Computer Science and Engineering

Grickly

Project Management Plan

Version 1.0

Document Number: SPMP-001

Project Team Number: A20

Project Team Members (name and NET_ID): Ge Yang (gy622)
Hengning Zhang(hz1704)
Lujie Zhao (lz1860)
Sicong Liu (sl6728)

REVIEW AND APPROVALS

Printed Name and Title	Function (Author, Reviewer, Approval)	Date	Signature
Professor Strauss	Author	Fall 2020	
Ge Yang	Author	11/04/2020	On File
Hengning Zhang	Author	11/04/2020	On File
Lujie Zhao	Author	11/04/2020	On File
Sicong Liu	Author	11/04/2020	On File

REVISION LEVEL

Date	Revision Number	Purpose
11/04/2020	Version 1.0	Initial Release

Table Of Contents

1. OVERVIEW 5
1.1 PROJECT SUMMARY 5
1.2 PURPOSE, SCOPE, AND OBJECTIVES 5
1.3 ASSUMPTIONS AND CONSTRAINTS 5
1.4 PROJECT DELIVERABLES 6
1.5 SCHEDULE AND BUDGET SUMMARY 6
1.6 EVOLUTION OF THE PLAN 6
2 REFERENCES 6
3 DEFINITIONS 6
4 PROJECT ORGANIZATION 6
4.1 EXTERNAL INTERFACES 6
4.2 INTERNAL STRUCTURE 7
4.3 ROLES AND RESPONSIBILITIES 7
5 MANAGEMENT PROCESSES 7
5.1 START-UP PLAN 7
5.1.1 Estimation Plan 7
5.1.2 Staffing Plan 8
5.1.3 Resource Acquisition Plan 8
5.1.4 Training Plan 8
5.2 WORK PLAN 8
5.2.1 Work Activities 9
5.2.2 Schedule Allocation 9
5.2.3 Resource Allocation 9
5.2.4 Budget Allocation 10
5.3 CONTROL PLAN 10
5.3.1 Requirement Control and Traceability 10
5.3.2 Schedule Tacking and Adjustment 10
5.3.3 Budget Tracking and Adjustment 11
5.3.4 Quality Control 11
5.3.5 Reporting Mechanisms 11
5.3.6 Metrics Collection Plan 11
5.4 RISK MANAGEMENT PLAN 12
5.5 POST IMPLEMENTATION PLAN 12
6 TECHNICAL PROCESSES 12
6.1 PROCESS MODEL 12
6.2 METHODS, TOOLS, AND TECHNIQUES 13

6.3 INFRASTRUCTURE PLAN 13
6.4 PRODUCT ACCEPTANCE AND MIGRATION PLAN 13
7 SUPPORTING PROCESSES PLANS 13
7.1 CONFIGURATION MANAGEMENT PLAN 14
7.2 QUALIFICATION (VERIFICATION AND VALIDATION) PLAN 14
7.3 DOCUMENTATION (LIBRARY) PLAN 14
7.4 QUALITY ASSURANCE PLAN 15
7.5 REVIEWS AND AUDITS 15
7.6 PROBLEM RESOLUTION PLANS 15
7.7 ENVIRONMENT MANAGEMENT PLANS 15
7.8 PROCESS IMPROVEMENT PLAN 16
8. ADDITIONAL PLANS 16
9 INDEX 16
10 RATIONALE 16
11 NOTES 16
12 APPENDICES 17
12.1 SCHEDULE TRACKING 18
12.2 DEFECT TRACKING 20
12.3 GANTT CHART/MICROSOFT PROJECT SCHEDULE 21

1. OVERVIEW

1.1 Project Summary

The **motivation** of Grickly is to provide a unique way of socializing for active participants of life events. The user can join an event, form his/her interest groups, and hang out with all different kinds of people round the world.

The **purpose** of this SPMP is to provide a comprehensive overview of the purpose, scope, budget, timeline and deliverables of Gricky. It will also contain the project assumptions and constraints, as well a plan for evolving the project. Having such a plan will keep the official release of the application Grickly on track. This SPMP will serve as the guideline of application development, in terms of what needs to be done, how they will be implemented, and when they will get done.

The **intended audience** of this document are investors and potential Grickly users who are interested in knowing the progress of the project. Other intended audiences are the development team, such as the design team, software team, marketing team, and other people who are willing to participate in the project. This document will be the communication document for the project and will always be updated with the latest relevant information.

1.2 Purpose, Scope, and Objectives

Grickly's mission is to gather all active participants of life events. The user can meet new people while enjoying interesting events. This software provides a reliable way to help users match playmates and build their communities on the internet. The software includes a matching system with many parameters that can be configured by the users, such as different themes, interests, events, etc. Once the user inputs the time and event that he/she would like to participate in, a matching group will be generated automatically. The matching and pairing system has filters that allow a more accurate matching among target participants. People in the group can perform their events virtually or in the real world.

The event organizer can publish an event on Grickly, and the event will be prompted to users who might be interested. Once the user registered the event, he/she will be invited to the

event group and meet people before the event starts. Thus there will be less awkwardness and loneliness at the time of the event. Besides the matching, users can also take advantage of the chatting system included in the software to communicate and interact with others to build their communities. For organizers, they can start an event and target users with different tags or preferences to match groups.

With all functionalities introduced above, Grickly has limits on specific functions. For the matchmaking service, with the idea of meeting strangers to play games with, Grickly is not designed to connect users' friends whom they have already known. Thus, Grickly **cannot** find existing users. For the chatting system, Grickly **cannot** offer users online chat history saving to save resources. For the events, at this point Grickly **cannot** give the organizers the privilege to manage participants; in other words, organizers on Grickly **cannot** choose the participants who are willing to come.

Grickly provides a social platform that allows users to connect to each other and engage in wonderful events as a group. Although there are plenty of applications that perform quality communication with people we already know or largely one-on-one dating functionality, there is still a huge need in meeting new friends with similar interests, especially given the current situation that most people are working or studying from home. Currently, the business model of Grickly involves advertising public events for event holders, a one-time charged fee for posting public events and a percentage cut on each ticket sold, and Grickly Premium subscriptions. As the application evolves, the updated business model will be included in the future release.

1.3 Assumptions and Constraints

Assumptions

Grickly is based on the assumption that nowadays people prefer an efficient way for communication and relationship. There are many applications that arise these years to satisfy customers' needs for that. However, most of the applications are one-on-one dating, and there is a market for people who like hanging out as a group for events and interests.

People would love to meet more friends with the same hobbies, group for the same goals or gather for an interesting event.

Constraints

Some current constraints of Gricklys involve regulatory constraints, hardware constraints, and interface constraints. In terms of regulatory constraints, the event organizers will only be verified and authorized individuals and organizations. We do not provide service for users under 18 years old. Personal data may be collected to better deliver and improve our service. We may disclosure or close suspicious accounts and accounts that violate community communication protocols. The hardware interface will mainly be the PC web browser or mobile touch screen. For hardware constraints, the application can be opened on a web server from a PC, tablet, or mobile phone. Data transfer between the client and server will need stable cellular data or wifi connections and all data will be securely encrypted. Some interfaces to other applications include user authorization to Gmail, Facebook, WeChat, and purchasing to Eventbrite and Ticketmaster.

Unexpected changes to any part of the system environment may affect the requirements. For example, if user authentication is not accessible from Gmail, Facebook, or WeChat, we need to either change our way of authentication or only allow registration using a phone number. When planning the work and project cost, some margin must be left to prevent the risk of unexpected changes. If the project encounters major difficulties in development or faces a shortage of funds, the features with higher priority will be implemented first within the remaining time and budget, and other parts will not be performed until the problems are resolved and an updated SPMP is released.

1.4 Project Deliverables

The type of delivery that will be used for this project will be incremental deliverable. The project work will be broken down into smaller deliverables that will be completed dynamically. Before rebuilding and redeploying the program, the development team will make small changes and have small increments one at a time within the timeline of an iteration. The program can be built and tested at all times. There are four milestones to be hit: planning, finish of Alpha version, finish of Beta version, and the ultimate release. For this semester, the

product deliverables include the project proposal, SRS - Domain Definition, SRS - Projects Requirements, SMP - Project Analysis, and SRS - Project Analysis.

