Computer Science and Engineering

The Givers

CS 4523 - Project Management Plan (SPMP)

Version 2.0

Document Number: SPMP-001

Project Team Number: B01, Section A

Project Team Members: David Bravo (db3454), Qilei Cai (qc542), Tasnim Nehal (tn1078), and Yikai Wang (yw3193)

REVIEW AND APPROVALS

Printed Name and Title	Function (Author, Reviewer, Approval)	Date	Signature
David Bravo	Author	September 29, 2020	David Bravo
Qilei Cai	Author	September 29, 2020	Qilei Cai
Tasnim Nehal	Author	September 29, 2020	Tasnim Nehal
Yikai Wang	Author	September 29, 2020	Yikai Wang
Professor Strauss	Reviewer	September 29, 2020	Professor Strauss

REVISION LEVEL

Date	Revision Number	Purpose
April 6, 2020	Version 1.0	Initial Release
September 29, 2020	Version 2.0	Defect Correction

Table Of Contents

OVERVIEW

1.1 Project Summary

Motivation

The Givers project team has problems with how charities are currently maintaining their websites. As of now you have to go to separate websites and enter your payment information or make an account each time if you want to support multiple charities. This is an inefficient process for both charities and users. We want to streamline this process by creating a system where only one account needs to be made allowing you to donate to several different charities. This makes the experience of wanting to assist several charities as simple as possible and incentivizes charities to work together.

More than just making the donation process easier, The Givers also plans to provide much needed analytics and data visualization to these organizations. As of now, there is no system in place to view the news feeds of several charities in one place. There is no way to categorize charities based on your location if you want to support local organizations. Nor is there a way to group charities by cause if you want to support a specific movement or idea. The Givers hopes to provide a solution to the problem of how charities can reach out to donors and vice versa.

Purpose

The goal of this Software Project Management Plan is to delineate how the development process of The Givers will be carried out. This includes the amount of time, cloud services, and budget that needs to be allocated to deliver a web application that can provide data visualization and securely service the payment processing of donators.

Audience

This document is intended to read by The Givers development team, investors, project managers, and charity managers that want their organization to join the system.

1.2 Purpose, Scope, and Objectives

Purpose

The purpose of this project is to provide a platform to connect charities with donors to streamline fundraising and address the pain points in the status

quo of charity operations. The project will launch a progressive web application that enables charities to showcase themselves and donors to easily find the causes they prefer. The application will estimate each donor's charity interests and match them with the causes they resonate with. The application will also enable donors to see the status updates of the charities they follow to keep themselves up to speed, which will enhance user engagement. The development team will inspect the application rigorously prior to release as well as inviting beta users to provide feedback.

Scope

Our project, The Givers, will launch an application to enable charities and donors to find each other more easily and enhance their engagement.

The inspiration for this project originated from the fact that the status quo of charity operations remains segmented in the sense that each charity advertises for itself independently, without a common stage. The consequence is that each charity is only able to reach a subset of prospective donors and donors have no chance to discover the many other charities they might be interested in. The Givers will provide a grand platform to grow the two-way exchange between charities and donors.

The Givers will identify each donor's charity interests by showing them a wide range of charitable causes and asking them to choose the ones they resonate with. Donors will be able to follow charities to see their status updates on a timeline; the app will learn from the types of charities a donor follows to fine-tune its recommendations and connect the donor to charities that are a right fit for them.

The web app will be easily accessible from both desktop computers and mobile devices. On mobile devices, users will see a layout optimized for viewing on smaller displays. It will support features including account registration, timeline viewing, making payments and updating settings.

Objectives

The objective of the project is to broaden the stage where charities advertise for themselves and connect donors to a wider range of charitable causes.

The first release of the application will support the essential features. It will feature a registration system where both charities and donors can sign up for accounts. It will feature a timeline where donors can see the status updates of the charities they have followed. The user interface of the web app runs on the browser, while the architectures such as recommendation systems run in the

background on the server. The UI design and back-end algorithms will be updated periodically in later releases.

