**CSCI 5530**

**Team 6**

**Griffin Bryant, Marquise Jennett, Dylan Myers, Mansi Pandya, Everrick Wright**

**Software Requirements Specification**

**Document**

**Version: 1** **Date: (02/13/2020)**

**Table of Contents**

[**1. Introduction**](#_fng9s7iy58v9) **3**

[**1.1 Purpose**](#_8w16ua8z5ky3) **3**

[**1.2 Scope**](#_e1tpsoo7ocap) **3**

[**1.3 Definitions, Acronyms, and Abbreviations.**](#_xx5uo21vjjx9) **4**

[**1.4 References**](#_mrw4wwg3hijb) **4**

[**1.5 Overview**](#_ify9for4ugmx) **5**

[**2. The Overall Description**](#_x7bdlwz4l00r) **5**

[**2.1 Product Perspective**](#_mnn7yk6vynqd) **5**

[**2.1.1 System Interfaces**](#_d04pyq2a01py) **5**

[**2.1.2 Interfaces**](#_3tmga3nhoxgp) **5**

[**2.1.3 Hardware Interfaces**](#_njofqeatkyh1) **5**

[**2.1.4 Software Interfaces**](#_cd8tr881jf3r) **6**

[**2.1.5 Communications Interfaces**](#_ntu2y3zdw9yb) **6**

[**2.1.6 Memory Constraints**](#_nzl2rplsn479) **6**

[**2.1.7 Operations**](#_ke4s2mp1yels) **6**

[**2.2 Product Functions**](#_lwhw6bdac27x) **7**

[**2.3 User Characteristics**](#_oqmmktgoeno9) **7**

[**2.4 Constraints**](#_o0gmaiyu3wma) **7**

[**2.5 Assumptions and Dependencies**](#_3vucqff016pz) **7**

[**2.6 Apportioning of Requirements.**](#_d09taovrqdy0) **7**

[**3. Specific Requirements**](#_4j3cbps6ax8i) **8**

[**3.1 External Interfaces**](#_dd9b29ecv0o8) **8**

[**3.1.1 User Interfaces**](#_8o5qwbwap7tn) **8**

[**3.1.2 Software Interfaces**](#_o7lv5w4c3lgq) **8**

[**3.2 Functions**](#_jt7j1w87z5zz) **8**

[**3.5 Design Constraints**](#_d6sq9hhkqq6v) **9**

[**3.5.1 Standards Compliance**](#_vi9e2uduj50z) **9**

[3.6 Software System Attributes](#_cqz2b6svy93s) 10

[**3.6.1 Reliability**](#_ar5anw6g70db) **10**

[**3.6.2 Availability**](#_azk918p4u4ax) **10**

[**3.6.3 Security**](#_skmtu2c9iuwe) **10**

[3.6.4 Maintainability](#_mcwx3t7c3d89) 11

[**3.6.5 Portability**](#_dfa4dcv2z41n) **11**

[**4. Change Management Process**](#_n1frdd6aabmb) **11**

[**5. Supporting Information**](#_ub3rc9qy5upz) **12**

[**5.1 Use Case Diagram**](#_qll2wizfmdp5) **12**

[**5.2 Use case stories.**](#_5ikh4pobbse1) **12**

[Case number: 1.0](#_it5y5swgs5jx) 12

[Case number: 1.1](#_8qbdgllp9e0j) 13

[Case number: 1.2](#_p0ldo3l4wc39) 14

[Case number: 1.3](#_xkmajh1md4nn) 14

[Case number: 1.4](#_ktl3y912zll4) 15

[Case number: 1.5](#_96vdnw9hw49t) 15

[Case number: 1.6](#_yyyi214qaeua) 16

[Case number: 1.7](#_3jwbzvb1ml5) 16

[Case number: 1.8](#_3eho06rvbarb) 17

[Case number: 1.9](#_3lxkwzstzn7i) 17

[Case number: 2.0](#_dgv168nj55tm) 17

[Case number: 2.1](#_50a5hr8xsnef) 18

[Case number: 2.2](#_lrwit2lq2hfa) 18

[Case number: 2.3](#_vyx5o7epdutk) 18

[Case number: 2.4](#_pmtzbq3sq7eu) 19

[Case number: 2.5](#_1shd56jb575r) 19

[Case number: 2.6](#_w9woquevl9mq) 20

[Case number: 2.7](#_aucvujn9tkmr) 20

[Case number: 2.8](#_ry1x868ojmyx) 21

[Case Number 2.9](#_2ofavau6jrmi) 21

[Case Number 3.0](#_hbryntswsvpw) 22

[Case Number 3.1](#_5r7dh7f7mqs1) 22

[Case Number: 3.2](#_9u0drxrwjzsd) 23

[5.3 Pert Chart](#_asj9uzenuihj) 24

[**5.4 Gantt Chart**](#_ajry0s4yi47u) **25**

[**5.5 Risk Assessment**](#_b90oenahh60t) **25**

# **1. Introduction**

The purpose of the Software Requirements Specification is to outline the requirements for the student Association for Computing Machinery application. Will be cross platform, will use a database, and will use a web hosting service.

## **1.1 Purpose**

The ACM application will be a mobile application to provide a platform for communication and resources for ACM members and computer science students.

## **1.2 Scope**

1. Our application will be a mobile application for the ACM.