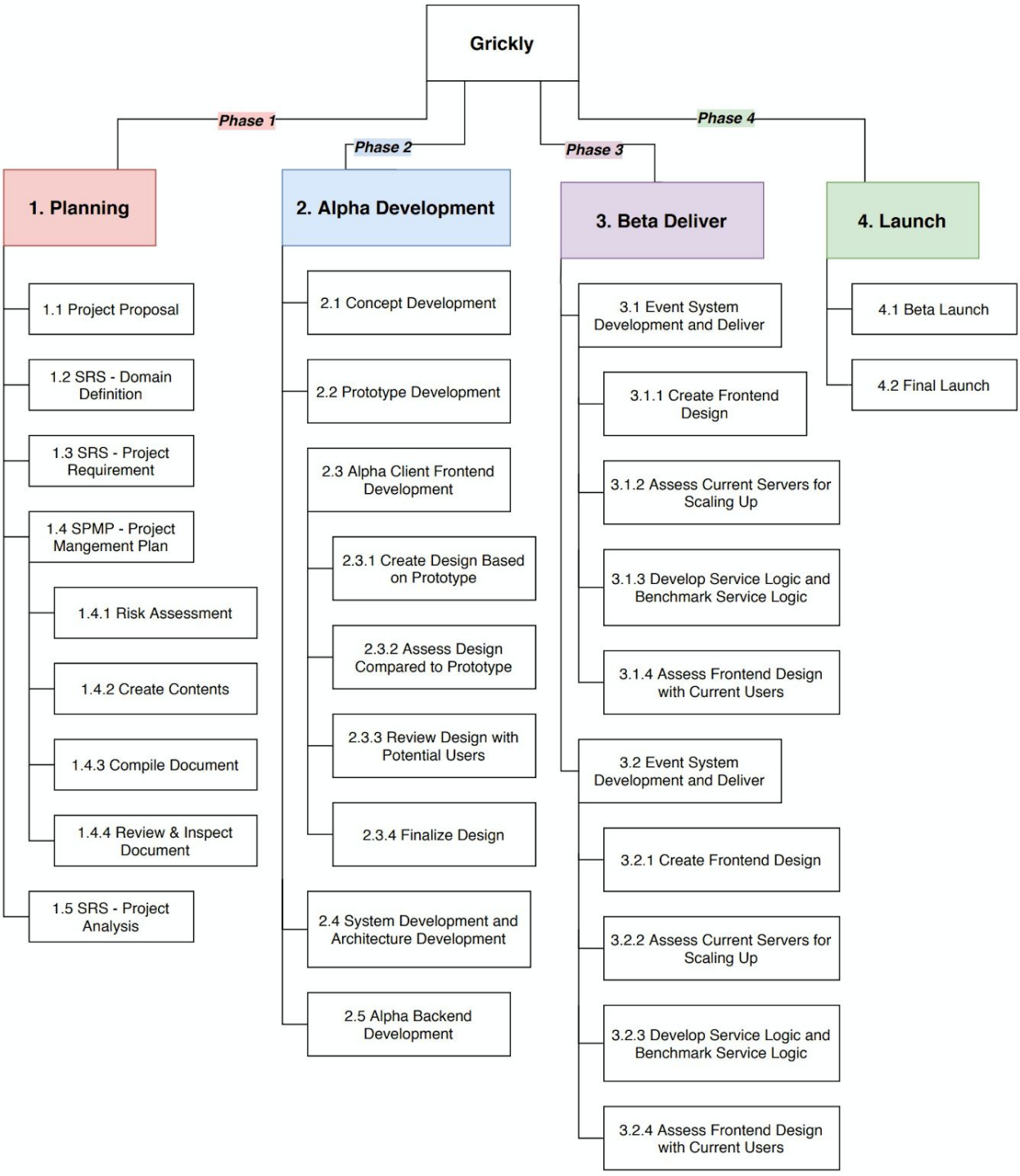
Products	Date	Organization	Quantity
Project Proposal	**10/08/2020**	**Software Development Team**	**N/A**
SRS - Domain Definition	**10/13/2020**	**Software Development Team**	**N/A**
SRS - Project Requirements	**10/21/2020**	**Software Development Team**	**N/A**
SPMP - Project Management Plan	**11/04/2020**	**Software Development Team**	**N/A**
SRS - Project Analysis	**11/17/2020**	**Software Development Team**	**N/A**
Alpha Prototype Testing	03/10/2021	Software Development Team	50
Beta Deliver	03/22/2021	Software Development Team	50
Event System Deliver	04/16/2021	Software Development Team	100
Event Matching Deliver	05/17/2021	Software Development Team	200
Beta Launch	05/31/2021	Software Development Team	5000
Full Function Launch	07/22/2021	Software Development Team	Unlimited

1.5 Schedule and Budget Summary

Schedule Summary

To illustrate our schedule, we outlined our development process in a Work Breakdown Structure (WBS) with four phases: Planning, Alpha Development, Beta Deliver and Launch. Testing will be held throughout the development process. For this semester, only the Project Planning phrase will be implemented. The top-level summary of the schedule in the WBS provided below. **Only the planning phase will be practiced during the current semester.**

Work Breakdown Structure of Gricky



Budget Summary

*This part is optional **(O)** and will be completed in the next release of SPMP after learning more class materials.* This subsection provides a top-level summary of the schedule and budget for

the project, The level of detail should be restricted to an itemization of the major work activities and supporting processes (for example milestones, and WBS) **Schedule (M) Budget (O)**.

1.6 Evolution of the Plan

Grickly, SPMP-001, Revision 1, 11/04/2020

For the initial release, team members have completed all mandatory sections that are marked with **(M)**. All optional sections**(O)** will be completed in the next release, which will also contain the implementation methods and specification of both scheduled and unscheduled updates. When the second release is published, the initial plan will be substituted by the newly released plan immediately. To control the amount of unexpected subsequential change, the unscheduled change in schedule and program should not exceed 30 percent of the entire project. All program updates will be done by the version control system GitHub for configuration management.

2. REFERENCES

Team A20, Project Proposal, Document Number, 001, 10/08/2020

Grickly, System Requirements Specifications, SRS-002, 10/21/2020

3. DEFINITIONS

Term	Definition
Distributed System	Distributed system is a system with multiple components located on computers in different networks. The components work together to achieve a common goal by communicating and acting in coordination.
GoDaddy SSL certificates	GoDaddy SSL Certificates protect users' privacy during transmission from their computer to your web server.

SMS notification	Out-of-band text messages sent in response to events or transactions which occur somewhere else.
Stability Test	The tests used to access a software's performance under intense usage or special cases.
UPS	Uninterrupted Power Supply.
AWS	Amazon web service, a cloud based computing solution by Amazon.
Azure	Microsoft Azure, commonly referred to as Azure, is a cloud computing service created by Microsoft.
ReactJS	React is an open-source, front end, JavaScript library for building user interfaces or UI components.
MongoDB	MongoDB is a document-oriented database program.
MySQL	MySQL is an open-source relational database management system by Oracle.

4. PROJECT ORGANIZATION

4.1 External Interfaces

The organizational boundaries are defined as the demarcation between the organization and its environment. The organizational boundaries of Grickly are the all sub-teams within the development team, including the algorithm team, the database (back-end) team, the frontend team, the marketing team and the recruitment team. The software quality assurance is an independent team that will be an external entity of the organization and interacting with the development. The software quality assurance team will review and validate the tasks that are submitted by the development team.

Some other external entities that Grickly project will interact with are applications that will be used as the authorization and ticketing charging. To create a profile, the user can authenticate using his/her Gmail, Facebook, or WeChat account or register using his/her phone number. The user can allow Grickly to access his/her Photos to import profile pictures and pull in contact information from Instagram and Contacts. If the user needs to purchase the ticket for a business event, he/she will be redirected to Eventbrite and Ticketmaster to make a purchase. All user interaction will be handled by our PC and mobile web interface through a touchscreen focused UI.

4.2 Internal Structure

Needs of the project aligns with the internal structure of this organization, consisting of a database (back-end) team, a front-end team, an algorithm team, a marketing team and a recruitment team. The database team is in charge of developing the distributed system that stores all the data. The front-end team is responsible for designing UI/UX and building the interface of the project. The algorithm team focuses on implementing the team that performs event and group matching. Currently, there is no member in the marketing team and recruitment team, and all team members will perform corresponding activities depending on their availability.

The main organizational entity that provides supporting processes for this project is GitHub for configuration management. The backend services will communicate within the server via the web service deployed on the servers, which will be from AWS or Azure. The project servers will be accessed by the development team members. For databases, we will be using MongoDB for better dockerized experience and faster response time compared to postGreSQL. More information about technology and tools we use can be found in Section 6.2 and software quality assurance can be found in Section 7.4.

4.3 Roles and Responsibilities

In terms of responsibilities within the team, all team members will actively engage in the project from every aspect, from developing to writing supportive documentation. They will share the technical responsibilities and people responsibilities throughout the entire developing process. For now, the marketing and recruitment activities will be performed by all team members, depending on their time availability.

In terms of roles for current team members, the team leader of the project Grickly is Sicong Liu, who is in charge of the upper management of the team. He will perform weekly meetings, measure the progress of the team, and assign tasks to the sub-teams and individuals. Hengning Zhang is the functional manager of the team and will supervise the development process of all team members while working as our head of the database sub-team. Ge Yang is the coach and the head of the algorithm sub-team. He will listen to the team members, make sure the project guidelines are followed, and resolve conflicts among the team members. Lujie Zhao is the project member, while serving as the head of the front-end sub-team. She will design the project releasing plan, keep track of budgets and schedule, monitor the performance of team members, and make public announcements of the development progress.

5. MANAGEMENT PROCESSES

5.1 Start-Up Plan

5.1.1 Estimation Plan

*This part is optional **(O)** and will be completed in the next release of SPMP after learning more class materials.* This subsection specifies the schedule and resources for completing the System/software project as well as policies, methods, tools, and techniques used in estimating cost, schedule, resources, and risk factors **(O)**.

5.1.2 Staffing Plan

*This part is optional **(O)** and will be updated in the next release of SPMP after learning more class materials.*

This project would require 3 or 4 experienced software engineers to lead the development of each component of the software. For each team we need 5 to 10 junior engineers (new grads) or interns to do the work led by the senior leaders. Existing staff includes 4 NYU Tandon students. We would hire some senior level software engineers from established companies. We would recruit interns and new grads from prestigious colleges and universities. Engineers' recruiting process would be like any other software company, including technical coding interviews and behavioral ones. We would also like to recruit social media influencers to host and join matchups and events. The main source would be social media.

5.1.3 Resource Acquisition Plan

*This part is optional **(O)** and will be updated in the next release of SPMP after learning more class materials.*

- Stipend of the project would be acquired by team members' donations and by sponsorship of NYU.
- Development equipment such as PCs would be needed for coding purposes and they would be brought to the project by the employees.
- Servers would be needed when getting the service online and they can be acquired from cloud services such as AWS and Azure.
- Training will be needed when new hires are onboard and it would be provided by the founding engineers.
- Food and working spaces would be needed when developers need to meet in person and these can be acquired around the school campus.

5.1.4 Training Plan

Writing:

- Academic writing: all team members will receive academic writing training from polytechnic writing center.

