NYU | TANDON SCHOOL OF ENGINEERING

Computer Science and Engineering

Happy Budget
Project Management Plan (SPMP)

Version 1.1

Document Number SPMP-002

Project Team Number: B27
Project Team Members: Amanda Lin (al5885), Gordon Lei (gl1776), Jason Li (jl8466), Jay Kang (yjk44)

REVIEW AND APPROVALS

<Team Members>	Function (Author, Reviewer, Approval)	Date	Signature
Amanda Lin	Author, Poster	11/05/2020	On File
Gordon Lei	Author	11/05/2020	On File
Jason Li	Author	11/05/2020	On File
Jay Kang	Author	11/05/2020	On File
Amanda Lin	Author, Poster	03/17/2021	On File
Gordon Lei	Author	03/17/2021	On File
Jason Li	Author	03/17/2021	On File
Jay Kang	Author	03/17/2021	On File

REVISION LEVEL

Date	Revision Number	Purpose
11/05/2020	Version 1.0	Initial Release
03/17/2021	Version 1.1	Updates/Defect Correction

Table of Contents

1. OVERVIEW

This section provides an overview of the purpose, scope, and objectives of the project, the project assumptions and constraints, a list of project deliverables, a summary of the project schedule and resources, and a review plan for this software project plan.

1.1 Project Summary

The Happy Budget system is an interactive educational personal finance web-based application that will provide an outlet to alleviate the learning gap for personal finances. The purpose of this document is to provide the software project management plan for the Happy Budget System by defining the system, project organization, management processes, technical processes, supporting processes plans, and additional plans. The intended audience of this document includes the client, the developers of the system, the software quality group, the project management team, the requirements writers, the auditing group, the operations group, and other related groups.

1.2 Purpose, Scope, and Objectives

The Happy Budget System is an interactive web-based application for personal budget planning and personal finance education. The system is intended for individual use by adolescents and young adults to provide an outlet to alleviate the learning gap for personal finances. The system will include the following functions:

- Connection to a personal bank account or user input of personal finance data
- Tracking and statistical analysis of money usage
- Interactive play
- Addition/Deletion, tracking, and statistical analysis of personal goals
- Financial tips

The system will take advantage of existing digital financial bank APIs to gather information about the user's standing balance and allow the user to access and use their money. In the case where this is not possible, the system will allow for user input of personal finance data. The system will not run on a new, different software, but on at least one type of web browser. Existing web application development languages such as Python will be used to develop the application. Additionally, existing visual statistics APIs will be used to provide visual statistics and existing database services to store financial tips and user information and data.

This release will satisfy business needs related to technology, economics, regulatory and legal, and market considerations. Technology allows this application to reach people in areas without proper facilities to receive quality financial education. The economic driver for the system is in the form of ad-revenue or a subscription-based system. The application must never compromise the privacy of the user, for ethical reasons as well as the legal complications it can ensue. The application must also make sure to follow the rules and guidelines of the financial institutions. The market audience of this program is primarily towards young adolescents and their parents who wish to develop good financial skills. The market the application can indirectly support is any market that deals with more expensive, long-use products, such as electronics, as the application encourages users to control constant, cheap spending for a long-term goal.

1.3 Assumptions and Constraints

The system will allow a connection to a personal bank account so that users will be able to see their total balance through this connection and interact with their money. Existing APIs for financial bank connections will be utilized and associated rules and guidelines of the institution will be abided by.

The application must track the user’s spending to provide quantitative data and inform the user about their spending habits visually. The user’s spending will be tracked through the financial bank APIs and the visual quantitative data will be displayed using a visual statistics API.

Interactive play to maintain the user’s interest and allow the user to gain first-hand experience will take the form of interactive pet interactions scattered throughout the system such as when the user puts money into their goal. These interactive pet interactions must be developed.

The application will support the addition, removal, and priority setting of personal goals that will provide first-hand experience in learning to save money towards a goal. A visual statistics API will be utilized to provide visual quantitative data on the goals progress.

Personal financing tips should also be provided throughout the system’s lifetime. These tips should be stored in a database for use throughout the system.

The system will not run on a new, different software, but on at least one type of web browser. Existing web application development languages such as Python will be used to develop the application.