The application will be developed with the agile principles, namely prioritizing individuals and interactions, working software, customer collaboration and responding to change. The first version of the application is planned to be released in Fall 2020.

1.3 Assumptions and Constraints

Assumptions

This system will be a platform that can work on any web browser and will not be tied to any specific hardware meaning it will work on any mobile or computer device. AWS will be used to host the platform.

Constraints

Since The Givers will be a non-profit platform there will be constraints on how the system gathers funding to maintain servers and provide bonuses to client charities. Aside from the budget constraints, the project must have an accessible UI and be modular allowing for updates or changes to features of the system down the line. This must all be accomplished within the given deadlines.

1.4 Project Deliverables

Deliverable	Date	Type
Project Proposal	02/13/2020	Documentation
Initial System Requirements Specification	03/01/2020	Documentation
Final System Requirements Specification	03/10/2020	Documentation
Software Design Description	05/05/2020	Documentation
First Product Released	12/10/2020	Final Product

1.5 Schedule and Budget Summary

Deliverable	Date
Project Proposal	02/13/2020 (*submitted*)
Initial System Requirements Specification	03/01/2020 (*submitted*)
Final System Requirements Specification	03/10/2020 (*submitted*)
Software Analysis Specification (SAS)	04/14/2020
Software Design Description	05/05/2020
First Product Released	12/10/2020

1.6 Evolution of Plan

Initial Release	4/7/20
2.0 Release	9/29/20

As of now the initial version of the SPMP has been released and there will be one more expected release once the initial version has been reviewed and edited. Unless more edits are required, the second release will be the final version of the SPMP.

Should there be any further updates needed for the SPMP after the release of the final version, the development team will review such needs and identify the reason for such changes as well as the optimal solution. If the team deems updates are necessary, the modifications will be made and published in the next release.

2. REFERENCES

- *Project Proposal*, Version 2.0, Team B01, September 15, 2020
    - Section 1
- *System Requirements Specification*, Version 1.3, Team B01, September 22, 2020
    - Section 2.1
    - Section 3

3. DEFINITIONS

User - A person using our system, either to donate or to manage a charity.

Non-profit - Charities that take contributions for philanthropy. Revenue may be used for further development of the system.

Relational Database - A database based on the relational model that makes relationships between data points easier to identify. SQL will be used to process and access data for the relational database used in the system.

Just-in-time Compilation - A method where compilation happens during execution. JavaScript is compiled by most browsers just before the program runs.

Markup Language - A language that uses tags to format documents. HTML is a markup language and is a popular tool for web browsers to display contents.

Backend Interpreted Language - Language for the server where source code is fed into an interpreter to become machine code. Python can be used as the backend language to process requests and data.

Kanban Board – “A Kanban board is an agile project management tool designed to help visualize work, limit work-in-progress, and maximize efficiency (or flow). Kanban boards use cards, columns, and continuous improvement to help technology and service teams commit to the right amount of work, and get it done” - Max Rehkopf from Atlassian.

4. PROJECT ORGANIZATION

4.1 External Interfaces

The application will interface with two types of external entities, users who want to donate and organizations who are interacting with users and advertising on the platform. Delivering the experience will be what the software will be in

charge of and information within it will only be shown to users after proper authentication and authorization.

4.2 Internal Structure

The internal structure we would like to follow is one where each member will be a part of the engineering team and also in charge of another specific component of the product. The components we will assign roles to are: Marketing, Business Development & Partnerships, Finance, and Product Management. This way all team members can be a part of the implementation and maintenance of the application in addition to supporting the growth of the user base and organization operations.

