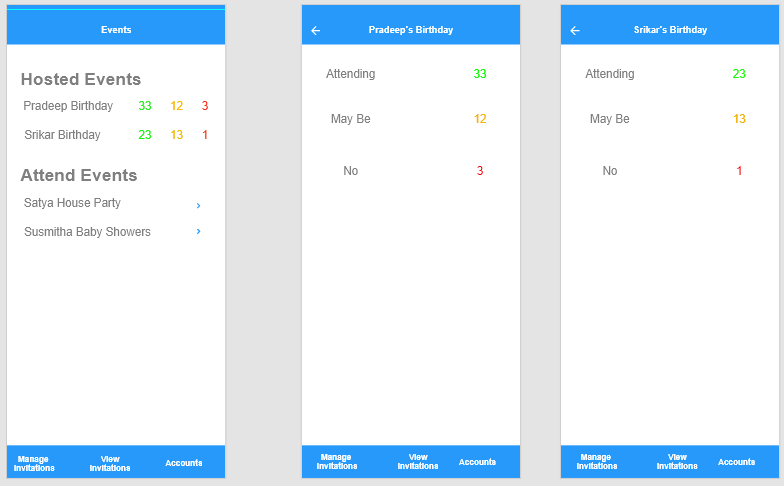


Edit a template Edit a template – 1 New Template

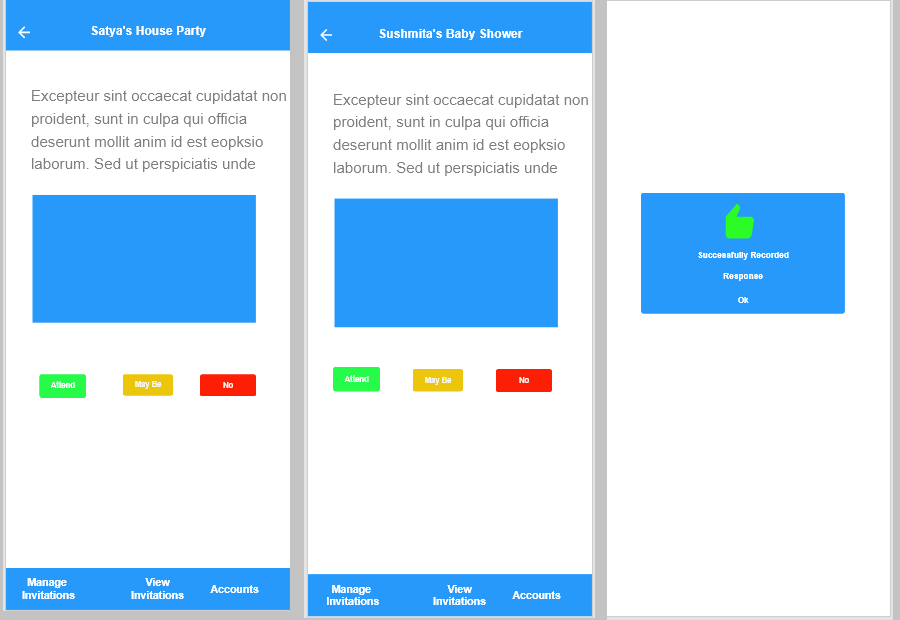
## 

1. **View Invitations:**

View Invitations Hosted Event 1 Hosted Event2 Details Details

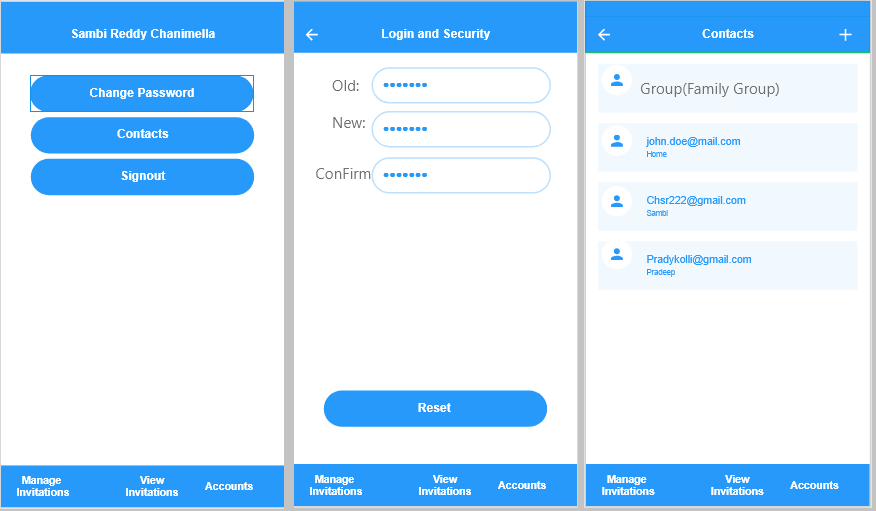


To Attend Event 1 Details To Attend Event 2 Details Confirmation Pop Up



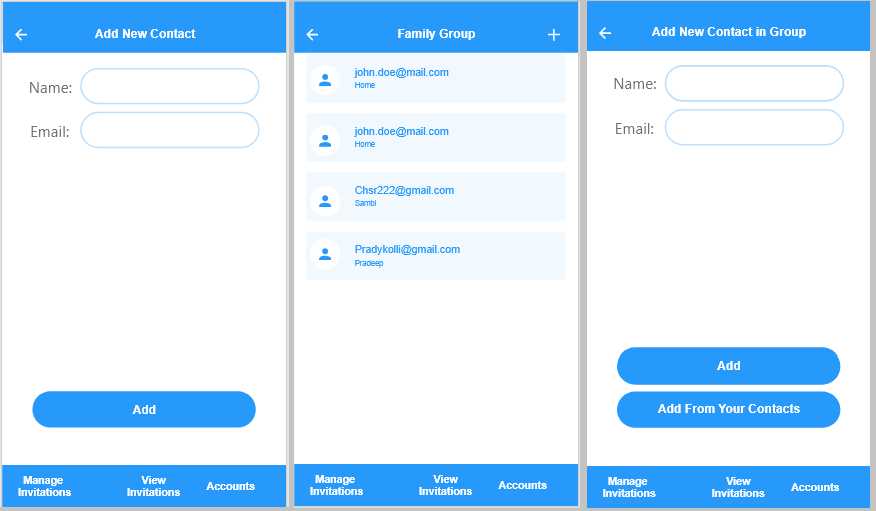
1. **Account Settings:**

Account Login and Security Contacts



## 

Add New Contact Groups Add New Contact In Group



## 5. Analysis Models

## 5.1. Data Flow Diagram

## 5.2. Sequence Diagram

## 6. Technical Manual

**6.1. Introduction:**