The system must be completed within the schedule (See section 13.3) and must not exceed the determined budget (See section 6.1.1). The software in which the system will utilize are Javascript, Python with Django or Flask, OAuth 2.0, and PostGres for the back-end and HTML, CSS, Bootstrap, and Javascript for the front-end. The system will interface with a Visual Statistics Visualization API and, if possible, bank institution APIs to gather information about the user's standing balance and allow the user to access and use their money.

1.4 Project Deliverables

Software Analysis/Requirements Specification (SRS) – 02/17/2021

Software Project Management Plan (SPMP) – 03/01/2021

Software Design Document (SDD) – 03/08/2021

Implementation/Demonstration (code and full documentation) – 05/05/2021

Presentation – 05/10/2021

1.5 Schedule and Budget Summary

The software project team will consist of four individuals: Amanda Lin, Gordon Lei, Jason Li, and Jay Kang. The work will be divided amongst the team with Amanda as the project team manager and Jay keeping track of the project schedule.

The following is a brief overview of the software project schedule:

1. The project team selection form was started on February 7, 2021 and was submitted February 18, 2021

2. The project proposal was started on February 7, 2021 and was submitted on February 9, 2021

3. The SRS-Requirements and Analysis was started on February 7, 2021 and was submitted on February 19, 2021

4. The SPMP-Project Management Plan was started on February 7, 2021 and was submitted on March 17, 2021

5. The project Description was started on February 22, 2021 and was submitted on March 1, 2021

6. The SDD-Design Description (Initial) was started on February 22, 2021 and was submitted on March 8, 2021

7. OKR March 10 was started on February 15, 2021 and was submitted on March 10, 2021

8. Front end development was started on March 10, 2021 and will be completed by May 5, 2021
9. Back end development was started on March 10, 2021 and will be completed by May 5, 2021
10. Database development was started on March 10, 2021 and will be completed by May 5, 2021
11. The SDD-Design Description (Final) will be started on April 1, 2021 and will be completed by May 5, 2021

For each major work activity, risk assessment, work execution, review and inspection, and document submission will take place. Refer to the Gantt chart in 12.3 for detailed information.

2. EVOLUTION OF THE PLAN

Requirements of the system always change—whether after installation or during development, and through new hardware, legislations, regulations, budgetary constraints, compromises, etc—and as such, the SRS should evolve along with these changes. Requirements evolution pans out as follows:

1. An initial understanding of the problem is developed through a project proposal, where initial requirements are drafted.
2. Getting feedback from select users and clients will change the understanding of the problem which will result in a set of changed requirements.
3. These changed requirements will then be validated and redistributed.

A formal change process to identify control, track, and report projected changes is documented as follows:

1. The client is required to complete an engineering change notice on an uniquely-identified existing requirement whether the change is adaptive, corrective, or perfective.
2. Management process must be addressed as this notice then goes to the development team, where every change will need to be looked at in adequate detail, to factor in its effect on the cost, schedule, dependencies, relation to the other requirements, etc. These considerations will be sent back to the client, who will then make a decision to approve or disapprove the change.

3. If the change is approved by the client, it will be sent to the change control board, where it will be determined whether the requirement suits the business needs. Negotiations will be done between the development team, the client, and the control board on aspects like the release date and which iteration should the change be implemented in.

4. If the change is approved, it will be sent to the requirement writers, who will update the SRS, validate it, and redistribute it. If it does not pass validation, the change will be sent back to be revised.

Every requirement can be traced using its unique number through analysis, design, implementation, and the code itself. A database tool will be used to track every requirement number and where it is implemented in all of the artifacts going forward.

3. REFERENCES

1. Team B27, Happy Budget System Requirements and Analysis Specification (SRS), Document Number SRS-004, Version 4.0, 02/17/2020.

2. Team B27, Project Proposal for Happy Budget, Document Number 001, Version 1.1, 02/07/2021.

4. DEFINITIONS

Term/Acronym	Definition
API	Application programming interface
CI	Continuous integration
IDE	Integrated development environment
UML	Unified modeling language

5. PROJECT ORGANIZATION

This section identifies interfaces to organizational entities external to the software project, describes the project's internal organizational structure, and defines roles and responsibilities for the project.

