Functional Specification for CA326

By James Fallon and Alex O Neill

Table of contents

Introduction

1.1 Overview

This functional specification outlines our proposed socialising/activity organising web-app with the intention of meeting new people based on common interests. This specification will go in depth with the system requirements and the overall design of our web-app.

1.2 Scope

Socialise is an activity organising app that allows people or existing friends to come together. This web-app will allow users to create and Manage activities or “events”. Users will also be able to join activities organised by other people.
The main menu that the user will be greeted with upon sign-in is the map interface. This map is the centre of all functions of the web-app. It illustrates where the user is in relation to other users and other events that are either nearing capacity or in need of more participants. Any event once clicked upon, will display a small number of details regarding the event. If an event is set to private a request to join must be sent

to the host. You can also choose to contact the host directly to enquire about the details of the event.

Each event is nested within its own activity homepage. This will include a feed of updates, and also a list of participants. The list of participants is represented by a small profile picture, and once clicked on they expand to display the user's summarized bio. This may include their top interests, small description, a summary of previous events attended that they choose to share. It is a centre of communication between all the participants. This helps keep everyone on the same page when it comes to the events date and details.

2.1 Product Perspective

Our web-based app is self-contained. It is not a component of a larger system. All components of the system are located within the system. "Friend finding" apps are already available. Although there is no clear market leader. We believe this is because of poor design and also a lack of community and activity on these platforms. They have no USP which keeps bringing users back.

This is why we believe the market is ready for a web-app that brings people together over common interests. What makes our web-app different is that it then acts on this common interest, while other apps do not. You are able to create, plan and join events, which ensure that activities always have sufficient numbers to go ahead.

2.1.1 System Interfaces

1. Log in/sign up/forgot password Interface
    a. Create an account
    b. Log in to existing account
    c. Forgot password help page
2. Map-based Interface
    a. Identify where the user is in relation to Events and other small clusters of users
    b. May Interact with google maps API
3. Create Event Interface
    a. What type of event are you looking to create
        i. It will recommend based on users top specified interests
    b. How many people would you like to attend? (give range)
    c. Date/Time of event
    d. Any special requirements/details
    e. Public/private
4. Event feed Interface

    a. Displays the event name, date, time, location
    b. Displays a list of the participants
    c. Allows for communication between all participants
    d. Participants can share pictures of the event or chat amongst themselves
5. Friends list interface
    a. Display a list of the user's friends
    b. Allow chatting to a friend
6. Chat interface
    a. Allow for 2-way communication between friends
7. User Profile Interface
    a. Every user will have a profile picture and this will be displayed
    b. User profiles will display a bio. This is a short description of the individual
    c. Profiles will also contain a user's interests or hobbies
    d. They can also share previous activities that they enjoyed

2.1.3 Hardware Interfaces

For our web-app, it will be required to function on a mobile operating systems web browser. Along with the system must also work on desktop-class web browsers. This means the web-app must be flexible in design and responsive to multiple screen/window sizes. It must cater to a keyboard and mouse IO, as well as a touch-based interface.

2.1.4 Software Interfaces

We will be using nodeJS to create, store and recover passwords using a SQL Database within our javascript script.
Google Maps API will be used in our "main menu" to illustrate the locations of nearby events organised by the other users. A nearby event will be displayed as a "pin" on the map. The map will be centred on the user's location.

2.1.5 Communication Interfaces

We will use sockets to provide communication over a single TCP connection. Unlike UDP, TCP is reliable and guarantees delivery of data. The sockets will provide fast and persistent communication between the server and the web browser.

2.2 Product Functions

The user will first have to visit our web-app on a modern web browser.
They will then have full access to the site and all its functions.

Log in/Create Account

When the site is launched, the user will be able to log in to their existing account. If they do not have an account, they will be prompted to create a new account.

Forgot Password

There will be a password recovery tool that will send an email to the user to reset the password if necessary.