**6.1.1. Purpose of Document:**

The purpose of the document is to provide the technical details of software components that this project is made of.

**6.1.2 Intended Audience**

Project Customer

Project Manager

Project Team Members

Users

**6.2 FeastVite IOS Application**

**6.2.1 Introduction**

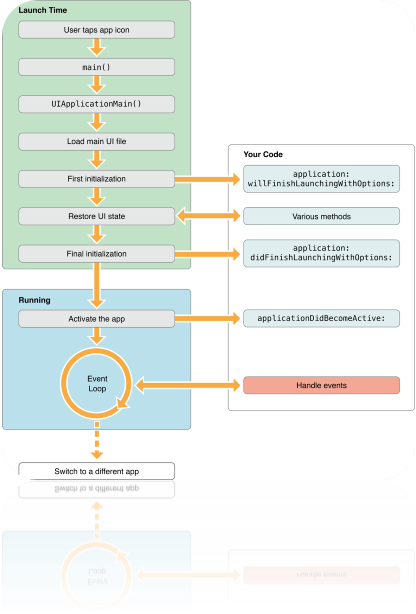
FeastVite is focused on online invitations and ecards. FeastVite allows event planners to create invitations, include gift, registry, and charitable options. When an invitation is sent, the users can accept or decline the invitation, therefore we maintain a count of people who are attending the event.