4.3 Roles and Responsibilities

Implementation - All Team Members

Maintenance - All Team Members

Testing - All members

Product Management - Yikai Wang

Marketing - David Bravo

Business Development & Partnerships - Tasnim Nehal

Head of Finance - Qilei Cai

5. MANAGEMENT PROCESSES

5.1 Start-Up Plan

Cloud servers from AWS will need to be bought for deployment of the web app and some additional time will be required for the team to learn some web development frameworks but the development team has experience on working with front-end and databases.

5.1.1 Training Plan

Two members of the development team will require training for learning a web development framework for the system front-end which should take four to five weeks of time. However, most of the development team has knowledge on

back-end systems and designing databases meaning only this amount of training is necessary.

5.2 Work Plan

5.2.1 Work Activities

The work activities differ between the front and back end.

Front end activities:

- Provide an interface using web development UI framework for the user.
- Receive account credentials by the user and pass them to the back end. The risk factor is that the credentials can be intercepted if the transmission is not encrypted. A possible solution is to ensure HTTPS connection.
- Receive user donation payment information and send it to the bank end. The risk factor is bigger in the case where bank information can be stolen if transmission is not encrypted. HTTPS connection can be used to mitigate this problem.
- Receive and forward user search request to the back end.

Back end activities:

- Use algorithms to predict and recommend charities to the user. Rick factor is that weight factors in the prediction algorithm may produce inaccurate results. Extensive testing is needed to refine results.
- Receive and process user input into SQL code to access data from the database. Risk factor is that the user may attempt to use sql injection techniques. User input must be parsed before directly translating into sql code.
- Insert new entries into the database.

5.2.2 Schedule Allocation

There will be two groups working concurrently on different aspects of the system. The first group of two developers will be responsible for making a skeleton of the client-end services. This includes the front-end, general site layout, and forms. The second group of two developers will be responsible for the server components. This includes the back end, the database creation, the categorization algorithm for grouping charities, and data visualization features. The front-end features will be prioritized first and the database creation second so the server group can work on back-end features allowing for smooth integration of new functionality. The last responsibility will be deployment and server stress testing.

5.2.3 Resource Allocation

Resource	Work Activity	Skill Level of Personnel	Number of Personnel
Computer	Using computers for all software development tasks and logistical tasks	Entry level and above	20
Open-source software	Using open-source software such as Flask and Python to deploy the web app	Software engineers and more senior staff	10
Internet connection	Deploying the web app and managing the logistics of the platform, such as communicating with charities and donors	Entry level and above	20
Cloud servers	Deploying the web app onto cloud servers and enabling charities and donors to access the website	Software engineers and more senior staff	10

Summary of Resource Requirements

Computers for 20 staff members
Open-source software for 10 staff members
Internet connection for 20 staff members
Cloud servers for 10 staff members

5.3 Control Plan

The control plan details the procedures on how the budget, resources, schedule, and project requirements will be tracked and maintained.

5.3.1 Requirement Control and Traceability

Staff members are encouraged to propose changes to the product requirements when they deem them necessary for the continued development of the product. When staff members determine that certain phases of development cannot proceed as expected and that revising product requirements can address such issues, changes to product requirements are warranted and will be considered by the staff as a whole.

Staff members will be able to propose changes during regular meetings of the entire staff. They will elaborate on why the changes are warranted, how the changes will be implemented and what the potential ramifications are, namely how they will impact the on-time delivery of the product, the estimation of expenses, the resource allocation and the risk factors.

To assess the impact of the changes, staff members will estimate the time commitment required to implement such changes. For instance, replacing a proposed feature with a better alternative may increase or reduce the estimated number of work hours allocated to that feature. Staff members will discuss how work hours will be adjusted in response to such change.

Staff members will also evaluate whether the proposed changes impact the original estimation of expenses and resource allocation. For instance, replacing a feature may or may not require the purchase of additional resources. In the event that more resources need to be purchased, staff members will calculate such costs and revise the estimation of expenses accordingly. Alternatively, replacing a feature may eliminate the need for certain resources. Staff members will revise the plans on resource allocation to reflect such changes.