2. The application will be used as a tool for ACM members, ACM Executive Board members, and Georgia Southern ACM Alumni members.
3. The tool should be mainly utilized as a chat system with additional add ons.
4. The benefits of our application will be to provide a streamlined application that will allow members and admin to communicate and schedule events more effectively.
5. The objectives of our software will be to allow members and admin to fulfill the requirements they have specified.
6. The goal will be to produce software that will be easy to use and efficiently complete its tasks.

## **1.3 Definitions, Acronyms, and Abbreviations.**

ACM - Association for Computing Machinery

MySQL - The database management system

AWS - Amazon Web Services

## **1.4 References**

<https://aws.amazon.com>

<https://dotnet.microsoft.com/apps/xamarin>

<https://visualstudio.microsoft.com>

## 

## **1.5 Overview**

This document contains all of the software requirement specifics. It contains a general description of the types of users who will be using our application, how it is going to work, and what technologies we are using to make it work. We will also outline and describe specific components of the project.

# **2. The Overall Description**

The main tasks with respect to our software will be to provide a user account for admin and users. The admin being the executive members and the users being the ACM members and alumni. The admin will be able to schedule events, post ACM videos, and be able send messages out to users. The users will then be given the opportunity to ask for mentorship on particular subjects related to the ACM. Alumni users will also be given the choice to still be involved with the ACM.

## **2.1 Product Perspective**

This product is designed to run on either Android or iOS devices. The sole requirement for the user is a mobile device and an active internet connection.

### **2.1.1 System Interfaces**

iOS - This system will be the interface for iPhone users. It will have the same functionality as the android system, and both will be connected to one database. The database will be built by us, thus no connecting API will have to be utilized. The iOS system will connect to Amazon Web Services.

Android - This system will be the interface for android users. It will have the same functionality as the iOS system, and will share the same database as the iOS system. This system will connect to Amazon Web Services.

### **2.1.2 Interfaces**

Our application will interface from whichever mobile device the user is on to the database.

### **2.1.3 Hardware Interfaces**

The ACM Application will work on Android and iOS mobile devices. Because most of the applications features will require an internet connection as well as location data, the devices will also require internet connectivity and GPS signal.

### 

### 

### **2.1.4 Software Interfaces**

1. Name: Xamarin
2. Mnemonic:
3. Version: 4.3
4. Source: <https://dotnet.microsoft.com/apps/xamarin>
5. Name: Visual Studios
6. Mnemonic: VS
7. Version Number: 16.0
8. Source: <https://visualstudio.microsoft.com>
9. Name: GitHub
10. Mnemonic: Git
11. Version Number:
12. Source: <https://github.com>
13. Name: Amazon Web Services
14. Mnemonic: AWS
15. Version Number: 21.0
16. Source: <https://aws.amazon.com>

### **2.1.5 Communications Interfaces**

The main interfaces for are communications will be a AWS cloud server that will handle the requests of clients. We also will use a API of a chat like GroupMe to integrate with our own app

### **2.1.6 Memory Constraints**

Our target RAM size is between 50 - 60M of RAM and not exceeding 60 M. This memory constraint is based off of GroupMe most current required memory constraint of 60MB.

