**Software Requirements Specification**

**for**

Sprout

**Version 2.0 approved**

**Prepared by Enoch Lin, Matthew Loewen, Andrew Piccirillo**

**Sprout Inc.**

**2/26/2019**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1. Introduction 3**

1.1 Purpose 3

1.2 Document Conventions 3

1.3 Intended Audience and Reading Suggestions 3

1.4 Product Scope 4

1.5 References 4

**2. Overall Description 4**

2.1 Product Perspective 4

2.2 Product Functions 5

2.3 User Classes and Characteristics 6

2.4 Operating Environment 6

2.5 Design and Implementation Constraints 6

2.6 User Documentation 6

**3. External Interface Requirements 7**

3.1 User Interfaces 7

3.2 Hardware Interfaces 7

3.3 Software Interfaces 8

3.4 Communications Interfaces 8

**4. System Features 8**

4.1 Matching 8

4.2 Ice Breaker Interaction 8

**5. Other Nonfunctional Requirements 9**

5.1 Performance Requirements 9

5.2 Safety Requirements 9

5.3 Security Requirements 10

5.4 Software Quality Attributes 10

5.5 Business Rules 10

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Sprout Inc. | 3/17/2019 | Updated functional and nonfunctional requirements and table of contents. | 2.0 |
|  |  |  |  |

# **Introduction**

## **Purpose**

The purpose of this document is to predict and describe the functional, non-functional, system, and user requirements for the “Sprout” dating application. In addition, user stories and a user interface mockup will be included. The scope of this document will cover the entire application.

## **Document Conventions**

Every requirement statement in this document will be assigned its own priority.

Below is a table defining standards that are used in this document.

|  |  |
| --- | --- |
| Term | Definition |
| SRS | Software Requirements Specification |
| User | Person who downloaded and uses the application. |
| Administrator | Person with elevated permissions that can manage users and other aspects of the application. |
| Match | Two users agreeing to start a dialog with each other. |
| Like | One user agreeing to open to dialog with another. |
| Dislike | One user restricting dialog with another |
| Icebreaker | A user’s opening question that can be answered by those who like the user’s profile. |
| UI | The applications user interface |

## **Intended Audience and Reading Suggestions**

This document is intended for developers, project managers, users, and documentation writers. The rest of this SRS document is structured based off the IEEE Recommended Practice for Software Requirements Specifications [1]. It is organized starting with the overall description of the application, to external interface requirements, system features, nonfunctional requirements, and appendices. The suggested sequence of reading this document is in order through each section. However, different readers may find it useful to skip to specific sections based on their informational needs.

## **Product Scope**

The Sprout dating application is designed with the intention of users to match with others and start relationships. A detailed description of the product including goals and similar applications can be found in the Project Description and User Story documents listed in the References section directly below [2, 3].

## **References**

[1] IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications”, October 20, 1998.

[2] Sprout Inc., “Project Description Document,” February 6, 2019

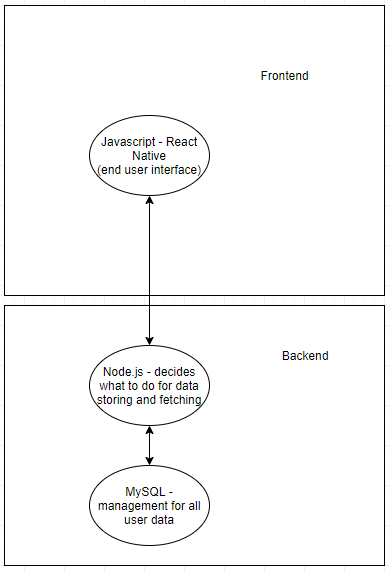
[3] Sprout Inc., “User Stories Document,” February 6, 2019

# **Overall Description**

## **Product Perspective**

The Sprout application will be a self-contained product. This product aims to be used on web, IOS and Android devices. Through the use of React Native (a Javascript library for building user interfaces compatible with all major devices) only one code base will be required for all end user platforms. The application will be developed with Javascript for the frontend, Node.js for the backend, and MySQL for the database. While users use the Sprout application they will be navigating the React Native front end. This frontend will make requests for data to the Node.js backend where the requests will be routed and validated. Once these requests are validated the Node.js backend sends a request for data to the MySQL database which retrieves or stores the data from the request (see visual 1).