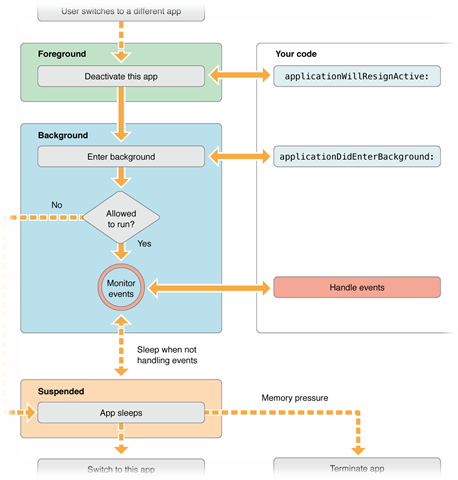
**6.2.2 IOS Operating System**

IOS – runs the iphone, ipad and ipad Touch devices.

**6.2.3 IOS Application Life Cycle**

Application life cycle is very important to understand for all developers.





When the user taps application icon on iphone/ipad springboard part of the that operates the home screen of iOS, launches your app. Your app and the shared libraries it need to execute, will be loaded into the memory, while springboard animates your app’s launch screen. eventually your app begins execution and application delegate receives the notifications.

Every IOS application, on the device will transit into several states, and there are:

6.2.3.1. Suspended:

An application is in the background, but it is not executing the code. The system moves the application to this state automatically and does not notify. In case of low memory, the system may abrupt suspended application without notice to make free space for the foreground application. Usually after 5 secs spent in the background, apps will transition to Suspend state, but we can extend the time if app needs.

6.2.3.2. Background:

An application is running in the background and executing the code. Freshly launching apps directly enter In-Active state and then to Active state. Apps that are suspended, will come back to this background state, and then transition to In-Active → Active states. In addition, an application being launched directly into the background enters this state instead of the inactive state.

6.2.3.3. Foreground (Active):

An application is running in the Foreground and receiving the events. This is the normal mode for the Foreground apps. The only way to go to or from the Active state is through the Inactive state. User normally interacts with UI and can see the response/result for user actions.

6.2.3.4. Running (Inactive):

An application is running in the Foreground but is not receiving any events. This could happen in case a Call or Message is received. An application could also stay in this state while in transition to a different state. In this State, we cannot interact with app’s UI.

6.2.3.5. UIApplication:

Object defines some methods which are called or will be responded to some of the above states which are most important, to let us work on those transition states regarding our app functionalities.

**6.2.4 Once successfully launched application, AppDelegate will initiate the few methods:**

6.2.4.1 application:willFinishLaunchingWithOptions :

This method is called after your application has been launched successfully. It is the first method from our app delegate, which will be called. You can execute your code if the launch was successful.

6.2.4.2 application:didFinishLaunchingWithOptions :

This method is called before the app’s window is displayed. You can finalize your interface and can provide the root View Controller to the window.

6.2.4.3. applicationDidBecomeActive :

This method is either called to let your app know that it moved from the inactive to active state or your app was launched by the user or the system or in case user ignores an interruption (such as an incoming phone call or SMS message) that sent the application temporarily to the inactive state. You should use this method to restart any tasks that were paused (or not yet started) while the app was inactive.

6.2.4.4. applicationWillResignActive :

This method is called to let your app know that it is about to move from active to inactive state. This can happen in case of any interruptions (such as an incoming phone call or SMS message or Calendar alerts) or when the user quits the app. You should use this method to pause any ongoing tasks or disable timers etc.