### **2.1.7 Operations**

The user will have several operations when logging onto the application. The main operation will be the main chat room where casually communication can take place between eboard members, standard acm members, and alumni. From the main chat room the user will have the options of changing chat rooms or selecting from several tabs. A regular user will have the options of ACM sponsored videos, events with gps, pay dues, and look for tutoring help. The admin will be able to add/drop members, change roles, add events for regular users to attend, accept dues, and assign people who are able to tutor. The chat will be saved and able to have recovery options due to semi-constant backups.

## **2.2 Product Functions**

The main function of the ACM application is the group chat functionality

* This function will allow users to send messages and the rest of the chat’s application will be updated with the sent message.

Other sub-functionalities include:

* GPS tracking to set ACM events
* Different chat rooms for different topics (alumni chat, executive board chat, memes chat, homework chat)
* A matching algorithm for mentors/mentees
* A section that displays/links ACM sponsored videos

## **2.3 User Characteristics**

ACM executive members will have the power to add events, moderate the chat, and upload ACM videos.

ACM members will be able to ask for mentorship, message others, select chat groups, and know the geographic location of events.

## **2.4 Constraints**

The application will comply with the Americans with Disabilities Act Standards for Accessible Design. Because the application will handle user accounts, security is important and all user information will be encrypted.

## **2.5 Assumptions and Dependencies**

It is assumed that users with iPhones will be using iOS version 13.3.1 or newer and users with Android devices will be using Android version 10.0 or newer.

## **2.6 Apportioning of Requirements.**

**Requirements that may be delayed until future versions of the system.**

* The system may have a point system to rank members based on the things they have done for the ACM club
* The system should allow users to get in contact with possible employers for internships, co-ops, and jobs
* The system should allow users to see a list of known ACM GSU employers
* The system may have the ability for administrators to keep a tally of the attendance at each meeting
* The system may allow administrators to accept electronic payments via paypal or venmo
* The system may allow administrators to create anonymous voting polls for the executive board

# **3. Specific Requirements**

## **3.1 External Interfaces**

### 3.1.1 User Interfaces

Users will interact with the ACM Application through their mobile device. The resolution and orientation of the application will depend on the device used and the orientation the user holds it at. The Xamarin Framework allows us to create a responsive application that will automatically scale for each device/orientation. When the user is navigating the app, there will be various menus and forms for them to navigate and use to submit information. There will also be an on-screen keyboard for the user to type with while chatting with other users or entering information into the forms.

### 3.1.2 Software Interfaces

The ACM Application will interface with AWS to host all of the back-end information including the database and files. We plan to use MySQL version 8.0.19 for our database.We also expect to use an API like GroupMe to handle all of the chat functionalities.

## **3.2 Functions**

* The system shall have the ability to log onto the application
* The system database shall store titles of board members
* The system shall set eboard members as club administrators
* The system shall allow users to view events
* The system shall allow users to open event locations in their default map application
* The system shall allow users to request a mentor
* The system shall allow the tutor to be matched by things like class experience
* The system may have a rating system for mentors
* The application shall be able to send notifications for meetings and events via email or push notification
* The system shall prevent messages containing profanity from being sent to in the chat
* Users shall be able to message other users directly
* Users shall be able to send messages inside of groups with set topics
* Non-members shall be able see events and create an account
* Administrators shall have the ability to approve new users
* Users shall have the ability to create a profile
* The system may have a point system to rank members based on the things they have done for the ACM club
* The system should allow users to see a list of known ACM GSU alumni
* The system shall have the ability to keep a tally of the attendance at each meeting
* The system shall allow a user to be removed from the ACM app after deferring to ACM constitution
* The system shall allow administrators to create anonymous voting polls for the executive board

## **3.5 Design Constraints**

Since we will be using a framework we will be limited to the features of whichever framework we use.

### **3.5.1 Standards Compliance**

The ACM application will conform to the Americans with Disabilities Act Standards for Accessible Design and the WAI’s Addresses Mobile Accessibility standards. These standards state that all electronic and information technology must be accessible to people with disabilities.