Staff members will evaluate whether the proposed changes magnify or reduce the risk factors. For instance, a proposed change to functionality may or may not have an impact on the security of the product. Staff members will evaluate the extent to which security risks will be magnified or reduced by such changes. If the risk factors are magnified by the proposed changes, staff members will weigh the tradeoffs and determine whether the proposed gains will outweigh the cons in terms of risk factors.

5.3.2 Schedule Tracking and Adjustment

The system will be broken up into its main features and members of the development team will be working on different features at the same time depending on their expertise. For example, a member can be working on newsfeed incorporation while others work on designing the database or the data visualization aspect. Each feature will have its own deadline and depending on the progress for each feature, deadlines will be adjusted accordingly.

5.3.3 Quality Control

Once one of the major system features is completed, the development team will be responsible for bug testing, security checks such as preventing SQL

injection, finding memory leaks, and optimizing runtime. If this current feature cannot be smoothly incorporated into the current system, then the members responsible for its design will be tasked to fix the highlighted issues before this process is repeated. Once the feature can be smoothly incorporated into the whole system, deadlines will be adjusted, and the associated developers can open themselves up to different responsibilities.

5.3.4 Metrics Collection Plan

The Givers would keep track of donations and location data of donators to provide meaningful statistics to users. This will be collected whenever a transaction is made. The donation amount, time of donation, and the charity name will be recorded either by an API (such as Shopify) or by the Flask application itself and stored in the database. If the user permits the website to obtain their location, that will be stored as well in the database along with the donation data. Users will be identified by their UUID to ensure privacy. Statistics about donations would be shown visually in the form of graphs to the user on their profile page, including frequency of donations, average donation amount, and types of charities they donated to. These statistics can be analyzed and validated using Python's modules, such as NumPy. Charities themselves can also compare their own collected data with the data of The Givers for validation.

5.4 Risk Management Plan

Some of the risks predicted to be faced are competition, low organization participation, malicious actors, and ability to fund operations. There are websites that already exist to rate and recommend non-profit organizations to everyday people such as Charity Navigator. However, this service does not allow for seamless subscription-based interactions between users and non-profits within the platform itself. Low organization participation will be a battle that can be handled by aggressive business development teams alongside teams dedicated teams to partnerships. Filtering for malicious organizations will be done by making the application and screening process of joining The Givers will be in place to avoid such organizations on the platform. Funding operations will be a challenge, so applying for grants, using as much open source software as possible, and having a focus on finance within the organization, through the head of finance position, will help The Givers help people connect and support non-profits that they truly care about.

6. TECHNICAL PROCESSES

6.1 Process Model

Upon completing features specified in the software requirements documents, at least one engineer needs to review the code implemented. If all tests pass and an engineer approves of the changes, the code is merged into the project and relevant functional requirements should also be verified through the application. Work will be divided into features and each feature will be further split into components that will include designs of front end, back end, and database structure. Upon the team reviewing the design proposals the engineers will then approve a proposal and carry out the implementation. The timelines will be estimated by engineers, including a buffer, and progress will be tracked using a Kanban board in a software operations product such as Jira. The timeline for the entire project will also be estimated by the engineers and each feature project termination will be defined before work starts on a given feature.

6.2 Methods, Tools, and Techniques

We plan to use a JavaScript front end library such as ReactJS to implement the front end. Use the Python library Flask for the back end. And finally use a SQL relational database. Throughout our development projects we plan to use GitHub to store our code repository, circle CI to run tests and build the software, and deploy our application to an AWS hosted web server. We plan to write tests when writing software that will include unit and integration tests. Once pushed as a pull request on the GitHub repo, CircleCi will run the tests of the software to ensure both unit tests and integration tests pass. If tests pass, the pull request can only be merged into the master branch if at least one reviewer has reviewed the request.