5.1 External Interfaces

The software project team will interface with the development environment, databases, APIs, the client, the software quality group, the auditing group, and bank APIs.

5.2 Internal Structure

The software project team consists of four individuals: Amanda Lin, Gordon Lei, Jason Li, and Jay Kang. The team will discuss and work in a democratic setting and will have one team leader, Amanda.

The development team and the quality assurance team consist of these same four individuals and will operate under the previously described setting. Additionally, weekly meetings will be conducted to discuss what has been completed, what needs to be completed, and any next steps.

The project consultant will include Professor Fred Strauss.

5.3 Roles and Responsibilities

Role/Responsibility	Organizational Unit
Software project team leader/organizer	Amanda Lin
Schedule tracking	Jay Kang
Formatting and submission of software project deliverables	Amanda Lin
Review and editing of the software project deliverables	Amanda Lin, Gordon Lei, Jason Li, Jay Kang
Writing the project proposal document	Amanda Lin, Gordon Lei, Jason Li, Jay Kang
Writing the SRS	Amanda Lin, Gordon Lei, Jason Li, Jay Kang
Writing the SPMP	Amanda Lin, Gordon Lei, Jason Li, Jay Kang
Evolution of the software project including evolution of requirements	Amanda Lin, Gordon Lei, Jason Li, Jay Kang

Development of the software project	Amanda Lin, Gordon Lei, Jason Li, Jay Kang
Quality assurance of the software project	Amanda Lin, Gordon Lei, Jason Li, Jay Kang

6. MANAGEMENT PROCESSES

This section specifies the software project management processes for the project. This section includes the project start-up plan, risk management plan, project work plan, control plan, and post implementation review plan.

6.1 Start-Up Plan

This subsection specifies the estimation plan, staffing plan, resource acquisition plan, and training plan.

6.1.1 Estimation Plan

The general schedule plan for the project can be found in section 1.5. The general structure of the project is in three parts. Front-end, back-end, and database. For the front-end, the website will be hosted using AWS EC2. Chart js will be used to deploy the visual statistics function of the application. Plaid API will be utilized for the application's connection to user's bank accounts. Amazon RDS will be used for managing the application's databases.

In estimating the cost of the project, the only real cost of the project will be the necessary fees to pay to utilize the services mentioned in the above paragraphs, as staff will not be paid.

For creating the schedule of the project development, a general schedule was devised in a meeting of the staff members. Every week of development, the staff will hold a meeting to discuss the work done and manage and shift the remaining schedule based on unexpected delays or developments.

Resources necessary for the project were decided upon in initial planning and research. As the project development continues, a need for new services may arise. In that case, budget estimation and schedule will be changed accordingly.

Risk factors for the project were estimated initially before the start of the software development phase. Throughout the software development, there may be new risk factors arising from implementation of software or use of new resources.

6.1.2 Staffing Plan

The project will be managed and worked on by four staff members in total. The four staff are existing members of the team. The project phase/sections will be divided into four general parts. The four parts are front end development, back-end development, database development, and bug fixes/ testing. The front end will be developed primarily by a single staff member. The member should be proficient in CSS, HTML, and javascript. They should also have basic familiarity with back-end language and framework for easier communication with the back-end team. The back end will be developed by at least two primary members, with additional members to facilitate successful connection and integration between front end, back end, and database. The back-end team should be proficient in the programming language used for the project, which will mostly be Python. They should also be knowledgeable about the API used for facilitating some of the requirement functions for the application. The database will be developed primarily by a single staff member but will work closely with the back-end development staff. The database team should be proficient in use of Amazon RDS, the primary service used for database management for this project.

6.1.3 Resource Acquisition Plan

To successfully complete the project, access to AWS resources, access to programming environments, and staff training must be acquired. No new hires are needed for the completion of this project. Current staff must undergo training to be fluent in using AWS resources as well as the programming and database languages to be used. Training will be heavily dependent on accessible online material such as W3Schools. Access to programming environments will be acquired through use of free environments available for download online. Access to AWS resources will be acquired through contacting AWS for an AWS educate account with free credits.

6.1.4 Training Plan

Team members must be trained to be fluent in using AWS resources and programming and database languages that will be used in this project. The training will rely on free online resources such as public videos and public informational sites. All four members will undergo this self-training until they are fluent enough to understand the general syntax.