## 

## 

## **3.6 Software System Attributes**

* The login process should be completed ***within 3 seconds***
* The system shall have the functionality of assigning roles with a displayed title such as ***President, Vice-President, Treasurer, Officer, Secretary***
* The system shall allow users to use a chat system ***that saves messages for 1 semester’s length***
* The system shall allow users to request a tutor ***and the request is sent within 1 minute***
* The application shall be able to send notifications for meetings and events via email or push notification ***that is sent out within 30 seconds from initiation***
* The system shall prevent messages containing profanity from being sent to in the chat ***immediately***
* Users shall be able to message other users directly ***that is sent within 1 minute***
* Administrators shall have the ability to approve new users ***members are added immediately (3-5 seconds)***
* Users shall have the ability to create a profile ***and it be displayed immediately***
* The system shall allow administrators to kick a user out of the ACM app ***access of logging into application should be enacted immediately after removal***

### **3.6.1 Reliability**

The system should be able to handle its required users concurrently. The system should be able to handle high messaging rate. The system should not have frequent crashes

### **3.6.2 Availability**

The system should be available 24/7. Because we are using AWS our application's server-side availability will be limited to Amazon's services uptime. Client-side availability will be dependent on the user's device availability.

### **3.6.3 Security**

We will utilize password encryption via RSA encryption to ensure password security. We will sanitize all incoming data. We will use getters and setters to avoid unwarranted access of data.

### 

### **3.6.4 Maintainability**

Due to the nature of mobile applications, the ACM Application will have to be updated periodically to suit the most recent version of whichever operating system is in use. The Xamarin framework simplifies this process with a built-in migration tool for the most recent API for iOS and Android devices.

### **3.6.5 Portability**

The portability of the application between iOS and Android will not be difficult because all of our development will be cross-platform through Xamarin. However, if we were to try to port it to a web application it would take a lot of extra work because Xamarin doesn't currently support web development. If we were to create a web-accessible version of our application, we would have to start from scratch for that portion.

# 4. Change Management Process

Our group will be open to minor changes to the application requirements as long as the team reaches a consensus on whether or not the change is feasible. All changes shall be submitted in writing to the team leader so they cannot be lost/forgotten. Requested changes will then be passed on to be discussed with the rest of the team for approval.

# 

# 5. Supporting Information

## 5.1 Use Case Diagram



## 5.2 Use case stories.

### Case number: 1.0

**Title (Goal):** The system shall have the ability to log onto the application

**Description:** Board member signs up and waits for approval for full usability of application

**Primary Actors:** User, Database system

**Main:** *Pre-condition: User must be a verified board member via the eboard*

1. User clicks ‘sign up’ page

2. The application sends user application to database

3. Eboard member logs on

4. Eboard member clicks ‘member verification’

5. Selects users to approve

6. Board member now has access to full application

Post Condition: None

### Case number: 1.1

**Title (Goal):** The system database shall store titles of e-board members

**Description:** Board member signs up

**Primary Actors:** User, Eboard member, Database system

**Main:** *Pre-condition: User must be a verified eboard member via the president*

1. User clicks ‘sign up’ page

2. The application asks if the user is a board member

3. The database checks if the given user is approved by the President

4. After approval, user profile now has attached title

Post Condition: None

### 

### Case number: 1.2

**Title (Goal):** The system shall set eboard members as club administrators

**Description:** Admins will change a boolean in database to allow eboard to have limited administrative abilities

**Primary Actors:** Admin, User, Database system

**Main:** *Pre-condition: User must be a verified eboard member via the admin*

1. Administrator logs in

2. The administrator finds user needing club admin status

3. The admin updates user to club user via isClubAdmin checkbox

4. Database updates

5. User now has club admin capabilities

Post Condition: None

### Case number: 1.3

**Title (Goal):** The system shall allow users to view events

**Description:**

**Primary Actors:** Admin, User, Database system

**Main:** *Pre-condition: User must be a verified board member via the admin*

1. Administrator logs in

2. The administrator finds user needing club admin status

3. The admin updates user to club user via isClubAdmin checkbox

4. Database updates

5. User now has club admin capabilities

Post Condition: None

### 

### Case number: 1.4

**Title (Goal):** The system shall allow users to open event locations in their default map application

**Description:** After user selects event to attend, the option of opening event location in default map application appears