When proposing designs of components and features, we plan to employ various models to help visualize the workflow of information in the application. Upon having some users, we also plan to have frequent conversations with them to understand their needs and how our platform UI could improve user experience.

6.3 Infrastructure Plan

The hardware requirements are divided into two categories: the client end and the server end. The client end handles user input and relays it to the server

end. The server end runs the essential algorithms of the application and delivers outcomes to the client end.

Software Requirements
Client: a desktop/mobile browser that supports HTML5, such as Chrome, Firefox or Safari.
Server: support for HTML and CSS.

Networking Requirements
Client: An Internet connection with more than 2 Mbps of bandwidth.
Server: An Internet connection with a static IP and more than 500 Mbps of bandwidth.

Hardware Requirements
Client: any computer or mobile device that can run a modern-day browser, such as Chrome, Firefox or Safari.
Server: a six-core, 2.0 GHz processor or faster; 32 GB of memory or greater; 2 TB of solid-state drive or greater; a dual-port network interface; 500W power supply or greater.

7. SUPPORTING PROCESSES PLANS

7.1 Configuration Management Plan

GitHub is planned to be used as a version control system, and it has built-in services that provide configuration identification, controls, status accounting, integrations for evaluation, and release management that is consistent with the agile strategy planned to be deployed. Every piece of code sent into the review and testing process will have a branch title that is consistent with a task on the Kanban board. The Kanban board will keep track of all progress in our projects. When we release the product, we will make a new production branch that will trigger special build and deploy actions so that the new changes we made in the master branch are published into the formal product accessible to all users. When a change is needed, product and engineering should agree as to the significance and worthiness of a change before an engineer starts to implement it. If a change is simply a fix to accomplish the predetermined requirements of a piece of software, the engineer does not need group approval before implementing such a change.

7.2 Qualification (Verification and Validation) Plan

To ensure quality, in addition to tests for every piece of software implemented, design conversations before implementations, and peer reviews before merging code into the master branch, product validation will also occur once code is merged into the master branch. Through users and internal testing with the UI, the product can be further validated that it accomplishes predetermined functional user requirements. Conversations with users will also guide the engineering for how to improve the platform incrementally.

7.3 Documentation (library) Plan

Documentation will be written by the author of any piece of software implemented and will be reviewed by the engineering team as well. Documentation of team meetings will also be done so that if a team member is not available, the team can still have the meeting and the absent team member can know the content. Discussions written directly on a task in the Kanban board will also serve as documentation of conversations related to a given task.

Document	Assigned	Reviewer	Review Due
Code Documentation	Author of code	Team	24 Hours upon submission for review
Meeting notes	One team attendee	Team	Immediately after meeting
Kanban Board Task	Task Author	Team	To be referenced when reviewing associated implementation

7.4 Quality Assurance Plan

The Software Requirements Specification (SRS), Software Design Description (SDD), Software Project Management Plan (SPMP), Software Analysis Specification (SAS), Software product code will all be tracked. The team will also discuss the quality of the meeting notes, code documentation, and writing in tasks.

7.5 Reviews and Audits

When reviewing code, engineers should adhere to the guidelines set by IEEE which contain the following components:

- Requirements compliance
- Design compliance
- Documentation quality and relevance
- Test case effectiveness
- Test case results

Informal audits will be conducted on software documentation after initial reviews while engineers extend with existing features. Audits on meeting minutes will also occur as team members read them in the case they were absent or needed a reminder of what happened during the meeting. In order to ensure that tasks in the backlog are relevant, audits on the backlog will also be performed to remove tasks that may no longer be relevant.

7.6 Problem Resolution Plans

If a problem is discovered a bug fix task will be filed in the Kanban board and necessary engineers will be tagged and notified when the task is filed. If the change requires a major change in the architecture of the software or user facing changes will occur, the whole engineering team should be notified, and a meeting or conversation should be held over how to fix the issue. By writing comments and documenting meetings, the problem and solution will be documented. The problem's progress will also be tracked by creating a task in the Kanban board.