*Visual 1 - Software architecture for the Sprout Application*

**

## **Product Functions**

With this application, users will only need to be able to perform several actions. Each of these actions are critical to a user's use of the application. If a user is not able to perform any of these actions they will not be able to use the application as intended. The required actions are:

* Account management
  + Create account
  + Delete account
  + Update account information
  + Customise profile
* Searching for matches
  + Ability to view other user's profile descriptions
  + Ability to respond to other users profiles
  + Ability to match or deny users that want to match with me
  + Ability to interact with a users profile
* Once matched with users
  + Ability to see other users full profile
  + Ability to message back and forth

## **User Classes and Characteristics**

Initial launch of this application will just be to UMaine students. With this in mind we are expecting technologically inclined users. All of these users will be considered the “standard user” this standard user will have permissions to just modify their own profile, view other users profiles, and message with their own matches. From here we expect these users to fall into one of two categories: someone who uses the app daily, and someone who uses the app a couple of times a week. In this case the app would be tailor in peak usage times to priorite the less frequent users to instill a more pleasurable experience for them.

## **Operating Environment**

The system will have the same working code base across all platforms - in this case IOS and Android smartphones. The client side code (the app) will be available to users through their respective app stores. This app will communicate to the projects backend hosted within a Linux Ubuntu computer running version 18.04.2. This computer will being running node.js version 10.15.1 and a MySQL database running MySQL version 8.0. The node.js backend will be the backend acting as a restful api for the entire application. The app on end users devices will be able to make get and post requests.

## **Design and Implementation Constraints**

The development of this project is under scrutiny of several factors. This project has a tight deadline as well as no funding. The developers working on the project are actually paying to work on the project. Moral is low. For the timeline the project has until May 5th 2019 to be completed. In addition to this a major of the developers have not worked on a project of this size. A large portion of development for this project will be spent learning.

