Senior Project Design

Arjun Gupta, Jonathan Chianglin

# 1 Introduction

We are members of the co-ed community service fraternity, Alpha Phi Omega. Each chapter of the fraternity utilizes a web-application called APO Online. This website allows members to create, sign up, and manage events for the fraternity, as well as cataloging all brothers, their requirements and their current status in the fraternity.

# 2 Project

### 2.1 Problem

The mobile web version of APO Online creates a user experience that does not compare to the standard desktop user experience. As of now, there is no mobile application that exists to connect to APO Online.

### 2.2 Solution

Our solution is to create a native mobile application for both iOS and Android users using React Native, a framework of building native applications using React. This will allow brothers of the fraternity to easily sign up and manage events. In addition, we plan to implement a new messaging system within the application to allow brothers to more easily communicate with one another or certain groups of brothers.

### 2.3 Milestones/Deliverables

We wanted to implement the core functionalities of APO Online into the mobile application. These include searching and signing up for events and completing events. We also plan to implement a new messaging feature. The ﬁrst milestone is to have an Alpha version submitted to both App Stores at the end of February. It is to account for the time it takes for each app store to process and approve an application. The second milestone is at the end of April where the Beta version is released with the additional functionality being implemented. The third and ﬁnal milestone is at the end of May where we make changes based on the feedback from the second milestone.

### 2.4 Tentative Schedule

We will be operating under the Agile methodology where we have monthly sprints. Each month is constructed of 3 weeks of development and 1 week of testing and bug ﬁxes. The schedule is described below in Table 1. Each week will include a SCRUM meeting with the engineering team and a meeting with our advisor.

### 2.5 Special Equipment

None.

# 3 Team Description

### 3.1 Members

1. Anthony Schmidt - Owner and Founder of APO Online

2. Nicholas Reaves - University of Kentucky Student

3. Arjun Gupta - Cal Poly SLO Student

4. Jonathan Chianglin- Cal Poly SLO Student

### 3.2 Responsibilities

Anthony will be in charge of the existing server-side code that currently works for the web application. He will left the decision making up to the students. The responsibilities listed are not permanent and are subject to change.

• Nicholas Reaves - Setting up OAuth 2.0 and implementing event features

• Arjun Gupta & Jonathan Chianglin - Designing and implementing messaging functionality

### 3.3 Cross Evaluating

We have weekly SCRUM meetings to evaluate our progress and address any issues from other members.

# 

# 

# 4 System Requirements

4.1 User Stories

1. As the president I want to easily message all the officers on my executive board to more easily communicate with them.
2. As an event coordinator, I want to message all individuals attending an event to send updates, reminders, and event information.
3. As an exec member I want to easily message members of my committee to more communicate pending tasks and improve productivity.
4. As a brother of the fraternity, I want a central channel to receive updates and information regarding the chapter.
5. As a brother of the fraternity, I want to be able to receive notifications to my phone about messages and updates to stay updated on fraternity happenings.

### 4.2 Use Cases

1. Create direct message with one or more individuals.

|  |  |
| --- | --- |
| Description | Users want to create a direct messaging forum with only selected individual(s) involved. |
| Flow | 1. User opens the Messaging module 2. User clicks button to create direct message 3. User enter individual(s)’ names in search bar 4. User selects the name(s) |
| Priority | High |

1. Create channel with one or more individuals.

|  |  |
| --- | --- |
| Description | Users want to create a channel, a forum with posts and messages, with selected individual(s) involved. |
| Flow | 1. User opens the Messaging module 2. User clicks button to create channel 3. User enter channel name 4. User enter individual(s)’ names in search bar 5. User selects the name(s) 6. User sets channel’s privacy setting 7. User confirms given information |
| Priority | High |

1. Add individuals to a channel based on search criteria.

|  |  |
| --- | --- |
| Description | User may quickly add certain members based on specific criterias like membership status and events attending. |
| Flow | 1. User opens the Messaging module. 2. User select channel to add individuals to. 3. User clicks button to add individuals to channel. 4. User select search criteria to search for members. 5. User enter query given search criteria. 6. User selects individuals listed from query. 7. User confirms to add selected individuals. |
| Priority | Medium |

1. Send a message to a channel.

|  |  |
| --- | --- |
| Description | User sends a message, any form of media, to the entire channel. |
| Flow | 1. User opens the Messaging module. 2. User selects channel or direct messaging group. 3. User types message in message bar. 4. User enters message to the channel. |
| Priority | High |

1. Remove individuals in a channel

|  |  |
| --- | --- |
| Description | User removes individual(s) given they have the correct permissions for that channel. |
| Flow | 1. User opens the Messaging module. 2. User selects to view individuals in the chat 3. User selects an icon near another user’s name to remove them from the channel 4. User confirms to remove the other user from the channel 5. The system confirms that the user has the correct permissions to remove another user 6. The other user is removed from the channel. |
| Priority | Medium |

1. Delete individual messages

|  |  |
| --- | --- |
| Description | User deletes messages within a direct message or channel given they have the permissions for that channel. |
| Flow | 1. User opens the Messaging module. 2. User opens a direct message or channel 3. User selects options under a message 4. User selects to delete the selected message 5. The user confirms to delete the selected message 6. The selected message is deleted |
| Priority | Low |

1. Delete direct messages

|  |  |
| --- | --- |
| Description | User is able to delete a chat with an individual(s). |
| Flow | 1. User opens the Messaging module. 2. User opens a direct message 3. User selects options for the opened direct message 4. User selects to delete a chat 5. User confirms to delete chat 6. Chat is deleted and removed from the users recent chats |
| Priority | Low |

1. Delete channel

|  |  |
| --- | --- |
| Description | User is able to delete a channel given the they have the correct permissions for that channel |
| Flow | 1. User opens the Messaging module. 2. User opens a channel 3. User opens chanel options 4. User selects to delete a channel 5. User confirms to delete channel 6. System validates users permissions 7. System removes channel from users channels and blocks other users from sending messages to it |
| Priority | Low |

### 4.3 Functional Requirements

|  |  |
| --- | --- |
| FR-1 | The system shall have a permissions system. |
| FR-2 | The system shall display an error if a message is unable to be sent. |
| FR-3 | The system shall display an error if the user is unable to connect to the internet. |
| FR-4 | The system shall allow a user to view an individual’s APO profile if their name is selected. |
| FR-5 | The system shall allow admins of a channel to control who can send messages in it. |
| FR-6 | The system shall prompt the user to enter channel name and members’ names upon creating a channel. |
| FR-7 | The system shall indicate how many unread messages a user has per direct message or channel. |
| FR-8 | The system shall allow the user to mute a specific channel. |
| FR-9 | The system shall allow users to send and receive files in a channel. |
| FR-10 | The system shall display a message to confirm if the user wants to leave a certain channel. |

### 4.4 Nonfunctional Requirements

|  |  |
| --- | --- |
| NFR-1 | The system shall utilize the APO Online API. |
| NFR-2 | The system shall take no longer than thirty minutes to be learned by a user with a nontechnical background. |
| NFR-3 | The system shall request to sync messages within 1 second of opening a channel. |
| NFR-4 | The system shall send push notifications to a user when they receive a new message. |
| NFR-5 | The system shall have a search bar to search for users. |
| NFR-6 | The system shall be designed to easily implement additional functionalities in the future. |