6.2 Work Plan

This section specifies the work activities, schedule, resources, and budget details for the software project.

6.2.1 Work Activities

Refer to the Gantt chart for this information.

6.2.2 Schedule Allocation

Refer to the Gantt chart for this information.

6.2.3 Resource Allocation

Work activity which involved written deliverables such as the Project Proposal, Project SRS, Project Team Selection Form, Project Management Plan, Project Description, SDD Design Description, and OKRs required participation of the four staff members. These staff members needed to be knowledgeable about Software Engineering, proficient in English, and have basic computer skills—which includes but is not limited to being proficient at using Google Docs, Microsoft Word, Microsoft Project (for gantt charts), draw.io (for diagrams), etc.

Developing the front-end requires one staff member to be proficient at CSS, HTML, and Javascript. Developing the front end requires access to AWS EC2.

Developing the back-end requires two staff members to be proficient at the languages used, which will primarily be Python. They will also need to have a firm understanding on how to use APIs. Developing the back-end requires access to AWS, Plaid API, and Chart.js.

Developing the database requires one staff member to be proficient at using Amazon RDS. Developing the back-end requires access to Amazon RDS.

6.2.4 Budget Allocation

Work Activity	Estimated Cost	Comments
Written Deliverables (SRS, SDD, SPMP, etc)	$0	Staff are not paid Software used (Google Docs, draw.io, Microsoft Project, etc) were used at no cost.
Front-end Development	$0	The cost of AWS EC2 is expected to be null with the assistance of New York University.
Back-end Development	$0	The cost of AWS is expected to be null with the assistance of New York University.

Database Development	$0	The cost of Amazon RDS is expected to be null with the assistance of New York University.

6.3 Control Plan

This subsection specifies the metrics, reporting mechanisms, and control procedures necessary to measure, report, and control product requirements, the project work schedule, budget, resources, and the quality of development processes and work products.

6.3.1 Requirement Control and Traceability

To keep product requirements manageable and trackable, each requirement will be given a number/tag. The application and its code will be commented to make sure that implementation of product requirements can be backward traced. During development, the team will hold weekly meetings to review product requirements and discuss any necessary change or addition to it. During the meetings, the team members will comment on current product requirements and put forth any suggestions. The other team members will review the suggestion and determine whether it would be beneficial or plausible for the edit to the product requirement to go through. Through the application's development cycle, there will be multiple prototypes. The prototypes will be used to confirm whether the product requirements are acceptable or in need of change.

6.3.2 Schedule Tracking and Adjustment

To keep track of the progress of work completed at major and minor milestones, the team will hold weekly meetings and an extra meeting before an important deadline. The meeting will be used to review the work done so far and to measure schedule progress. To measure schedule progress, the team will refer to the Gantt chart and the schedule tracking and defect tracking charts keeping track of the amount of work done. By comparing actual progress against the estimated progress the team should have made in the Gantt chart, the team can identify whether and how much the team is behind or ahead of schedule. In the case of being ahead of schedule, the team can spend extra time reviewing the work to make sure it is of high quality and error-free. In the case of being behind schedule, the team will make sure to adjust the schedule accordingly by either extending the estimated time before completion for tasks or putting in extra hours to get back on schedule. In the case that extra hours are necessary, the team will hold a short meeting to effectively and fairly divide up work. If a change is made, the project schedule will be adjusted accordingly and redistributed as per the evolution of the plan section (1.6).

6.3.3 Budget Tracking and Adjustment

As staffing costs are nonexistent and software tool costs are expected to be null with the assistance of New York University (NYU), the cost of all completed work completed is expected to be zero.

Nevertheless, this expected cost may change, perhaps in the event of the addition of a paid staff member, or acquisition of a deemed necessary software tool, or an NYU student software acquisition policy change. As such, there is need for control mechanisms for budget tracking and adjustment:

1. Earned value tracking will be used to report the budget and schedule plan, schedule progress, and cost of work completed (which are expected to be consistently zero).
2. The milestones established—making sure that actual cost conforms to estimated cost—will be visited every week of the project development duration.

3. In the event that a milestone is not met, the staff members will meet and discuss corrective action to be implemented. Majority agreement must be reached before the new budget is set and the budget tracking is adjusted accordingly.