Technical:

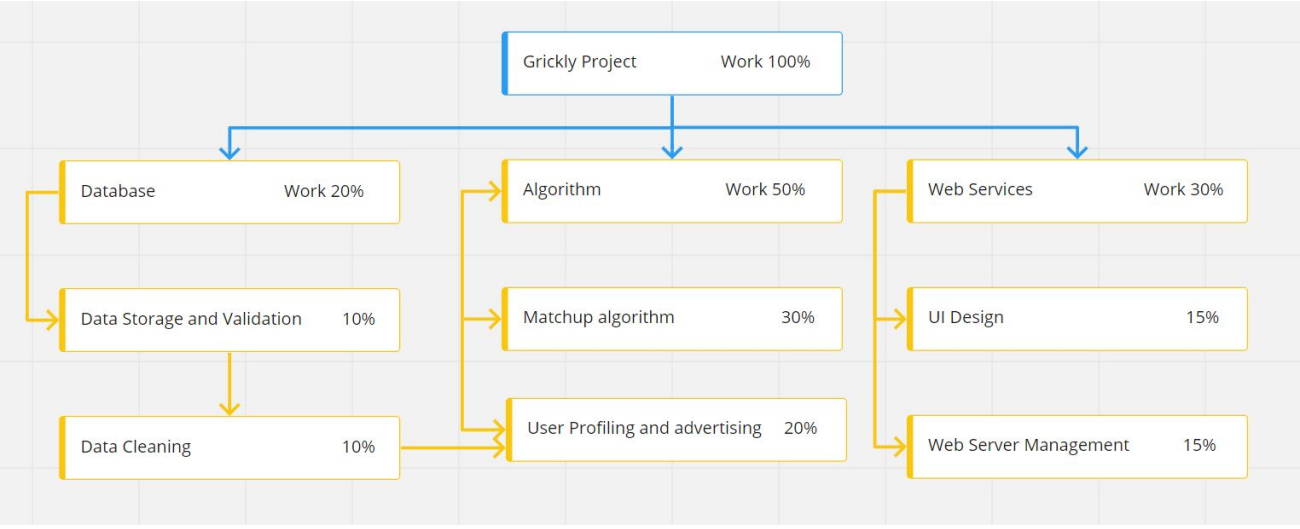
- Algorithms training: all back-end development staff need to receive this. This training requires basic coding abilities to enter and exit requires passing a test. It would be taught by our founding engineers.
- Front end design training: all front-end development staff need to receive this. This training requires basic web page development experience and exit requires showing the team a design project result. It would be taught by our founding engineers.
- Github training: all development staff need to complete the Github training to be familiar with source control and version control systems. Skills after training include but not limited to Github actions, version control, branches, merges and remotes.

Management:

- Communication training: all the staff need to receive this. There is no prerequisite for this course. All team members would receive evaluations from each other and those who pass can exit. Basic psychology and communication skills would be taught by our founding engineers.
- Presentation training: all the staff need to receive this. Team members should learn how to properly present and how to give effective and efficient feedback to the presenter.

5.2 Work Plan

5.2.1 Work Activities



WBS	Activity	Task	Assigned
1.0	SPMP		Team
1.1	Risk Assessment		Hengning Zhang
1.2	Create SPMP		
1.2.1		Write Section 6 Create Gantt Chart	Ge Yang
1.2.2		Write Section 1,4,12	Lujie Zhao
		Write Section 5	Hengning Zhang
		Write Section 7	Sicong Liu
		Rest of document	Sicong Liu
		Compile document	Sicong Liu
1.3	Review/inspect		SQA, Team
1.3.1		Rework	Team
1.4	Post/Delivery		Team
2.0	SRS-Domain		Team
2.1		Write Section 1,2.1,2.2,2.3,13.5	Ge Yang
2.2		Write Section 4,2.4,2.5,2.6,13.4	Lujie Zhao
2.3		Write Section 5.4,5.5,5.6,13.1	Hengning Zhang
2.4		Write Section 5.1,5.2,5.3,11,12,13.2	Sicong Liu
2.5		Compile document	Lujie Zhao
2.6		Review	Lujie Zhao
2.7		Post	Lujie Zhao
3.0	SRS-Requirements		Team

3.1		Write section 6.1	Ge Yang
3.2		Write section 8.1,8.2	Lujie Zhao
3.3		Write section 6.2	Hengning Zhang
3.4		Write section 7	Sicong Liu
3.5		Compile document	Hengning Zhang
3.6		Review	Hengning Zhang
3.7		Post	Hengning Zhang

- The risk factors can be found in **section 5.4**.
- Schedule durations are flexible as long as the milestones set by the whole team can be met every time. Estimated development time of each component is about 3 hours for each percentage of work.
- The acceptance criteria for all these components would be approval from the teammates in the quality control process. General scenarios will be designed at the start of the project to test the general functionality of each version. Random particular use scenarios may be imposed by team members to test whether the software can handle all possible edge cases. Stress tests and security tests may be done on servers to validate their reliability.

5.2.2 Schedule Allocation

- The work of writing the SPMP and SRS are assigned to each of the team members on day one. Deliver date of the deliverables and expected time needed to finish them are provided and specific schedule allocation is done by the individuals.
- The whole project team would be split into subgroups so that each subgroup can focus on their specific tasks. The algorithm component is the core of this project so it would need the most senior engineers and most time. A prototype of this should be delivered at the early stage of development so the other components can be built around it.

- UI design and database can be built concurrently while the senior engineers are building the algorithm.
- There would be a general schedule of project milestones where the work of each subgroup would be integrated to form a complete version of the software to be validated in the control plans. The teams can allocate their time according to each component's development time mentioned in **section 5.2.1.** They are responsible for delivering their assigned tasks on time.
- Milestones should be the delivery of database and UI design, the delivery of algorithms prototype, the integration of all these three and delivery of each new version after.
- Work can be done asynchronously in different groups, letting each subgroup have their own development schedule. Working process would be flexible so that the team may yield better results.
- Peer evaluation and cooperation at each milestone would help execute this plan.
- External constraints include user research for feedback on the UI design, external audits and advertising effectiveness.

5.2.3 Resource Allocation

Team Member	Task	Time
Hengning Zhang	SRS - Domain 5.4, 5.5, 5.6 & 13.1 SRS - Requirement 6.2 SPMP Section 5	20 hours
Sicong Liu	SRS - Domain 5.1, 5.2, 5.3 & 13.2 SRS - Requirement 7 SPMP Section 7	20 hours
Lujie Zhao	SRS-Domain Section 2.4, 2.5, 2.6 & 13.4 SRS - Requirement 8.1, 8.2 SPMP Section 1, 4, 12	20 hours
Ge Yang	SRS-Domain Section 2.1, 2.2, 2.3 & 13.5 SRS - Requirement 6.1 SPMP Section 6, Gantt Chart	20 hours

Subgroup	Personnel	Software/tools	Support

Database	Senior engineer: 1 Intern: 5	MySQL MongoDB	AWS
Front-end	Senior Engineer: 1 Senior Designer: 1 Intern: 5	HTML CSS ReactJS	AWS
Algorithms	Senior Engineer: 5	Python C++	None
Marketing	Product Manager: 1 Public relationship: 1	MS Office	Media Advertisements
Recruitment	Human Resource personnel: 1	MS Office	Media Advertisements

5.2.4 Budget Allocation

*This part is optional **(O)** and will be completed in the next release of SPMP after learning more class materials.* This subsection provides a detailed breakdown of necessary resource budget for each major work activity in the project work breakdown structure. The activity budget will include the estimated cost for activity personnel and may include, as appropriate, costs for factors such as meetings, software tools, development and test environments, and management support. A separate line item will be provided for each type of resource in each activity budget. The work activity budget may be developed using a spreadsheet presented in tabular form. **(O)**.

5.3 Control Plan

5.3.1 Requirement Control and Traceability

The team would meet about every 3 day and everyone's writing progress would be checked by each other.

Requirements would be reviewed and discussed at every team meeting. Leaders of the functionalities should describe how their developments are going on and show the other

team members their progress so they can be assessed. Then the result would be included in charts. Each team member would suggest requirement changes and then the suggestions would be compared to determine if better choices exist.

The impact of requirements change should be evaluated by the team leader on the cost, schedule, risk and effect. The procedures for a change request are:

1. Identify: identify and document the change
2. Validate: verify that the change is valid and modification is needed
3. Analyze: analyze the impact of the change including schedulem cost, risk and effect
4. Control: decide whether to make the change
5. Action: assign team or individuals to revise and modify
6. Close: validate the change is complete and close the request

5.3.2 Schedule Tracking and Adjustment

Progress schedule should be discussed at the start of this project. Milestones and checkpoints should be set to provide a quantified tracking method. Major milestones are identified at **section 5.2.2**. Actual progress would be compared to the scheduled progress at each team meeting. If work is not on schedule especially behind schedule too much that may not be able to meet the due date, schedule adjustments may be utilized. There are four steps to follow:

1. Analyze: analyze the document written and determine where need correction
2. Control: decide to choose what corrective actions (e.g. extension for due, more developers)
3. Calculate: recalculate the schedule according to the control
4. Close: validate the change on this work schedule and other work schedule affected

5.3.3 Budget Tracking and Adjustment

*This part is optional **(O)** and will be completed in the next release of SPMP after learning more class materials.* This subsection specifies the control mechanisms used to measure the cost of completed work completed, compare planned cost to budgeted cost, and implement corrective action when actual cost does not conform to budgeted costs. The budget control

plan will specify the intervals at which cost reporting will be accomplished and the methods and tools used to manage the budget. The budget should include frequent milestones. A mechanism such as earned value tracking should be used to report the budget and schedule plan, schedule progress, and the cost of work completed **(O)**.

5.3.4 Quality Control

Quality control milestones should be discussed at the start of this project. They can be used to assess quality assurance at each stage of the development. The qualifications should be done at each team meeting. Unqualified developments should be redone while qualified ones should be documented. Peer review would be used to determine whether each milestone and checkpoint has been reached. Details of the quality assurance plan can be found at **section 7.4**.