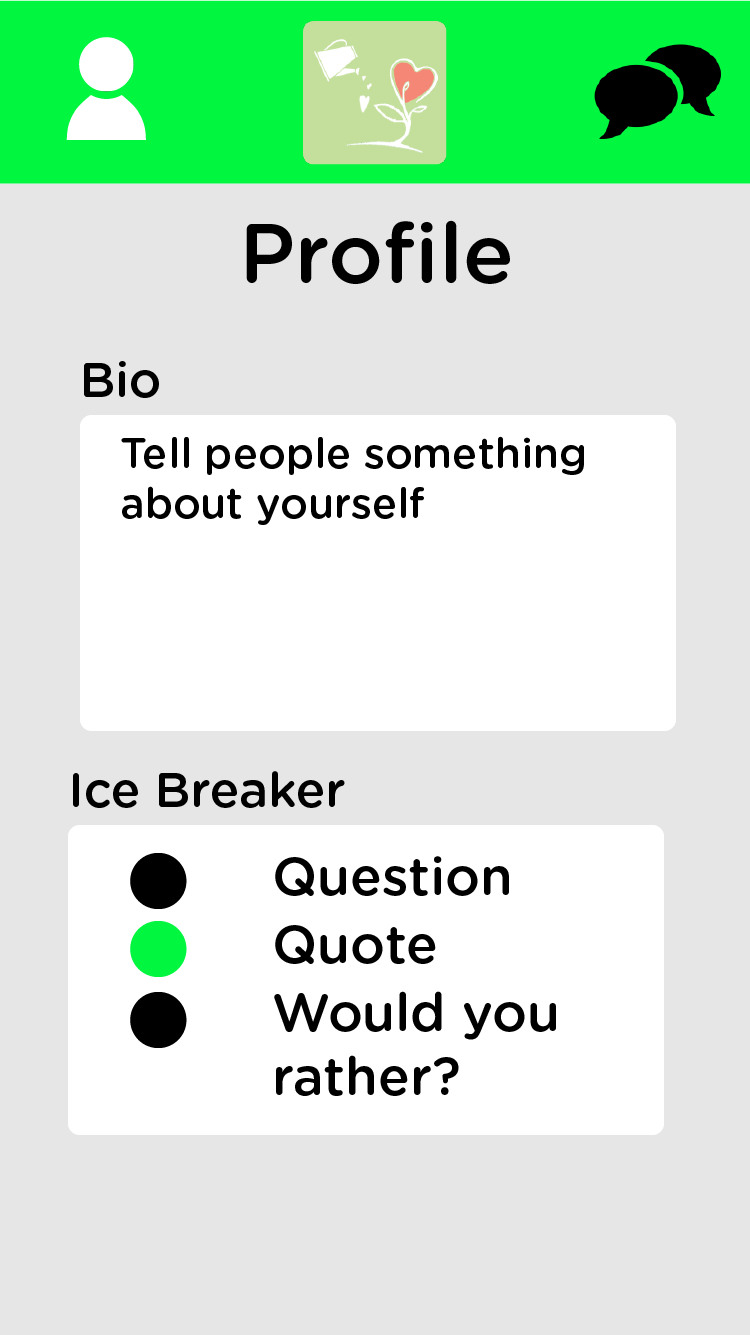
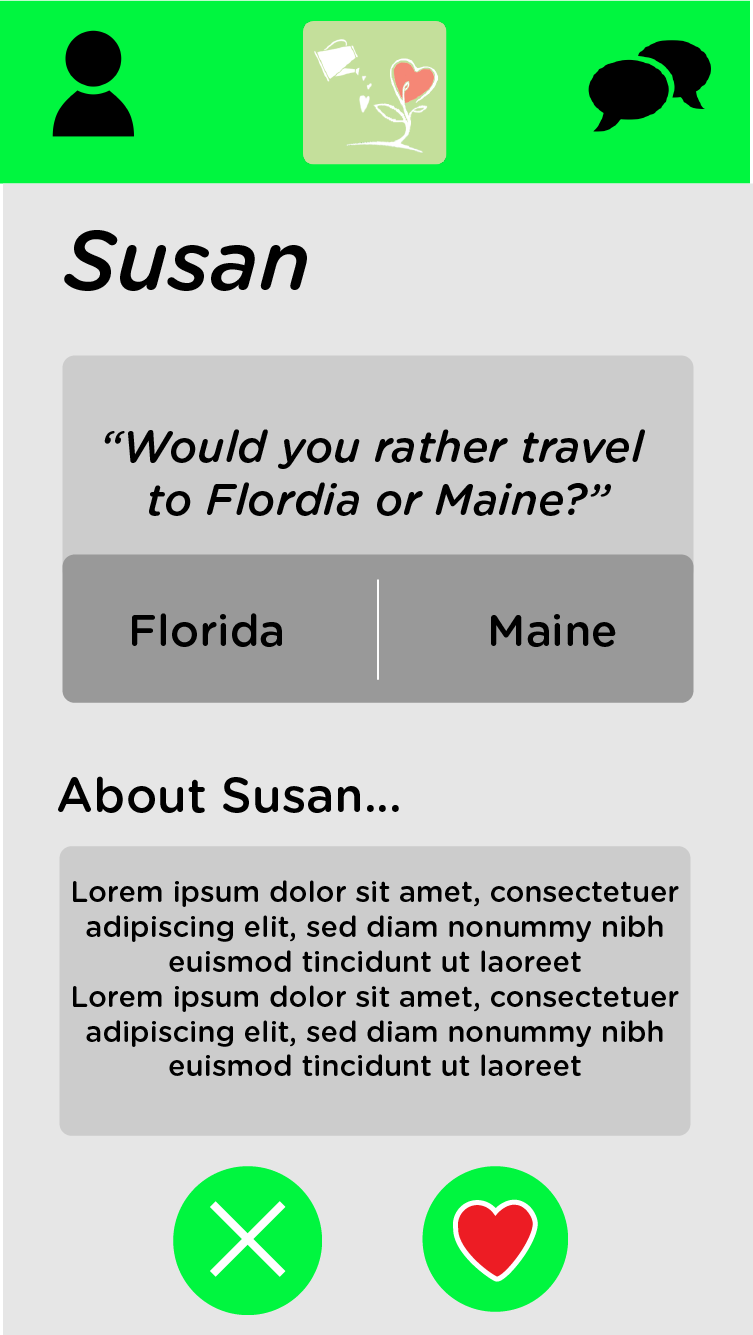