6.2.4.5. applicationDidEnterBackground :

This method is called to let app know that it is not running in the foreground. You have approximately five seconds to perform any tasks and return. In case you need additional time, you can request additional execution time from the system by calling [beginBackgroundTask(expirationHandler:)](https://developer.apple.com/documentation/uikit/uiapplication/1623031-beginbackgroundtask). If the method does not return before time runs out your app is terminated and purged from memory.

6.2.4.6. applicationWillEnterForeground :

This method is called as a part of the transition from the background to the active state. You should use this to undo any change you made to your app upon entering the background. applicationDidBecomeActive method is called soon after this method has finished its execution which then moves the app from the inactive to the active state.

6.2.4.7. applicationWillTerminate :

This method is called to let you know that your app is about to terminate. You should use this method to perform any final clean-up task. You have approximately five seconds to perform any tasks and return. If the method does not return before time expires, the system may kill the process altogether. This method may be called in situations where the app is running in the background (not suspended) and the system needs to terminate it for some reason. There are some cases when applicationWillTerminate won’t be called before app termination. For example, the system will not call applicationWillTerminate when the device reboots.

**6.3 Login page**

This page has username and password text fields. When user enters username and password verifies user credentials.

3.1.1 The Login authorization of user is done using Back-endless userService. We have UI validations for username and password and will displays proper feedback to user.

3.1.2 Forgot password functionality is implemented using Back-endless in-built e-mail services. A temporary password is sent to the user through E-mail.

**6.4 Registration**

User must register with required details. When user clicks on Registration Button, if all the details given are correct then feedback message user is sent “Registration Successful”. If the details are wrong, an error message is sent to user.

All the Valid details are stored to Back-endless database when the user is successfully registered.

**6.5 Forgot password page**

Incase user forgets his/her password we provide the functionality to retrieve the password by sending the user an Email to his registered mailbox.

A temporary password is sent to the user’s email Id. User can either update to a new password or skip or start using the same password that is sent.

We would recommend the user to change password once in a while.

**6.6 Remember password**

Since this is an app, user would prefer to log in once and stay logged in unless purposefully logged out.

Remember password switch should record the state and take the user to the home page once he is logged in.

**User Functionality**

**6.7 Manage invitations**

This tab keeps track of all the existing invitations and an option to create a new invitation.

**6.7.1 Edit previous invitation**

The user can edit a previous invitation and send the edited invitation to the other people.

**6.7.2 Create a new invitation**

The other way is to create a new invitation and send it to the intended people or group of people.

There are two ways of doing this. We could pick from predefined templates and start adding details of the event like event type, date, time, venue, special message etc.

We could upload a new pic and make it a background of the template. The rest of the process remains same.

**6.8 View invitations**

This is a tabular form of showing all the invitations. The invitations we received and the invitations we have sent will be listed in two different sections of this page.

**6.8.1 Hosted Events**

The events we are hosting and the invitations we have sent will be listed in this section. On click of one invitation, it shows the count of the users that coming, may be coming and will not be coming to the event so that it will be easy to keep track of the things.

**6.8.2 Invited Events**

The events that we are invited to will be listed in this section. On click of an invitation, we will be shown preview of this event. Apart from this we also have an option to notify the host if we could make it to the event or not by clicking on the buttons provided.

**6.9 Accounts**

This tab will deal with account settings and maintaining contact groups and contacts.

**6.9.1 Change password**

This section lets you update the password and use a different password from the next time you try to log in.

**6.9.2 Manage contacts**

This section will help you to manage contacts such as saving, updating and deleting the contacts. And also we have the option to create groups if required.

**6.10 Database Functionality**

Database that we have used for this application is “Back-endless” which is of MBaas type.

**3.8.1 Connection to Back-endless:**

3.8.1.1 Register & create an app in Back-endless. After created every app has API-Key in Back-endless (can see API-Key in “Settings”).

3.8.1.2 Connect app to back-endless using API-Key (embed this API-key in APP DELIGATE file).

3.8.1.3 Create a back-endless instance, using this back-endless perform CRUD operations.

Using back-endless user service, we have implemented login, registration and forgot password functionalities.

## 