5.3.5 Reporting Mechanisms

*This part is optional **(O)** and will be completed in the next release of SPMP after learning more class materials.* This subsection specifies the reporting mechanisms, report formats, and information flows used in communicating project status within the project and to management and the customer. The methods, tools, and techniques of communication will be specified. The frequency and detail of the reports related to measurement and control should be consistent with the project scope, criticality, risk, and visibility **(O)**.

5.3.6 Metrics Collection Plan

The frequency of the collection would be once per week. Sample software will be given to users and peer developers so that we can validate the development progress. The satisfaction of randomly selected users and our developers combined would be the metrics that determines whether the project is successful or not.

Metric collection techniques include surveys, questionnaires, one-on-one interviews and observations.

Metric usages include:

1. Problem analysis: analyze the gap between the expected and actual result.
2. Process improvement: make adjustments on the process according to the metrics
3. Goal setting: goals and milestones are validated according to the metrics
4. Budgeting: if improvement is shown, more funding can be justified.

5.4 Risk Management Plan

The team members' enthusiasm and expectation towards the project would resolve the risks of inability to deliver. They would need to finish it in front of all the other team members if they fail to schedule their work on time.

Business Risk

- Description: People may not use our application if other established companies take our idea and implement similar functionalities.
- Probability: High
- How discovered: Feedback from potential users
- Responsible party status: other software companies
- Mitigation Plan: Develop exclusive functionalities and cooperate with event holders to offer exclusive events.

Operational Risk

- Description: This app may be used for illegal operations such as drug dealing and prostitution.
- Probability: High
- How discovered: Observation of other softwares
- Responsible party status: a portion of the users
- Mitigation Plan: We may be able to use natural language analyzers to censor such information or alert human admins to take care of it.

Technological Risk

- Description: The servers may be affected by power outage or natural disasters
- Probability: Medium
- How discovered: Research about servers
- Responsible party status: nobody
- Mitigation Plan: Use backup power like UPS for the servers to mitigate potential power outage hazards. Server locations should be chosen carefully, a well established server base would prevent natural disasters' hazards.

Economic Risk

- Description: The app may not work in special times such as pandemics.
- Probability: Medium
- How discovered: Observation of economy in pandemics
- Responsible party status: nobody
- Mitigation Plan: We would try to explore ways that can help people connect with each other without physical contact, like hosting events that allows users to come up with contactless events ideas and reward the best ideas' providers.

5.5 Post Implementation Plan

*This part is optional **(O)** and will be completed in the next release of SPMP after learning more class materials.* This subsection contains the plans necessary to ensure orderly termination of the software project. Items include plans for archiving project materials, baseline software deliverables, and post-mortem debriefings of project staff, and preparation of the Post Implementation Review (PIR) **(O)**.

6 TECHNICAL PROCESSES

6.1 Process Model

There are three milestones to be hit: finish of Alpha version, finish of Beta version, and the ultimate release.
During the development of Grickly, from the early development to the incremental development stages, developers should all follow the **life cycle** of: initiate, design, execute, assessment and closing processes.
During the early development of Grickly (before finishing the Alpha version), developers will follow the **waterfall model**. They will pass their code to a peer developer for code review. The review contains scanning for logic errors, syntax errors and redundant coding. After the code review, the original writer of the code will test the module by unit test: for backend with C++, the developer should be responsible for writing the unit tests while they develop certain functionalities; for frontend with JavaScript, the developer can take advantage of browser's Inspection function to test units one by one. All the code should be tested once they finish and pushed correctly onto the git repository. No user testing will be conducted at this stage. At the end of the phase, which should be after three months of the start of the phase, the developers should have an Alpha version deliverable for testing. The Alpha version should contain the functionality of basic matching for person and group posting.

The process model will transfer to **Incremental model** once the beta version is released (before finishing the Beta version). During this phase, all requirements for the developers will remain while the developers should also be responsible for reviewing the comments by the users who have tested the software. All the changes made for the comments should be precisely recorded by using comments in the code. In the end of this phase, all functionalities mentioned above should be implemented.

6.2 Methods, Tools, and Techniques

Grickly will be using incremental development overall to ensure users can engage in our core functions before other miscellaneous functionalities are available.

For project planning, the tools used are Microsoft Office and Google Apps Suite.

Since Grickly will mostly be a web-based software, we will be using more middleware to achieve both adaptability and running efficiency. The development can be broken down to two parts - frontend and backend. For frontend, we will be using **JavaScript** language with **React.JS** library for flawless user experience while maintaining low cost on adapting different

devices spanning from PC to mobile devices. For the backend, we will be using **C++** language to program service logic, and containerize different services into docker daemons for better risk containment and light deployment. For databases, we will be using **MongoDB** for better dockerized experience and faster response time compared to postGreSQL. The backend services will communicate within the server via the web service deployed on the servers, which will be from **AWS** or **Azure**, for their stability. When the users communicate with Grickly via web pages, the connection will be encrypted by **SSL** certificates signed and delivered by **GoDaddy**. All the services will be **distributed** across different servers and networks for easier risk containment.

All the codes in Grickly will follow Bottom-Up programming approach and will be object-oriented. This kind of methodology used will be easier to spot errors especially in a context where all services are divided into sub-services and deployed distributedly. All the codes will be stored on a single **Github** repository for version control.

6.3 Infrastructure Plan

For Frontend design, designers should all use Adobe Indesign to facilitate communications between developers and designers. Designers should document their thoughts and accomplishments by using a text file with the same name as the design file.

For Development, developers will use their own devices running the latest Ubuntu Linux to ensure the latest dependencies installed and security. All developers will be using Microsoft Visual Code to write and edit code files, and debuggers will be attached to Microsoft Visual Code by the developer himself. All the codes should comply with the industry standard about comments and spaces. All the codes written will be pushed with comments onto a private Github repository restricted to certain GPL/SSH key access. When developing simultaneously, it is required for the developers to decide if a branch is necessary. After development, testers should also be responsible for a testing branch on the git repository.

For testing, Grickly should be tried on most mainstream devices, including mobile phones, PC, and tablets. All tests should be conducted remotely from servers, meaning all testing devices should connect to the internet instead of local networks to ensure issues such as latency can be addressed.

Grickly will be using computing resources mostly hosted by **AWS** or **Azure**. The servers we will be renting from those platforms and install the same distribution of Linux to have same dependencies. They will be assembled into the same network to act like a cluster of servers serving background dockered services. Since they will be in one virtual network, all docker daemons will be able to communicate with each other as well. The daemon which is responsible for frontend responses will talk to the service logic daemon via a certain port on the server.

6.4 Product Acceptance and Migration Plan

*This part is optional **(O)** and will be completed in the next release of SPMP after learning more class materials.* This subsection specifies the plan for quality assurance and operations acceptance of the deliverable work product generated by the System/software software project. Objective criteria for determining acceptability of the deliverable work products are specified in this plan. Any technical processes, methods, or tools required for product acceptance are specified in the product acceptance plan. Methods such as testing, demonstration, and inspection should be specified in the plan **(O)**.

7 SUPPORTING PROCESSES PLANS

7.1 Configuration Management Plan

Configuration identification

For the purpose of configuration management, Grickly is divided into the following components:

- Database
- User interface
- Matching service
- Chat service
- Group service
- Profile service

All the components share the same version and release number. New versions released will have higher version numbers so that the customer can determine if they get the latest version.

Release management

A major release includes changes that modify or add functionality of the system, which makes the user interact differently. The changes are tested after the release in a certain period and user feedbacks are collected to ensure that new features are well rated. Every major release includes new documentation to instruct users. The version number of the software will be incremented before the decimal place (e.g. from version 1.2.3 to 2.0.0).

A minor release includes improvements and modifications of the existing features. These changes will also be tested after the release to ensure users do not need to change their experience using it. A minor release often does not need new documentation, and the version number is incremented after the first decimal place (e.g. from version 1.2.3 to 1.3.0).

A bugfix release includes a bug fix to allow users continue using the product. The version number is incremented after the last decimal place (e.g. from version 1.2.3 to 1.2.4).

Configuration control

- Source control: The changes to the components are tracked using a source control management system to maintain the history of all code and files.
- Version control: Each software developer has own repository to work. When their work is finished, each of their repositories is merged into the master repository in line. A release comes in one main branch in the master repository. When all the repositories are merged and changes are reviewed, the new version will be released.

Baseline

1. Build a baseline for a component

   A baseline is built when a new change of the component is ready for testing.

2. Prompt the new version

   When components are modified, the changes are checked by the version control system to avoid conflicting.

3. Maintain the components

The new components are kept in the source control system until it is deleted in the future.

Status accounting

CMDB(Configuration management databases) is used to track submission and changes. CMDB can provide visibility and accountability. All the records of the configuration items are maintained inside CMDB, and each request or change has a unique descriptor or identifier for trackability. the CMDB contains detailed information about the configuration items including the reason for the changes, the type of the change, and so on.