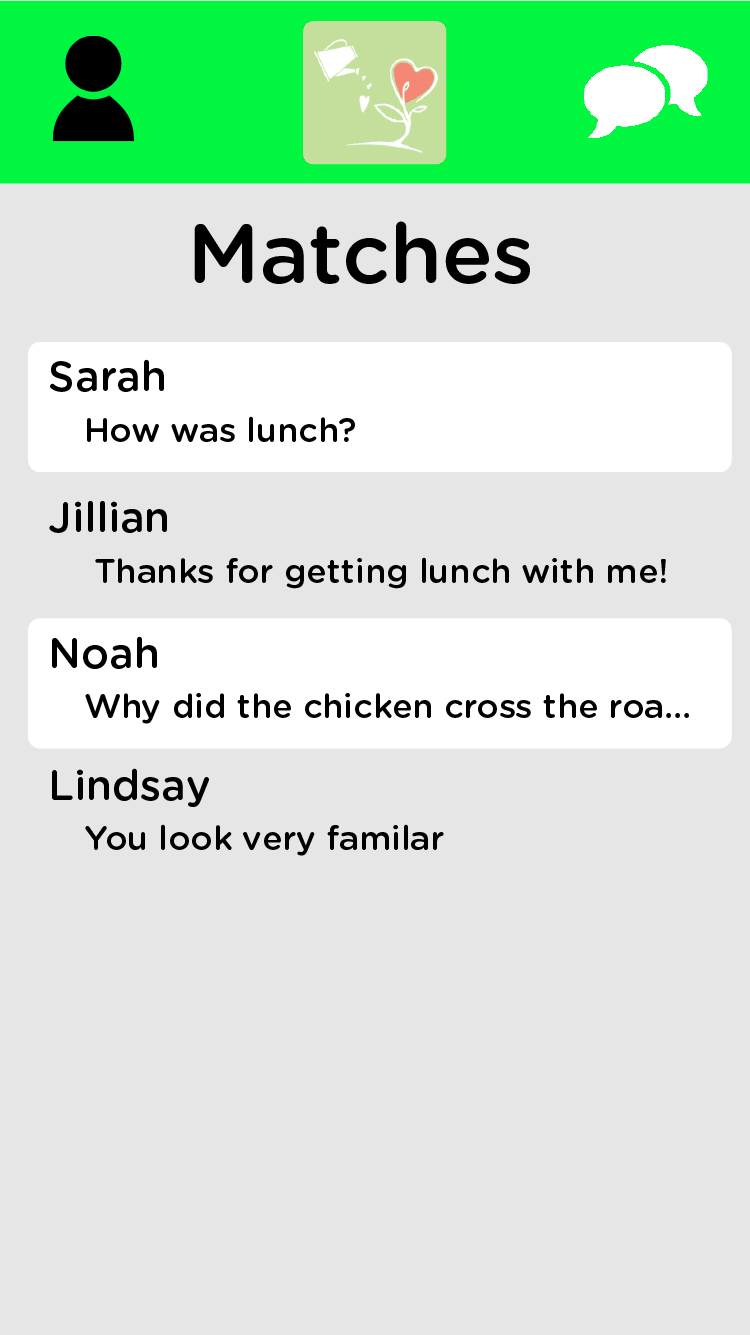
## **User Documentation**