8. ADDITIONAL PLANS

None

9. INDEX

None

10. RATIONALE

None

11. NOTES

None

12. APPENDICES

12.1 Schedule Tracking

Artifact or Deliverable	Who (individual and team)	Estimated (hours)	Actual (hours)	Difference (hours)
SRS – Business Definition	David Bravo	4	3	1
	Qilei Cai	4	3	1
	Tasnim Nehal	4	4	0
	Yikai Wang	4	4	0
	Summary for entire team	16	14	2

Artifact or Deliverable	Who (individual and team)	Estimated (hours)	Actual (hours)	Difference (hours)
SRS – Requirements	David Bravo	2	1	1
	Qilei Cai	3	3	0
	Tasnim Nehal	3	3	0
	Yikai Wang	3	3	0
	Summary for entire team	11	10	1

Artifact or Deliverable	Who (individual and team)	Estimated (hours)	Actual (hours)	Difference (hours)
SRS – Complete	David Bravo	3`	3	0
	Qilei Cai	3	3	0
	Tasnim Nehal	3	2	1
	Yikai Wang	3	2	1
	Summary for entire team	12	10	2

Artifact or Deliverable	Who (individual and team)	Estimated (hours)	Actual (hours)	Difference (hours)
SRS – Version 1.3	David Bravo	2	2	0
	Qilei Cai	2	3	1
	Tasnim Nehal	2	2	0
	Yikai Wang	2	2	0
	Summary for entire team	8	9	1

Artifact or Deliverable	Who (individual and team)	Estimated (hours)	Actual (hours)	Difference (hours)
SPMP	David Bravo	5	4	1
	Qilei Cai	3	2	1
	Tasnim Nehal	3	3	0
	Yikai Wang	3	3	0
	Summary for entire team	14	12	2

Artifact or Deliverable	Who (individual and team)	Estimated (hours)	Actual (hours)	Difference (hours)
SPMP – Version 2.0	David Bravo	2	2	0
	Qilei Cai	3	3	0
	Tasnim Nehal	3	3	0
	Yikai Wang	3	3	0
	Summary for entire team	13	11	2

Cumulative

Who (individual and Team)	Estimated (hours)	Actual (hours)	Difference (hours)
David Bravo	19	16	3
Qilei Cai	18	17	1
Tasnim Nehal	18	17	1
Yikai Wang	18	17	1

12.2 Defect Tracking

Artifact or Deliverable	Who (individual and team)	Estimated (cases)	Actual (cases)	Difference (cases)
SRS – Business Definition	David Bravo	4	3	1
	Qilei Cai	4	4	0
	Tasnim Nehal	4	2	2
	Yikai Wang	4	2	2

Artifact or Deliverable	Who (individual and team)	Estimated (cases)	Actual (cases)	Difference (cases)
SRS – Requirements	David Bravo	3	1	2
	Qilei Cai	3	2	1
	Tasnim Nehal	3	1	2
	Yikai Wang	3	2	1

Artifact or Deliverable	Who (individual and team)	Estimated (cases)	Actual (cases)	Difference (cases)
SRS – Complete	David Bravo	4	1	3
	Qilei Cai	4	3	1

	Tasnim Nehal	4	2	2
	Yikai Wang	4	2	2

Artifact or Deliverable	Who (individual and team)	Estimated (cases)	Actual (cases)	Difference (cases)
SRS – Version 1.3	David Bravo	2	1	1
	Qilei Cai	2	1	1
	Tasnim Nehal	2	1	1
	Yikai Wang	2	2	0