7.2 Qualification (Verification and Validation) Plan

Verification plan

- **Traceability:** a traceability analysis ensures that the software design fulfills all the requirements and all aspects should be traceable according to the requirements. The source code traceability is also important. The analysis should ensure that each specification is implemented in code and functions in code can be traced to the specifications.
- **Milestone review:** the milestone review is held by a meeting between customer representatives and the management team. All the participants need to review the materials before the meeting. During the meeting, the participants will go through the requirements and business case. The participants need to make the decision at the end of the meeting. The results of the phase can be accepted, partially accepted and not accepted.
- **Peer review:** when developers finish their own feature and want to merge their repository to the master repository, they need to open a pull request. At least two colleagues need to review the code and comment for modifications. After all the changes are approved and auto tested, the repository will be merged.
- **Prototyping:** To design a prototype, there is mainly four steps to follow:
    - Identify the requirements
    - Build the initial prototype
    - Test and feedback for the prototype

- Revise and repeat

Validation plan

- **Testing:** the testing process should include the following:
  - The expected result of the test should be predefined.
  - Testers should use different tools from the developers.
  - Tests should include unusual cases.
  - Test case should have a high probability of finding the error.
- **Analysis:** All test cases that do not pass should be analyzed. The analysis work list is predefined including the list of test cases. During analysis, analyzers should check the log to see if the test ran correctly. If the problem is due to invalid test case, the test case should be redone. For valid test cases, analyzers should try to find an explanation for the defect and mark it as “Analyzed” afterwards.
- **Inspection:** The inspection process is carried out by moderators, testers and developers. The moderator is responsible for the processThe responsibility of a moderator includes: manage the meeting, determine whether criterias are met, choose inspection participants. After the inspection meeting, the developer receives a report of the defects. After the defects are repaired, the moderator can conclude if all the defects are repaired or if another inspection meeting should be held.

7.3 Documentation (library) Plan

Documentation phases

The documentation process includes the following phases:

1. Research
2. Plan
3. Write
4. Review
5. Revise
6. Approval
7. Post

A technical writer and SQA(Software quality assurance) team are assigned in the team for writing and reviewing the documentation. The third and fifth phase are the responsibilities of the documentation writer, and other phases are responsibilities of the SQA and team.

Review process

- Informal reviews: when the writer finishes writing the draft, the developer reviews the draft and gives feedback for revision.
- Formal reviews: when the writer completes the final draft and is ready to upload the documentation, the document will be passed to the reviewers through at least two review cycles.
- Final Approval: after the document went through the formal review process, the approver will sign on the sign-off sheet.

Distribution list

Document	Tasks	Writer	Reviewer	Approver	Date
Project Proposal		Team	SQA, team	SQA, team	9/22/2020
SRS-Domain			SQA, team	SQA, team	10/8/2020
	Section 1,2.1,2.2,2.3,13.5	Ge Yang			
	Section 4,2.4,2.5,2.6,13.4	Lujie Zhao			
	Section 5.4,5.5,5.6,13.1	Hengning Zhang			
	Section 5.1,5.2,5.3,11,12,13.2	Sicong Liu			
SRS-Requirements			SQA, team	SQA, team	10/22/2020
	Section 6.1	Ge Yang			
	Section 8.1,8.2	Lujie Zhao			
	Section 6.2	Hengning Zhang			

	Section 7	Sicong Liu			
SPMP			SQA, team	SQA, team	11/6/2020
	Section 1,2,3	Ge Yang			
	Section 4,5	Lujie Zhao			
	Section 5	Hengning Zhang			
	Rest of document	Sicong Liu			
Architectural descriptions		Ge Yang	SQA, team	SQA, team	TBD
Design specifications		Ge Yang	SQA, team	SQA, team	TBD
Interface specifications		Ge Yang	SQA, team	SQA, team	TBD
Traceability metrics		Hengning Zhang	SQA, team	SQA, team	TBD
Test plans		Hengning Zhang	SQA, team	SQA, team	TBD
Meeting minutes		Hengning Zhang	SQA, team	SQA, team	TBD
Review reports		Lujie Zhao	SQA, team	SQA, team	TBD
Source code/ Executable code		Lujie Zhao	SQA, team	SQA, team	TBD
Manuals		Lujie Zhao	SQA, team	SQA, team	TBD
Online help		Sicong Liu	SQA, team	SQA, team	TBD
Regression test scenarios/scripts		Sicong Liu	SQA	SQA	TBD
Configuration library,		Sicong Liu	SQA	SQA	TBD
Online debugging/replay tool		Sicong Liu	SQA	SQA	TBD

7.4 Quality Assurance Plan

Product Quality

Two main factors are used to evaluate the product quality:

- How does it meet the requirements
- How well does it measure against established standards and guidelines

Process Quality

Process quality is defined as the degree to which an acceptable process is implemented and followed. The process quality objectives are:

- Ensure the processes are well defined and followed
- Define and agree on the criteria
- Ensure the processes measure and trace product quality

Standards and guidelines

The standards and guidelines will be used for development. Standards and guidelines can help team members deliver consistent quality and provide reviewers with criteria.

The following standards and guidelines will be followed by the project team:

- Programming standards and guidelines
- Test standards and guidelines
- Documentation guidelines (see **section 7.3** for details)
- Security standards and guidelines

Reviews and audits

It is important to assure quality throughout the reviews and audits. There are mainly three types for that:

1. **Walkthrough:** this is a type of informal review where members review each other's work and provide feedback.
2. **Inspection:** this is a formal review led by a moderator andis used to review technical specifications (see **section 7.2** for details).
3. **Review:** this is often referred as milestone review, is held by a meeting between customer representatives and the management team. The participants need to make the decision at the end of the meeting (see **section 7.2** for details).

Assessment

Assessment is a disciplined examination of the development process. It is performed in an open environment and is useful for improving the software quality.

The assessment includes three phases:

1. Plan and preparation: define the assessment scope, plan and prepare an initial document review
2. Conduct the onsite assessment: conduct meetings, interviews and presentations.
3. Report results: present the results and conduct an executive session.

The results of the assessment are documented, and reviewed.

7.5 Reviews and Audits

Polytechnic University management review

Management reviews can determine the need for change or improvement and whether policies and objectives are suitable. Both internal and external issues should be discussed in the management reviews as following:

Internal:

1. Policies, objectives
2. Assessment of risk management
3. Results of quality management system

External:

1. Regulations change
2. New marketing strategies
3. New services or activities

Developer peer review

Peer review is a type of review that a work product such as code and documentation is reviewed by the colleagues. This is an effective way and the first step to review an individual's work. The detailed process is explained in **section 7.2 and 7.3.** The schedule of the peer review is flexible.

Technical review

Technical review is a type of formal review that a team of developers examine the product according to the specifications and standards. It focuses on the technical quality of the product and it can suggest direct alterations to the product. The purpose of this is to improve

the technical aspect of the product by correction of the defects and recommendation of alternative approaches. A technical review is often scheduled before a milestone review. The reviews include a decision maker who determines if the objectives of the review are achieved, a review leader who leads the process of the reviews and a recorder who documents the review. After the review the developer should make the adjustments and submit to the review leader to decide if the quality is achieved.

Walkthrough

Walkthrough is a type of informal review with peers and managers. The objectives are to gain feedback from the technical team and make other team members familiar with the work. The schedule is flexible.

Inspection

The goal of the inspection is to find defects. In an inspection, a work product is selected and a team is gathered for the meeting. The phases for inspection are:

- Planning: moderator plans the inspection
- Overview meeting: author describes the work
- Preparation: inspectors examine the work
- Inspection meeting: inspectors give feedback
- Revise: author makes changes to the work
- Follow-up: changes are checked

Detailed process is specified in **section 7.2**.

Software audit

Software audit is a type of review that auditors who are not the member of developers conduct examination of the product. The objectives include providing independent evaluations of the product. The process includes contacting an audit team, meeting to explain the phases of the audit, data collection and reporting of the finding. The schedule of the audit is based on the milestone of the product.

7.6 Problem Resolution Plans

Reporting

A problem can be created whenever a team member finds a question, issue or condition that needs to be tracked to resolution. A problem has the potential to negatively affect project

components, including schedule, budget, resources, quality, releases and so on. The problem should be submitted as using Github with a unique identifier, a description, and category of the problem. A problem can be opened directly from a comment in an issue or a pull request review.

Analyzing

If the problem is only related to the individual's work, the developers can modify their code and go through the review process on their own. If the problem involves the whole team, it should be discussed in the weekly sync. The analyst should rank the issues as urgent and not urgent and also assign developers to solve the problem. Analysis includes:

- Assessing the results of delayed resolution on quality, budget and schedule.
- Assessing the impact of the problem on the whole project
- Determining the potential risks associated with the problem
- Determining possible responses for resolutions
- Providing a status update of the problem

Prioritizing

The problems are escalated according to the impact, urgency and cost.

- High priority: the problems are identified as directly related to project implementation and release, or key milestone.
- Medium priority: the problems are identified as having a strategic impact on the project
- Low priority: the problems are only related to individual's work

Closing

The problem can be closed when the problem no longer exists. The problem owner contacts the analyst and requests to close the issue. If agreed by the analyst, the problem will be added to the next meeting agenda for formal approval to close. After the problem is closed, the results and how it was resolved will be documented and other affected team members will be notified for the closure. On Github, the issue will be marked as closed.

Roles and responsibilities

Role	**Responsibilities**
Problem Originator	Identify problems
Analyst	Analyze and problems
Problem owner	refine issue and identify resolution plan

7.7 Environment Management Plans

*This part is optional **(O)** and will be completed in the next release of SPMP after learning more class materials.* This subsection contains plans for configuring and managing the development and testing environments. The development and test environment description and plans are specified in the plan for each environment The plans should include developer and management access to the automated tools described in relevant sections and subsection of the project

plan, **(O)**.

7.8 Process Improvement Plan

*This part is optional **(O)** and will be completed in the next release of SPMP after learning more class materials.* This subsection includes plans for periodically assessing the project, determining areas for improvement, and implementing improvement plans. The process improvement plan should be closely related to the problem resolution plan. (Each reported problem should be examined to determine the root cause of the problem, leading to a possible process improvement). This analysis can significantly reduce rework during the project life cycle. The goal of the process improvement plans is to enhance the effectiveness and efficiency of the project life cycle activities. Implementation of improvement plans should be examined to identify processes that can be improved without serious disruptions to the ongoing project and those which can best be implemented at the organizational level **(O)**.

8. ADDITIONAL PLANS

*This part is optional **(O)** and will be completed in the next release of SPMP after learning more class materials.* This section contains additional plans that may be required to satisfy

product requirements and agreements. Additional plans for a particular project may include plans for assuring that safety, security, performance, capacity, and special equipment are met. Other plans may include migration, maintenance, network, or product support **(O)**.