Along with the final application, documentation video tutorials will be created to show users how to use the application if they so need the extra help. In addition to this a help setting will be put in the user settings page that user can click on to view text tutorials along with the video tutorials.

# **External Interface Requirements**

## **User Interfaces**

* The system shall allow users to switch among the Profile Screen, Match Screen and Message Screen.
* The system shall allow users to make changes to their profile.
  + The system shall allow users to change the background color of their profile.
  + The system shall allow users to add ice breaker questions to their profile.
* The system shall display other users’ profiles to the currently user one at a time.
* The system shall display one ice breaker question on one’s profile.
* The system shall display a list of people the user matched with.

## **Hardware Interfaces**

* The application shall be able to run on Android devices.
* The application shall be able to run on iOS devices.

## **Software Interfaces**

* The system shall be able to store user information.
* The system shall allow users to “like” a profile.
* The system shall allow users to “dislike” a profile.
* The system shall allow users to access other people’s detailed profile.
* The system shall allow users to message others they have matched with.

## **Communications Interfaces**

* The system shall allow users to sign up for the application through e-mail address.
* The system shall send users an e-mail to confirm registration.

# **System Features**

## Matching

4.1.1 Description and Priority

If two users “like”d each other, both of them should be notified. They should show up on each other’s match/message list. It is of high priority.

4.1.2 Stimulus/Response Sequences

1. User 1 “like”s user 2. 2. Check if user 1 exists in user 2’s “like”d list, if

yes, notify both users it’s a match

4.1.3 Functional Requirements

REQ-1: The system shall allow users to “like” a profile

REQ-2: The system shall allow users to “dislike” a profile

REQ-3: The system shall notify users when both profiles have “like”d each other.

## Ice Breaker Interaction

4.2.1 Description and Priority

When a user views the profile of another, the first thing they see is an interactable ice breaker question which helps people get to know each other at least a little bit before deciding whether they “like” or “dislike” someone. It is of high priority.

4.2.2 Stimulus/Response Sequences

1. User 1 select an answer on user 2’s profile 2. Display and inform user 1 what

user 2’s choice was.

4.2.3 Functional Requirements

REQ-1: The system shall allow users to add ice breaker questions to their profiles.

REQ-2: The system shall allow users to make changes to their existing ice breaker questions.

REQ-3: The system shall allow users to interact with others’ ice breaker questions.

REQ-4: The system shall provide suggested ice breaker questions.

# **Other Nonfunctional Requirements**

## **Performance Requirements**

The following nonfunctional requirements discuss how the Sprout application shall perform. The most critical requirements relate to speed, storage, and reliability.

5.1.1 The system shall run on Android and IOS.

5.1.2 The system’s response time for user “likes” must not exceed 3 seconds, 99% of times.

5.1.3 The system’s response time for user “dislikes” must not exceed 3 seconds, 99% of times.

5.1.4 The system’s response time for user “matches” must not exceed 3 seconds, 99% of times.

5.1.5 The system’s response time for user messages to be sent must not exceed 10 seconds, 99% of times.

5.1.6 The system shall support storing a maximum of 3 images on user profiles.

5.1.7 The system’s response time for changing user interface screens must not exceed 3 seconds, 99% of times.

5.1.8 The system shall take no longer than 5 minutes to notify two users that they have matched with each other.

5.1.9 The system shall be available to users 99% of the time.

## **Safety Requirements**

The following requirements discuss the safety of users of the Sprout application. The most critical requirements relate to removing user accounts with a history of harassment, disallowing account access to people under the age of 18, and allowing users to delete their own accounts.

5.2.1 The system shall not grant user accounts to people under the age of 18.

5.2.2 The system shall allow administrators to remove user accounts.

## **Security Requirements**

The following requirements discuss the security and privacy issues surrounding the Sprout application. The most critical security requirements relate to securely storing a user’s personal account information and login identity authentication.

5.3.1 The system shall protect the account username.

5.3.2 The system shall protect the account password.

5.3.3 The system shall use the user’s account name and password for identity authentication.

5.3.4 The system shall only authenticate a user if their account name and password matches the credential database record for it.

## **Software Quality Attributes**

The Sprout application is designed for the users. Therefore, every element of the user interface is carefully thought out and planned to maximize ease of use and minimize any human confusion. The following requirements relate to these software quality attributes.

5.4.1 The system shall have a UI that fills the 100% of the available screen space.

5.4.2 The system shall have UI buttons that are no less than 48x48 dp.

5.4.3 The system shall have a UI that uses the contrasting colors white, black, green, red, and gray.

5.4.4 The system shall have the company logo visible on all three user navigable screens.

## **Business Rules**

The two user classes for this application are users and administrators. Users have the basic functions needed operate the app for its desired purposes of creating and editing a profile, liking or disliking users, and messaging matched users. Administrators have elevated roles that allow user moderation. Moderation includes the deletion of user accounts that have been reported by users.