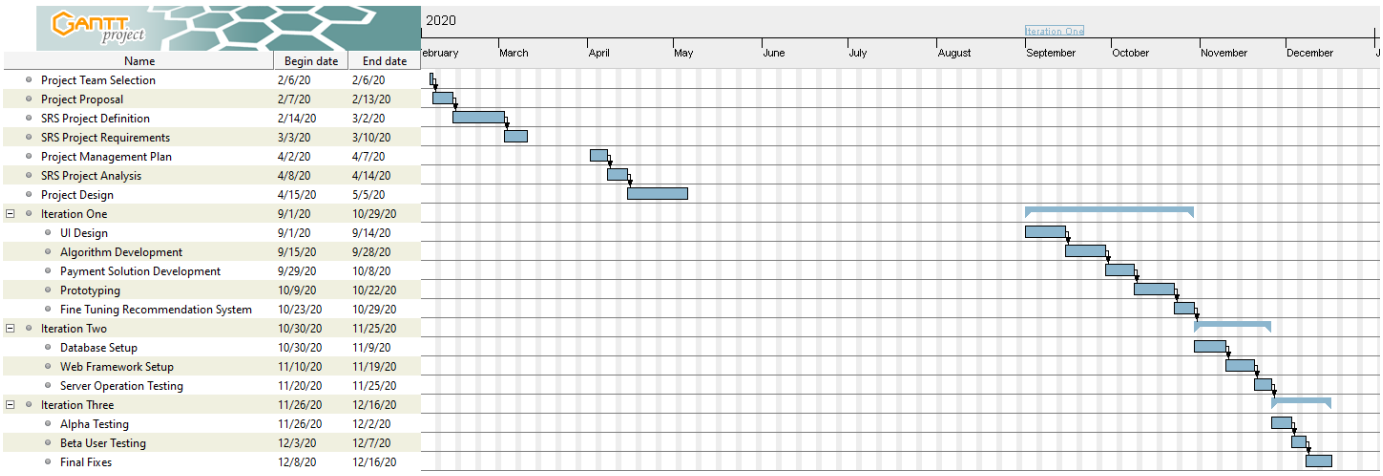
Artifact or Deliverable	Who (individual and team)	Estimated (cases)	Actual (cases)	Difference (cases)
SPMP	David Bravo	5	4	1
	Qilei Cai	3	1	2
	Tasnim Nehal	4	3	1
	Yikai Wang	4	1	3

Artifact or Deliverable	Who (individual and team)	Estimated (cases)	Actual (cases)	Difference (cases)
SPMP – Version 2.0	David Bravo	2	2	0
	Qilei Cai	2	1	1
	Tasnim Nehal	1	1	0
	Yikai Wang	3	2	1

Cumulative

Who (individual and team)	Estimated (cases)	Actual (cases)	Difference (cases)
David Bravo	20	12	8
Qilei Cai	18	12	6
Tasnim Nehal	18	10	8
Yikai Wang	20	11	9

12.3 Gantt Chart and Microsoft Project Schedule



Name	Begin date	End date
Project Team Selection	2/6/20	2/6/20
Project Proposal	2/7/20	2/13/20
SRS Project Definition	2/14/20	3/2/20
SRS Project Requirements	3/3/20	3/10/20
Project Management Plan	4/2/20	4/7/20
SRS Project Analysis	4/8/20	4/14/20
Project Design	4/15/20	5/5/20
Iteration One	9/1/20	10/29/20
UI Design	9/1/20	9/14/20
Algorithm Development	9/15/20	9/28/20
Payment Solution Development	9/29/20	10/8/20
Prototyping	10/9/20	10/22/20
Fine Tuning Recommendation System	10/23/20	10/29/20
Iteration Two	10/30/20	11/25/20
Database Setup	10/30/20	11/9/20
Web Framework Setup	11/10/20	11/19/20
Server Operation Testing	11/20/20	11/25/20
Iteration Three	11/26/20	12/16/20
Alpha Testing	11/26/20	12/2/20
Beta User Testing	12/3/20	12/7/20
Final Fixes	12/8/20	12/16/20