Map Traversal

The system will use the map to display nearby events and the user's current location. Its function is to give the users simple control over the information in an easily understandable and familiar way. Using the map, an interface that is already understood by the user means the user will not have to learn new controls.

Event Watch List

The event watch list is a pop-up menu that displays events that you are interested in but have not yet joined. The function is to keep all events in one convenient location, so then if you are available you can join the events from here, rather than searching the map all over again.

My Events

This is the hub for all the information of all the events

2.3 User Characteristics and Objectives

Users will be able to access Socialise on any web browser. Our users will be those looking to meet new people and make new friends but also people who want to organise activities with their existing friends or clubs/societies. The target audience for the web-app is individuals or small groups of people who are looking for a larger number of individuals to partake in a specified event or activity.

The web-app will support groups of all sizes, that can join or create events based on users interests. That being said the app can be used by anybody with the desire of its intended purpose.

Socialise will be developed with a user-friendly layout. We want it to be as intuitive as possible. We don't want our users to have to figure out how to interact with the web app. It will be seamless and very efficient for the user. It will be designed with all users in mind and therefore will follow best design practices. For example, we will take into account the size of buttons so users will be able to press these at ease.

2.4 Operational Scenarios

Create Account:

When a user opens the web app for the first time and has not already created an account, they will be requested to create an account. They will enter a username, password and email address.

Log in:

The user will be required to sign-in to their account using their email address and password. The web app will be available to use once they log in.

Look for activities/events near you:

The user will be displayed with the map interface. Here, they can manoeuvre around the map finding activities or events that interest them. These events will be marked on the map for the user to see.

Add friend:

The user can choose to add the friend by searching users and through viewing their account may choose to send a friend request which has to be mutually accepted by the recipient. If you come across a user by any other means such as in an even you will be able to add this user without searching for their user ID.

Delete/Removing Account:

You can delete a friend by searching for the friend or by viewing their profile in any other means. In place of the “add friend” button, there will be a “remove friend function”. This will remove the user from the given friend’s list permanently.

Clicks on an activity:

Once the user clicks on one of the activities on the map, they are presented with a drop-down/pop up page which gives the details of the activity. The user will be able to see the name of the event, how many participants, the day and time of the event. From here, the user will be able to request to join the activity, add the activity to their “event watch list” or can go back to the map interface.

The user wants to join the activity:

When the user wants to join the event, they click “request to join”. This will send a message to the host.

Host accepts the request:

When the host of the activity accepts the request to join, the user will be able to view the activity feed. This is a form of communication between all the participants. They can chat and send pictures here. The feed also contains the important details of the activity(date, time, location etc). This feed can also be accessed from the “My Events” tab.

Create an event:

If the user isn’t looking to join an event/activity, they also have the option to create an event of their own. Once they click “Create new event”, they fill out the following details:

1. The name of the event
2. The date of the event
3. The time of the event
4. Number of people they are looking for to join the event
5. Set to public/private

Once these are filled out, the user officially creates the event. If set to public, this will then be displayed on the map for everyone to see. If set to private, only a select number of people will be able to see and join the event.

Event watch list:

When the user wants to see the details of an event that they thought was interesting, they can go to their event watch list assuming they added it to their list. Here, they can view the event again and decide if they want to join. If they are interested, they can request to join the event, otherwise, they can remove the event from their watch list.

Delete Account:

If a user wishes to delete their account they can access this facility by going into their account settings. The settings will have the following options; “suspend” or “delete”. Suspending the account will allow the account to still exist but no-one will be able to see your account. The user can choose to reactive the account within 30 days, after this the account will be deleted. Deleting the account will permanently delete the account.

2.5 Constraints

Time Constraints:

As we have just over two months to complete the project, we will face a time constraint. We will have to stick to a strict timeline in order to get the work done

User Constraints:

As we intend a wide range of users, we will have to meet all their needs. E.g large buttons and enough spacing for those who are visually impaired.