**Primary Actors:** User, Database system

**Main:** *Pre-condition: User must be logged into application*

1. User logs in

2. User clicks on ‘events’ tab

3. User selects event wanting to attend

4. Notification appears stating “Open Event Location in Maps?”

5. User clicks ‘yes’

6. Default map application opens with location of event

Post Condition: None

### Case number: 1.5

**Title (Goal):** The system shall allow users to request a mentor

**Description:** User can request a mentor for a class subject

**Primary Actors:** User, Database system

**Main:** *Pre-condition: User must be logged into application and mentor must be approved tutor*

1. User logs in

2. User clicks on ‘mentor’ tab

3. User selects class they need mentoring in

4. User selects request tutor

5. User request/profile is sent to database to await matching of tutor

6. Notification is sent to eboard member handling matching of tutors

Post Condition: Go to Case #1.7

### 

### Case number: 1.6

**Title (Goal):** The system shall allow the tutor to be matched by things like class experience

**Description:** Tutor is matched with requested user for specific class experience

**Primary Actors:** User, Database system

**Main:** *Pre-condition: A request must have been sent for a tutor*

1. Club admin logs in

2. Club admin clicks on ‘mentor requests’ tab

3. Club admin selects mentor request application

4. Club admin selects users in application with specific class experience requested

5. Club admin approves tutor

6. Tutor receives mentor link message

7. Tutor approves link request

8. User receives notification that he/she is matched with tutor

Post Condition: None

### Case number: 1.7

**Title (Goal):** The system shall prevent messages containing profanity from being sent to in the chat

**Description:** Predefined explicit words that are not allowed will not appear in any chat

**Primary Actors:** User, Database system

**Main:** *Pre-condition: User must be logged into application and mentor must be approved tutor*

1. User logs in

2. User clicks on ‘mentor’ tab

3. User selects class they need mentoring in

4. User selects request tutor

5. User request/profile is sent to database to await matching of tutor

6. Notification is sent to eboard member handling matching of tutors

Post Condition: Go to Case #1.7

### 

### Case number: 1.8

**Title (Goal):** The application shall be able to send notifications for meetings and events via email or push notification

**Description:** Notifications are sent to user when they sign up for events and meetings

**Primary Actors:** User, Club admin, Database system

**Main:** *Pre-condition: user must be logged into application*

1. User logs in

2. User clicks on ‘Upcoming events’

3. The user selects an event or a meeting to attend

4. A push notification and an email is sent to the user on the event day

Post Condition: None

### Case number: 1.9

**Title (Goal):** Users shall be able to message other users directly

**Description:** User uses the personal messaging feature to connect with another registered user

**Primary Actors:** User, database system

**Main:** *Pre-condition: user must be logged into application*

1. User logs in

2. User selects another user from ‘direct message’ list

3. The user connects to another user

Post Condition: None

### Case number: 2.0

**Title (Goal):** Users shall be able to send messages inside of groups with set topics

**Description:** The topics are set by the admin for discussions

**Primary Actors:** User, Admin

**Main:** *Pre-condition: user must be logged into application*

1. User logs in

2. User clicks on ‘Discussions’

3. User gets added to the “Discussions” chat with other members

Post Condition: None

### 

### Case number: 2.1

**Title (Goal):** Non-members shall be able see events and create an account

**Description:** Prospective users have access to read club’s upcoming events

**Primary Actors:** User, database system

**Main:** *Pre-condition: user must have the application installed*

1. User opens the application

2. User clicks on ‘Upcoming events’

3. To sign up and get more details, the application asks for the user to sign up

4. User signs up using the ‘sign up’ page

5. User creates an account

Post Condition: None

### Case number: 2.2

**Title (Goal):** Users shall have the ability to create a profile

**Description:** the user adds personal information for the profile

**Primary Actors:** User, database system

**Main:** *Pre-condition: user must be approved by the admin to sign up for the club*

1. User logs in

2. User clicks on ‘User profile’

3. User updates ‘contact address’, ‘courses taken’, ‘current courses’

4. User saves the profile

Post Condition: Go to case #2.3

### Case number: 2.3

**Title (Goal):** The system should allow users to see a list of known ACM GSU alumni

**Description:** Members will be able to see all members of ACM that have graduated from GSU

**Primary Actors:** User, database system

**Main:** *Pre-condition: user must be logged in and be a member*

1. User opens application

2. User must log in

3. Users will be able to see other users with the tag of Alumni

Post Condition: None

### 

### Case number: 2.4

**Title (Goal):** The system shall have the ability to keep a tally of the attendance at each meeting

**Description:** ACM board members and/or the administrator can use the app to keep track of who has/hasn't show up to a meeting