9 INDEX

10 RATIONALE

11 NOTES

12 APPENDICES

12.1 Schedule Tracking

Artifact or Deliverable	Who (individual or Team)	Estimated	Actual	Difference
Initial SRS - Requirement	Team	10/10/2020	10/13/2020	+3 Days
Initial SRS - Requirement	Ge Yang	10/10/2020	10/13/2020	+3 Days
Initial SRS - Requirement	Hengning Zhang	10/10/2020	10/13/2020	+3 Days
Initial SRS - Requirement	Lujie Zhao	10/10/2020	10/13/2020	+3 Days
Initial SRS - Requirement	Sicong Liu	10/10/2020	10/13/2020	+3 Days

Artifact or Deliverable	Who (individual or Team)	Estimated	Actual	Difference
Second SRS - Business Domain	Team	10/22/2020	10/21/2020	-1 Days

Second SRS - Business Domain	Ge Yang	10/22/2020	10/21/2020	-1 Days
Second SRS - Business Domain	Hengning Zhang	10/22/2020	10/21/2020	-1 Days
Second SRS - Business Domain	Lujie Zhao	10/22/2020	10/21/2020	-1 Days
Second SRS - Business Domain	Sicong Liu	10/22/2020	10/21/2020	-1 Days

Artifact or Deliverable	Who (individual and team)	Estimated	Actual	Difference
SRS - Analysis Complete	For each team member	N/A	N/A	N/A
SRS - Analysis Complete	Team	N/A	N/A	N/A

Artifact or Deliverable	Who (individual and team)	Estimated	Actual	Difference
SPMP	Team	11/05/2020	11/06/2020	-1 Days
SPMP	Ge Yang	11/05/2020	11/06/2020	-1 Days
SPMP	Hengning Zhang	11/05/2020	11/06/2020	-1 Days
SPMP	Lujie Zhao	11/05/2020	11/06/2020	-1 Days
SPMP	Sicong Liu	11/05/2020	11/06/2020	-1 Days

Cumulative

Artifact or Deliverable	Who (individual or Team)	Estimated	Actual	Difference
Initial SRS - Requirement	Team	10/10/2020	10/13/2020	+3 Days
Initial SRS - Requirement	Ge Yang	10/10/2020	10/13/2020	+3 Days
Initial SRS - Requirement	Hengning Zhang	10/10/2020	10/13/2020	+3 Days

Initial SRS - Requirement	Lujie Zhao	10/10/2020	10/13/2020	+3 Days
Initial SRS - Requirement	Sicong Liu	10/10/2020	10/13/2020	+3 Days
Second SRS - Business Domain	Team	10/22/2020	10/21/2020	-1 Days
Second SRS - Business Domain	Ge Yang	10/22/2020	10/21/2020	-1 Days
Second SRS - Business Domain	Hengning Zhang	10/22/2020	10/21/2020	-1 Days
Second SRS - Business Domain	Lujie Zhao	10/22/2020	10/21/2020	-1 Days
Second SRS - Business Domain	Sicong Liu	10/22/2020	10/21/2020	-1 Days
SPMP	Team	11/05/2020	11/06/2020	-1 Days
SPMP	Ge Yang	11/05/2020	11/06/2020	-1 Days
SPMP	Hengning Zhang	11/05/2020	11/06/2020	-1 Days
SPMP	Lujie Zhao	11/05/2020	11/06/2020	-1 Days
SPMP	Sicong Liu	11/05/2020	11/06/2020	-1 Days

12.2 Defect Tracking

Artifact or Deliverable	Who (individual or Team)	Estimated	Actual	Difference
Initial SRS - Requirement	Team	10/10/2020	10/13/2020	+3 Days
Initial SRS - Requirement	Ge Yang	10/10/2020	10/13/2020	+3 Days
Initial SRS - Requirement	Hengning Zhang	10/10/2020	10/13/2020	+3 Days
Initial SRS - Requirement	Lujie Zhao	10/10/2020	10/13/2020	+3 Days
Initial SRS - Requirement	Sicong Liu	10/10/2020	10/13/2020	+3 Days

Artifact or Deliverable	Who (individual or Team)	Estimated	Actual	Difference
Second SRS - Business Domain	Team	10/22/2020	10/21/2020	-1 Days
Second SRS - Business Domain	Ge Yang	10/22/2020	10/21/2020	-1 Days
Second SRS - Business Domain	Hengning Zhang	10/22/2020	10/21/2020	-1 Days
Second SRS - Business Domain	Lujie Zhao	10/22/2020	10/21/2020	-1 Days
Second SRS - Business Domain	Sicong Liu	10/22/2020	10/21/2020	-1 Days

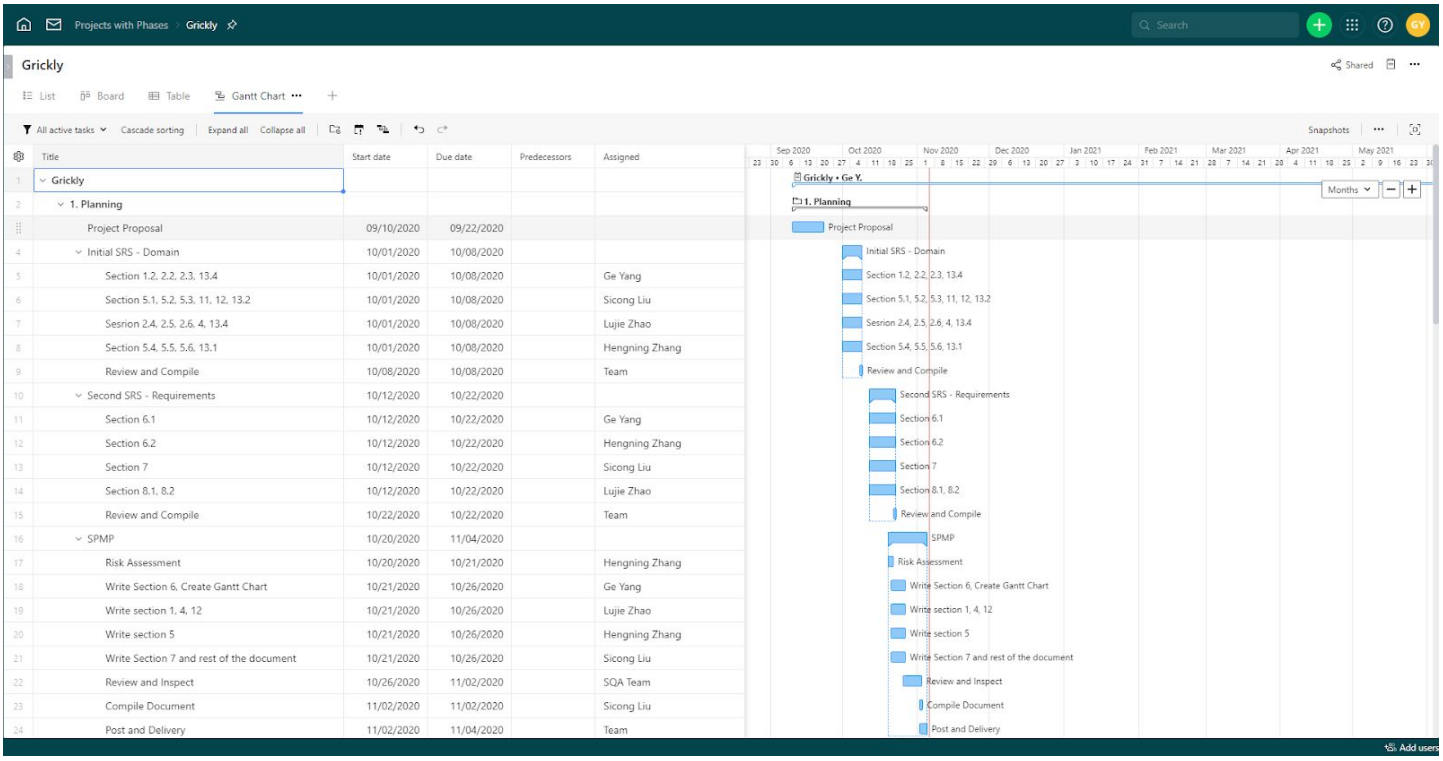
Artifact or Deliverable	Who (individual and team)	Estimated	Actual	Difference
SRS - Analysis Complete	For each team member	N/A	N/A	N/A
SRS - Analysis Complete	Team	N/A	N/A	N/A

Artifact or Deliverable	Who (individual and team)	Estimated	Actual	Difference
SPMP	Team	11/05/2020	11/06/2020	-1 Days
SPMP	Ge Yang	11/05/2020	11/06/2020	-1 Days
SPMP	Hengning Zhang	11/05/2020	11/06/2020	-1 Days
SPMP	Lujie Zhao	11/05/2020	11/06/2020	-1 Days
SPMP	Sicong Liu	11/05/2020	11/06/2020	-1 Days