Database memory constraints:

As the MySQL database will have limits on how much memory can be used, this will impact us on how many users we will be able to store on the database.

3 Functional Requirements

3.1 Create Account:

Description:

This will be the first interaction with the web app. The user clicks on the “create account” link and this will redirect them to a form that will have to be filled out. The form will ask the user to supply their name, email address and password. The name they supply here will be their username for the web app and the password will be

used for them to access their account. Once this form has been submitted, they will receive an email confirmation.

Criticality:

This is essential for the users to do. Without this, the user will not have access to the web app. They need to create an account to be able to see, join, and create events.

Technical Issues:

The most important issue here is making sure all login data is stored correctly in the database. Each login must be unique to that user.

Dependencies:

Without this function, a user will not be able to login to the web app and access its features.

3.2 Login

Description:

If a user has already created an account, they will be able to login using their email address and password that they registered with. Once they login, the user will have access to the web-app.

Criticality:

Logging in is essential. The user will not have access to the web app without logging in to their account.

Technical Issues:

Retrieving the user's details from the database will be an issue. The user will be logging in using the credentials they previously supplied, so getting those details back correctly could be an issue.

Dependencies:

The user must have previously created an account in order to log in. Otherwise, they will not have access to any features of the web app

3.3 Look for activities/events near you:

Description:

Here, the user has gained access to the web app and is able to look for activities they would be interested in joining. They can traverse the map interface and see all the activities in their area.

Criticality:

It is important that the map interface displays the activities correctly. It needs to update when a new activity is created and the user should be able to see these changes on the map. It needs to also display the activity in the correct location on the map(the host's location).

Technical Issues:

The map failing to update after each new event created is an issue that could arise. Displaying the event in the correct location is also an issue.

Dependencies:

The user must be logged in to their account in order to have access to the map interface.

3.4 Clicks on an activity:

Description:

Once the user clicks on one of the activities on the map, they are presented with a drop-down/pop up page which gives the details of the activity. The user will be able to see the name of the event, how many participants, the location, the day and time of the event. From here, the user will be able to request to join the activity, add the activity to their "event watch list" or can go back to the map interface.

Criticality:

It is crucial that the activity that the user clicks on pops up the correct information relating to that activity.

Technical Issues:

The activity information not displaying correctly to the user is an issue.

Dependencies:

The user has to click on the activity when on the map interface

3.5 User requests to join:

Description:

When a user requests to join an activity, the host of the activity is messaged/pinged. The message will ask if they want to accept the user into the activity. The host can view the user's profile and decide to accept or reject the user from joining the event.

Criticality:

It is important that the host is asked for permission to join the activity because they won't want just anyone joining. It allows the host to make sure that the user fits the group well (interests/hobbies are similar to those in the event already, or the user is in the correct age group etc).

Technical Issues:

As soon as the user requests to join, a message should be sent to the host. This message mightn't send correctly.

Dependencies:

The user clicks "request to join".

3.6 User adds the activity to their "event watch list"

Description:

If a user is interested in an activity or event but not ready to join it, they can add it to their event watch list. This list will store all the user's interesting events and allow them to go back to the event's details quickly without having to traverse through the map to find it again.

Criticality:

This is important because it would be very tedious and time consuming for the user to have to go back to the map interface to find an event they saw before that they were interested in. This makes life easier for the user.

Technical Issues:

An issue could arise here if it doesn't get added to the list.

Dependencies:

User has to click "add to event watch list"

3.7 User goes back to the map interface

Description:

When a user clicks on an activity and has decided to join or not, they can click outside the pop-up/drop-down in order to get back to the map interface.

Criticality:

It is crucial that the user can exit the details of an activity and return to the map interface.

Technical Issues:

An issue could be that the user could get stuck on the activity details and not be able to return to the map interface.