**Primary Actors:** User, database system

**Main:** *Pre-condition: User must be a board member or an administrator*

1. User opens application

2. User open ‘Events’ Page

3. User selects event or meeting that they would like to do attendance for

4. User selects which user attended that meeting/event

Post Condition: None

### Case number: 2.5

**Title (Goal):** The system shall allow a user to be removed from the ACM app after deferring to ACM constitution

**Description:** Board members will be allowed to ban users from using the app based on violation of ACM constitution

**Primary Actors:** Administrator, database system

**Main:** *Pre-condition: user must be an administrator*

1. User opens application

2. User clicks the name of another user

3. User selects ‘Ban’

4. User selects reason for ban

5. Selected user is then banned

Post Condition: None

### 

### Case number: 2.6

**Title (Goal):** The system shall allow administrators to create anonymous voting polls for the executive board

**Description:** Polls will be created for ACM board to vote on

**Primary Actors:** Board Members, Administrators, database system

**Main:** *Pre-condition: To post a poll the user must be an administrator and to vote user must be a part of the ACM executive board*

1. User opens application

2. User goes to the ‘Chat’ page

3. User selects ‘Create Poll’

4. User fills out poll form

5. User submits post

6. Applicable users will be able to vote on the poll

Post Condition: None

### Case number: 2.7

**Title (Goal):** The system should allow users to get in contact with possible employers for internships, co-ops, and jobs

**Description:** Users of the app will be able to see information of companies that are hiring

**Primary Actors:** users

**Main:** *Pre-condition: The user must be a member*

1. User opens application

2. User goes to the ‘Employment Opportunity' page

3. Users can browse possible employment options

Post Condition: None

### 

### Case number: 2.8

**Title (Goal):** The system should allow users to see a list of known ACM GSU employers

**Description:** Users of the app will be able to see information of companies that have hired ACM GSU alumni

**Primary Actors:** users

**Main:** *Pre-condition: The user must be a member*

1. User opens application

2. User goes to the ‘Employment Opportunity' page

3. Users can browse possible employment option

### Case Number 2.9

**Title (Goal):** The system may have the ability for administrators to keep a tally of the attendance at each meeting

**Description:** There may be a calendar or notes section for the e-board members to record meeting attendance

**Primary Actors:** User, Database system

**Main:** *Pre-condition: E-board member must be logged in to log attendance*

1. E-board member logs into the application

2. E-board member clicks on the e-board control panel

3. E-board member clicks on the attendance tab

4. E-board member either creates a note to log the attendance for that date

OR

5. E-board member selects the date of the meeting in a calendar to log the attendance for that date

Post Condition: None

### 

### Case Number 3.0

**Title (Goal):** The system may allow administrators to accept electronic payments via paypal or venmo

**Description:** There may be a page with links for members of the club to submit electronic payments to pay their dues

**Primary Actors:** User, Database system

**Main:** *Pre-condition: Users must be logged in to pay their dues*

1. User logs into the application

2. User clicks on the club information tab

3. User clicks on the pay dues button

4. User clicks on the paypal/venmo button

5. User is redirected to paypal/venmo to send their payment

Post Condition: User profile will be updated with paid symbol

### Case Number 3.1

**Title (Goal):** The system may allow administrators to see a list of members who have paid their dues

**Description:** It is important for members of the e-board to be able to keep track of who has and hasn't paid dues. This feature will solve this problem by providing e-board members with an easy-to-access place to keep track of who has paid their dues

**Primary Actors:** User, Database system

**Main:** *Pre-condition: Members of the e-board must be logged in to see this information*

1. E-board member logs in

2. E-board member goes to the e-board control panel

3. E-board member opens the member information tab

4. Each member will have a check mark next to their name if they have paid their dues and are a member of the ACM club

Post Condition: None

### 

### Case Number: 3.2

**Title (Goal):** The system may allow administrators to create anonymous voting polls for the executive board

**Description:** Users will be able to vote on polls that are created by members of the e-board

**Primary Actors:** User, Database system

**Main:** *Pre-condition: Members of the e-board must be logged into the application to create polls and users must be logged into the application in order to anonymously vote*