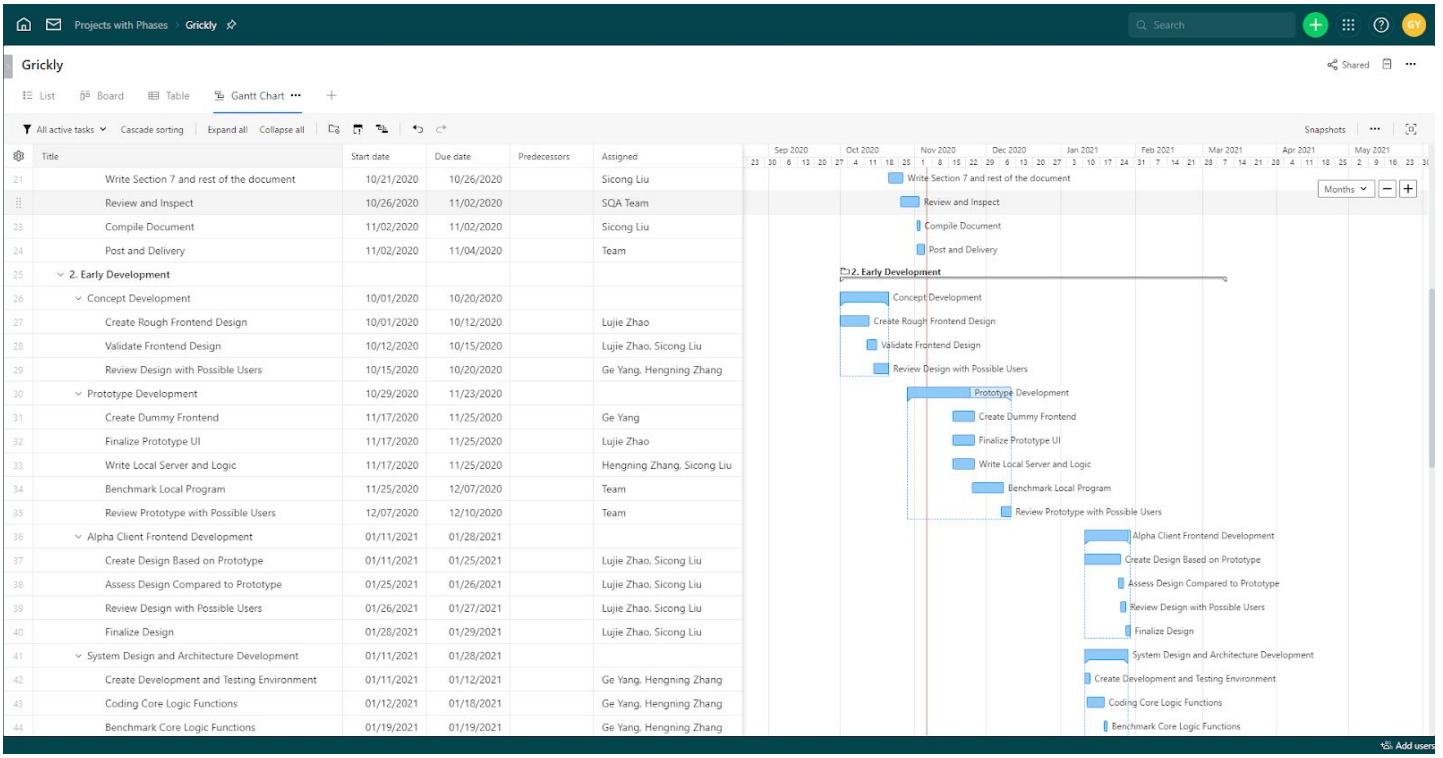
Cumulative

Artifact or Deliverable	Who (individual or Team)	Estimated	Actual	Difference
Initial SRS - Requirement	Team	10/10/2020	10/13/2020	+3 Days
Initial SRS - Requirement	Ge Yang	10/10/2020	10/13/2020	+3 Days
Initial SRS - Requirement	Hengning Zhang	10/10/2020	10/13/2020	+3 Days
Initial SRS - Requirement	Lujie Zhao	10/10/2020	10/13/2020	+3 Days
Initial SRS - Requirement	Sicong Liu	10/10/2020	10/13/2020	+3 Days
Second SRS - Business Domain	Team	10/22/2020	10/21/2020	-1 Days
Second SRS - Business Domain	Ge Yang	10/22/2020	10/21/2020	-1 Days
Second SRS - Business Domain	Hengning Zhang	10/22/2020	10/21/2020	-1 Days
Second SRS - Business Domain	Lujie Zhao	10/22/2020	10/21/2020	-1 Days
Second SRS - Business Domain	Sicong Liu	10/22/2020	10/21/2020	-1 Days
SPMP	Team	11/05/2020	11/06/2020	-1 Days
SPMP	Ge Yang	11/05/2020	11/06/2020	-1 Days
SPMP	Hengning Zhang	11/05/2020	11/06/2020	-1 Days
SPMP	Lujie Zhao	11/05/2020	11/06/2020	-1 Days
SPMP	Sicong Liu	11/05/2020	11/06/2020	-1 Days

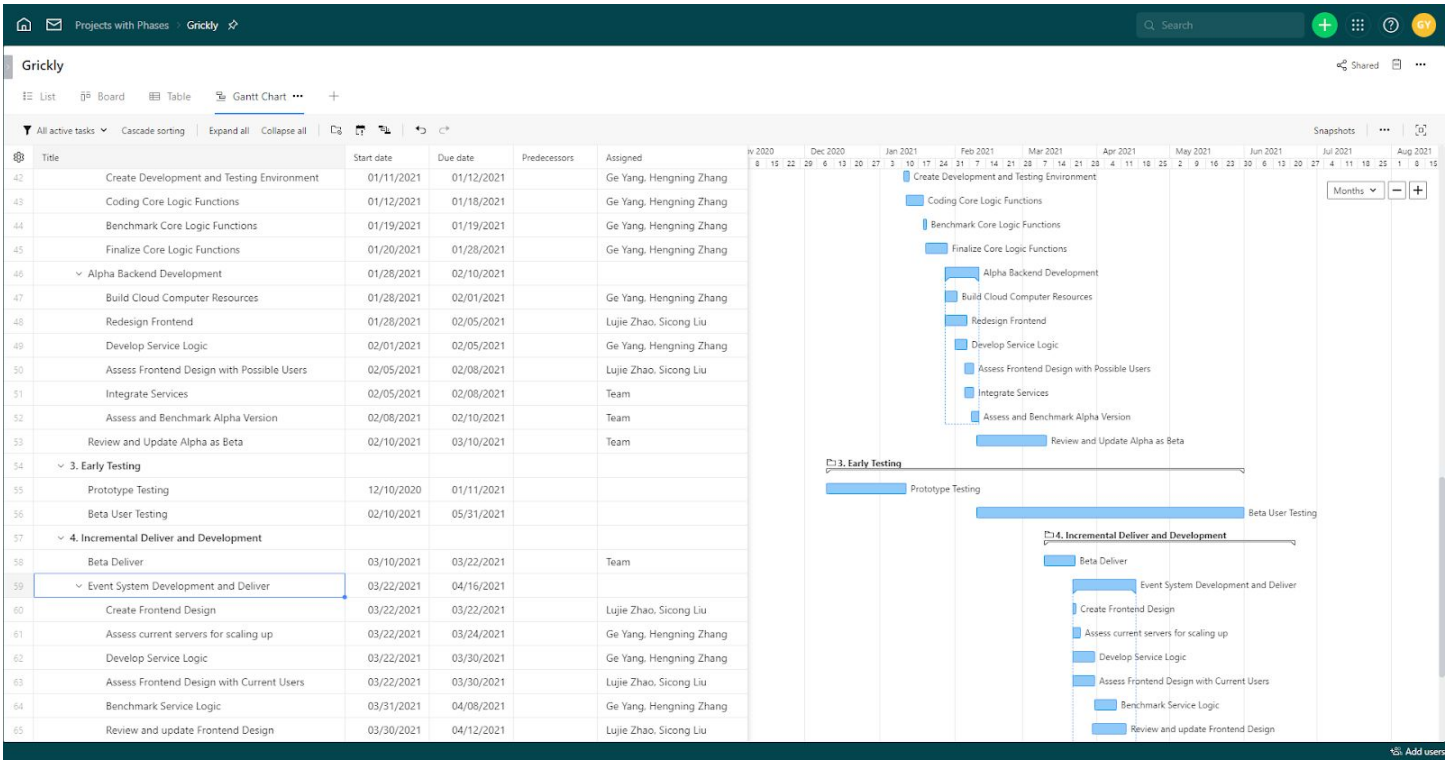
12.3 Gantt Chart/Microsoft Project Schedule