Dependencies:

User clicks outside the event pop up

3.8 User gains access to the event/activity (Host accepts request)

Description:

The user will be able to view the activity feed. This is a form of communication between all the participants. They can chat and send pictures here. The feed also contains the important details of the activity(date, time, location etc). This feed can also be accessed from the “My Events” tab.

Criticality:

The user must gain access to the event feed to be able to see the participants who have joined, see the day, time, location of the event and communicate to others in the group.

Technical Issues:

Allowing the user to have access could be an issue

Dependencies:

The user must have requested access to the event/activity

3.9 Leave event

Description:

If the user doesn't want to be a part of the activity/event anymore, they can choose to leave the event/activity.

Criticality:

It is crucial that the user is able to remove themselves from the participant's list if they can't attend so it frees up a space for anyone else wanting to join.

Technical Issues:

No technical issue at the moment

Dependencies:

The user must be part of an event/activity to be able to leave the group

3.10 User communicates with the group

Description:

When a user joins an activity and has access to the activity feed, there is text chat communication between all participants. This chat will be handled by web sockets. The users will send messages to the server and the server will then broadcast these messages to the group chat

Criticality:

It is important that every member of the group can chat with each other. This allows people to get to know each other before the activity/event date.

Technical Issues:

If users have difficulty connecting to the server, they won't be able to send and receive messages. The database can only store a certain amount of messages and so older messages will have to be deleted after a period of time.

Dependencies:

A user must have joined an activity to communicate with the group.

3.11 Create activity

Description:

A user can create an activity of their own. By clicking “Create Activity”, they will supply the name of the activity, the date it will take place and the location. Once they click “Create”, this activity will be added to the map interface for every user to see.

Criticality:

It is crucial that the activity created gets added to the map interface. If this doesn’t happen, no one will be able to see the activity.

Technical Issues:

The activity not adding to the map correctly

Dependencies:

The user must have logged into their account to create an activity

3.12 Add Friend

Description:

A user can choose to add a friend to their friend list. When a user clicks on another user’s profile, they will have the option to add them as a friend. Once two people are friends, they can send messages to each other.

Criticality:

The database will locate the name of the user. The friend’s name will be supplied to the friend’s list once the database retrieves it.

Technical Issues:

No technical issues at the moment

Dependencies:

The user must be logged in and part of an activity to add a friend.

3.13 Remove Friend

Description:

A user can choose to remove a friend from their friend list. This will delete the name of the friend from the friend's list.

Criticality:

A user should be able to remove a friend from the list if they don't get on well with each other.

Technical Issues:

No technical issues at the moment

Dependencies:

The user must have added a friend.

3.14 Delete Account

Description:

A user can choose to delete their account and this will permanently remove the user's data from the database.

Criticality:

To free up space on the database, if a user doesn't use their account anymore, they should delete the account.

Technical Issues:

No technical issues at the moment

Dependencies:

The user must have created an account.

4. System Architecture

4.1 Backend

The backend database will be run on a SQL database. The function of the database will be to store any data that is related to a user. This may be personal information like their name and date of birth. It may also attribute such as their friend's list. We were planning to use the name as the key in the database. This proved to not be acceptable as it did not accommodate for people with the same name.

We decided to create a "username" that would only be used as a key in the database and also used for adding a new friend. This solved our duplicate problem as the name was now a value in the database associated with the "username" which was unique. The username would be in the form "JohnDoe123456". This will be used when adding new friends.

We are planning to use java to monitor our database. From our research java works well with MySQL through the "java.sql" library. We trust that this will give us the most control over the database.

4.2 Frontend

On the frontend, we will be using python, HTML, CSS and javascript. We are aiming to use flask as our framework in python as we are most familiar with it. Django is also an option that will be considered if it is necessary or more efficient. The frontend will be hosted on HTTPS. We will use javascript to make an interactive and engaging website that is a joy to use for the user. The UI will be simple and elegant so the user will always be aware of the options that are available to them at any given moment.