**E-board perspective**

1. E-board member logs in

2. E-board member goes to the board control panel

3. E-board member opens the polls tab

4. E-board member creates and submits poll

**User perspective**

1. User logs into the app

2. User goes to polls page

3. User chooses their vote and submits it

4. User's vote is recorded and statistics are shown

Post Condition: None

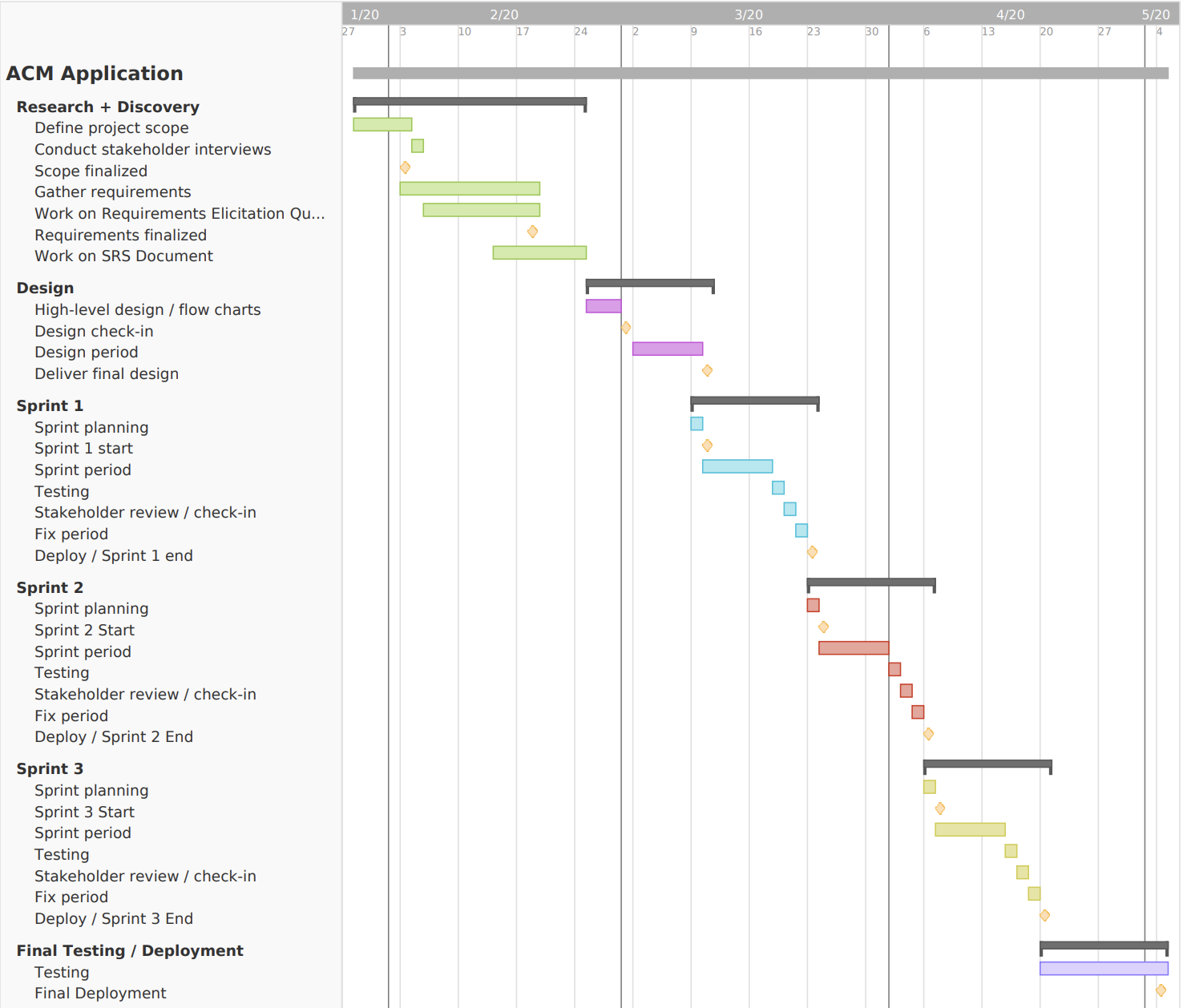
## 

## 5.3 Pert Chart



## 

## 5.4 Gantt Chart



## 

## 5.5 Risk Assessment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Risk | Likeliness | Severity | Mitigation |
| Using AWS cloud server | AWS could go down | Unlikely | Major | Have local test server |
| Using Xamarin | Could find integration issues | Likely | Minor | Revise implementation to work around |
| Team members | Could get sick | Likely | Minor | Give tasks of sick member to others |
| Working online | Internet could go out | Likely | Somewhat | Make up for time when internet comes back |
| Using visual studio | Visual studio could crash | Likely | Major | Make sure to push to github frequently |

#### 