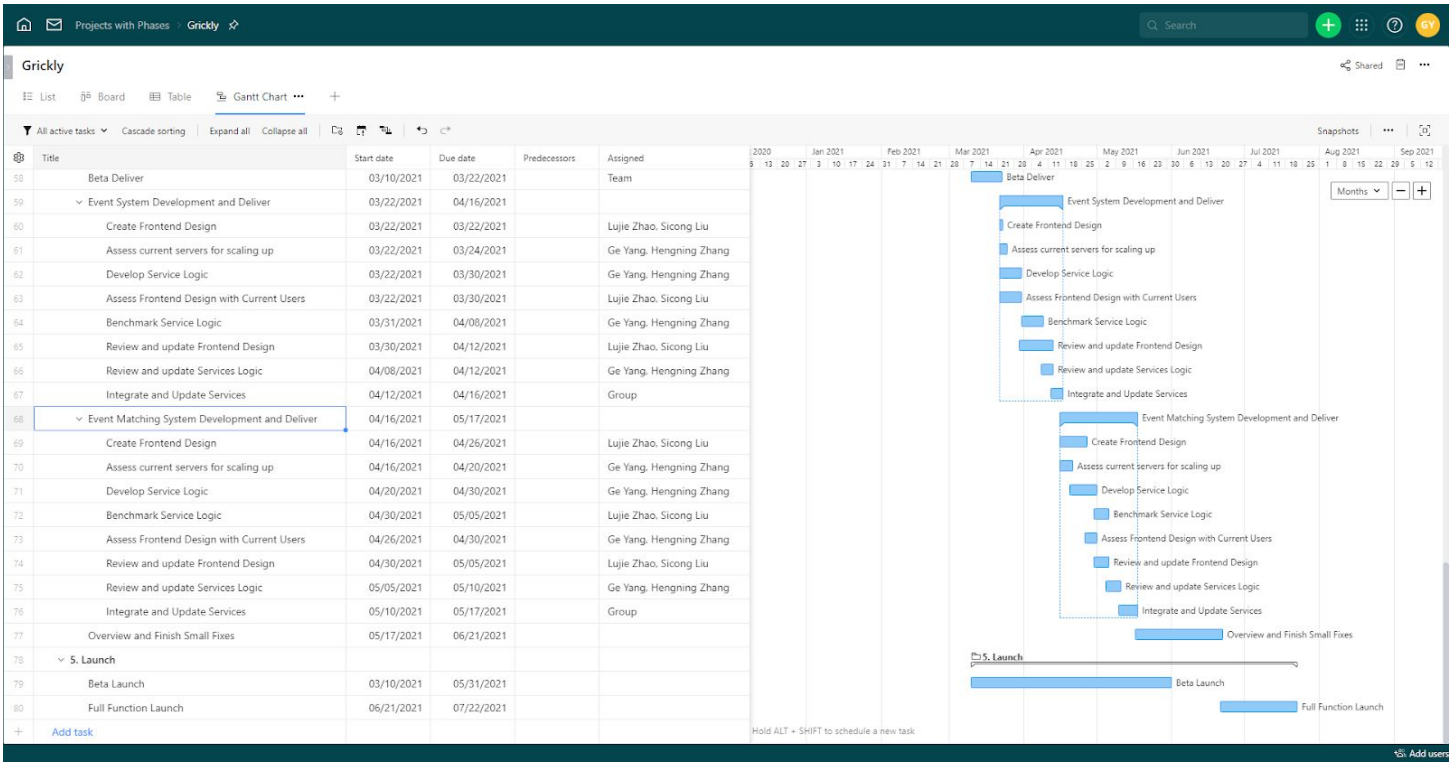
Title	Start date	Due date	Predecessors	Assigned
Grickly				
1. Planning				
Project Proposal	09/10/2020	09/22/2020		
Initial SRS - Domain	10/01/2020	10/08/2020		
Section 1.2, 2.2, 2.3, 13.4	10/01/2020	10/08/2020		Ge Yang
Section 5.1, 5.2, 5.3, 11, 12, 13.2	10/01/2020	10/08/2020		Sicong Liu
Sesrion 2.4, 2.5, 2.6, 4, 13.4	10/01/2020	10/08/2020		Lujie Zhao
Section 5.4, 5.5, 5.6, 13.1	10/01/2020	10/08/2020		Hengning Zhang
Review and Compile	10/08/2020	10/08/2020		Team
Second SRS - Requirements	10/12/2020	10/22/2020		
Section 6.1	10/12/2020	10/22/2020		Ge Yang
Section 6.2	10/12/2020	10/22/2020		Hengning Zhang
Section 7	10/12/2020	10/22/2020		Sicong Liu
Section 8.1, 8.2	10/12/2020	10/22/2020		Lujie Zhao
Review and Compile	10/22/2020	10/22/2020		Team
SPMP	10/20/2020	11/04/2020		
Risk Assessment	10/20/2020	10/21/2020		Hengning Zhang
Write Section 6, Create Gantt Chart	10/21/2020	10/26/2020		Ge Yang
Write section 1, 4, 12	10/21/2020	10/26/2020		Lujie Zhao
Write section 5	10/21/2020	10/26/2020		Hengning Zhang
Write Section 7 and rest of the document	10/21/2020	10/26/2020		Sicong Liu
Review and Inspect	10/26/2020	11/02/2020		SQA Team
Compile Document	11/02/2020	11/02/2020		Sicong Liu
Post and Delivery	11/02/2020	11/04/2020		Team



Title	Start date	Due date	Predecessors	Assigned
Write Section 7 and rest of the document	10/21/2020	10/26/2020		Sicong Liu
Review and Inspect	10/26/2020	11/02/2020		SQA Team
Compile Document	11/02/2020	11/02/2020		Sicong Liu
Post and Delivery	11/02/2020	11/04/2020		Team
2. Early Development				
Concept Development	10/01/2020	10/20/2020		
Create Rough Frontend Design	10/01/2020	10/12/2020		Lujie Zhao
Validate Frontend Design	10/12/2020	10/15/2020		Lujie Zhao, Sicong Liu
Review Design with Possible Users	10/15/2020	10/20/2020		Ge Yang, Hengning Zhang
Prototype Development	10/29/2020	11/23/2020		
Create Dummy Frontend	11/17/2020	11/25/2020		Ge Yang
Finalize Prototype UI	11/17/2020	11/25/2020		Lujie Zhao
Write Local Server and Logic	11/17/2020	11/25/2020		Hengning Zhang, Sicong Liu
Benchmark Local Program	11/25/2020	12/07/2020		Team
Review Prototype with Possible Users	12/07/2020	12/10/2020		Team
Alpha Client Frontend Development	01/11/2021	01/28/2021		
Create Design Based on Prototype	01/11/2021	01/25/2021		Lujie Zhao, Sicong Liu
Assess Design Compared to Prototype	01/25/2021	01/26/2021		Lujie Zhao, Sicong Liu
Review Design with Possible Users	01/26/2021	01/27/2021		Lujie Zhao, Sicong Liu
Finalize Design	01/28/2021	01/29/2021		Lujie Zhao, Sicong Liu
System Design and Architecture Development	01/11/2021	01/28/2021		
Create Development and Testing Environment	01/11/2021	01/12/2021		Ge Yang, Hengning Zhang
Coding Core Logic Functions	01/12/2021	01/18/2021		Ge Yang, Hengning Zhang
Benchmark Core Logic Functions	01/19/2021	01/19/2021		Ge Yang, Hengning Zhang



	Title	Start date	Due date	Predecessors	Assigned
42	Create Development and Testing Environment	01/11/2021	01/12/2021		Ge Yang, Hengning Zhang
43	Coding Core Logic Functions	01/12/2021	01/18/2021		Ge Yang, Hengning Zhang
44	Benchmark Core Logic Functions	01/19/2021	01/19/2021		Ge Yang, Hengning Zhang
45	Finalize Core Logic Functions	01/20/2021	01/28/2021		Ge Yang, Hengning Zhang
46	Alpha Backend Development	01/28/2021	02/10/2021		
47	Build Cloud Computer Resources	01/28/2021	02/01/2021		Ge Yang, Hengning Zhang
48	Redesign Frontend	01/28/2021	02/05/2021		Lujie Zhao, Sicong Liu
49	Develop Service Logic	02/01/2021	02/05/2021		Ge Yang, Hengning Zhang
50	Assess Frontend Design with Possible Users	02/05/2021	02/08/2021		Lujie Zhao, Sicong Liu
51	Integrate Services	02/05/2021	02/08/2021		Team
52	Assess and Benchmark Alpha Version	02/08/2021	02/10/2021		Team
53	Review and Update Alpha as Beta	02/10/2021	03/10/2021		Team
54	**3. Early Testing**				
55	Prototype Testing	12/10/2020	01/11/2021		
56	Beta User Testing	02/10/2021	05/31/2021		
57	**4. Incremental Deliver and Development**				
58	Beta Deliver	03/10/2021	03/22/2021		Team
59	Event System Development and Deliver	03/22/2021	04/16/2021		
60	Create Frontend Design	03/22/2021	03/22/2021		Lujie Zhao, Sicong Liu
61	Assess current servers for scaling up	03/22/2021	03/24/2021		Ge Yang, Hengning Zhang
62	Develop Service Logic	03/22/2021	03/30/2021		Ge Yang, Hengning Zhang
63	Assess Frontend Design with Current Users	03/22/2021	03/30/2021		Lujie Zhao, Sicong Liu
64	Benchmark Service Logic	03/31/2021	04/08/2021		Ge Yang, Hengning Zhang
65	Review and update Frontend Design	03/30/2021	04/12/2021		Lujie Zhao, Sicong Liu



	Title	Start date	Due date	Predecessors	Assigned
58	Beta Deliver	03/10/2021	03/22/2021		Team
59	Event System Development and Deliver	03/22/2021	04/16/2021		
60	Create Frontend Design	03/22/2021	03/22/2021		Lujie Zhao, Sicong Liu
61	Assess current servers for scaling up	03/22/2021	03/24/2021		Ge Yang, Hengning Zhang
62	Develop Service Logic	03/22/2021	03/30/2021		Ge Yang, Hengning Zhang
63	Assess Frontend Design with Current Users	03/22/2021	03/30/2021		Lujie Zhao, Sicong Liu
64	Benchmark Service Logic	03/31/2021	04/08/2021		Ge Yang, Hengning Zhang
65	Review and update Frontend Design	03/30/2021	04/12/2021		Lujie Zhao, Sicong Liu
66	Review and update Services Logic	04/08/2021	04/12/2021		Ge Yang, Hengning Zhang
67	Integrate and Update Services	04/12/2021	04/16/2021		Group
68	Event Matching System Development and Deliver	04/16/2021	05/17/2021		
69	Create Frontend Design	04/16/2021	04/26/2021		Lujie Zhao, Sicong Liu
70	Assess current servers for scaling up	04/16/2021	04/20/2021		Ge Yang, Hengning Zhang
71	Develop Service Logic	04/20/2021	04/30/2021		Ge Yang, Hengning Zhang
72	Benchmark Service Logic	04/30/2021	05/05/2021		Lujie Zhao, Sicong Liu
73	Assess Frontend Design with Current Users	04/26/2021	04/30/2021		Ge Yang, Hengning Zhang
74	Review and update Frontend Design	04/30/2021	05/05/2021		Lujie Zhao, Sicong Liu
75	Review and update Services Logic	05/05/2021	05/10/2021		Ge Yang, Hengning Zhang
76	Integrate and Update Services	05/10/2021	05/17/2021		Group
77	Overview and Finish Small Fixes	05/17/2021	06/21/2021		
78	**5. Launch**				
79	Beta Launch	03/10/2021	05/31/2021		
80	Full Function Launch	06/21/2021	07/22/2021		
	Add task				