6.3.4 Quality Control

Quality control will be achieved through a myriad of techniques such as quality assurance of work products, qualification, reviews, audits, and process assessment. To ensure that the system conforms to its specification and meets expectations of the system customer, the quality assurance team will use verification by conducting milestone reviews at each stage of the software process with the development team. The quality assurance team will also employ the use of automated tools like Selenium or Travis CI, to allow for testing, demonstration, analysis, and inspection. Additionally, jointly led peer reviews and code walkthroughs between the development and quality assurance team will be periodically conducted to find bugs in the program.

6.3.5 Reporting Mechanisms

The project status will be reported every week in team meetings. As outlined in 6.3.2 (Schedule Tracking and Adjustment) and 6.3.3 (Budget Tracking and Adjustment), both the status of the schedule and budget will be reviewed in these meetings. Changes made to the schedule or budget will be put in effect as soon as possible. For practicality, the project status will also be updated in its independent channel on the team Discord, as well as its page on the project's Github.

In the event that customers and or test users will need to be informed of the project status, a team email will be made. The status report, with only information relevant to these users, will be generated in PDF format before emailed to them.

6.3.6 Metrics Collection Plan

The scheduled work hours and defects will be kept track in a table where each member is responsible for filling out their own information. For each major milestone or artifact, each member is responsible for filling out the estimates for the amount of time they will spend for their work. The actual hours spent for each member will be recorded later. The team members should discuss how many defects they are expecting, and later report the actual number of defects produced.

6.4 Risk Management Plan

Risk Table:

Business Risk: “Going out of business”

Description: Members of the group might need to delay taking the software development class due to unforeseen circumstances, dismantling the team.

Probability: Very unlikely

How discovered: Communication between team members

Responsible Party: Any team members

Status: Not present

Mitigation Plan: Acquire new group members, join a new project

Operational Risk: GitHub Merge Conflicts

Description: When working on the same GitHub repo, it is possible that the repos on our local machine are not synced as people have pushed changes onto the repo. If anyone else tries to push their changes, a merge conflict may occur.

How discovered: Merge problems are always a consideration when working with GitHub

Probability: Likely

Responsible Party: Developers

Status: Not present

Mitigation Plan: Pull the repo before adding or committing to the repo or manually fix the merge conflict.

Technology Risk: Bank Account conflict

Description: Users may not able to connect their bank account to the application because the banks do not allow this or have a usable API that will allow them to do so.

Probability: Very likely

How discovered: Could be a potential problem as banks are usually hesitant to give out access to account information

Responsible Party: Banks

Status: Not present

Mitigation Plan: Users can input their own data into the system to simulate their bank account

Technological Risk: Database Security

Description: Because users will be making accounts, user account data must be stored and secured so their data will not get stolen.

Probability: Likely

How discovered: User data should always be protected.

Responsible Party: Developers

Status: Not present

Mitigation Plan: Developers can create a system of 2FA or OAUTH to make accounts more protected, as well as using salting user information.

6.5 Post Implementation Plan

Orderly termination of the software project will be ensured through the archiving of project materials, baseline software deliverables, and post-mortem debriefings of project staff. The archived materials will exist on the GitHub repository for the project. In preparation of the Post Implementation Review, the team members will assess OKRs developed throughout the project's lifecycle and evaluate whether the OKRs were successfully met, how effectively the project was completed, and what can be done better in the future. The team members will also measure the project's success against the project plan, budget, deadlines, and the quality of deliverables.

7. TECHNICAL PROCESSES

This section specifies the development process model, the technical methods, tools, and techniques to be used to develop the various work products, plans for establishing and maintaining project infrastructure, and product acceptance plans.

7.1 Process Model

The project process will be object oriented and will mainly follow a spiral model. The project will begin by researching the required tools and techniques required to create the software, as well as any background knowledge needed. After this phase is completed, the first cycle of the spiral will begin; developers will agree upon a set of objectives to complete in this spiral or cycle. After the completion of this phase, a set of risks and possible solutions will be determined by the development team. When the development team agrees on the aforementioned points, the development phase will begin followed by a phase for planning the next iteration of the project. Extensive planning will be done in between phases to save costs and time developing the project. The project will be deemed completed if it meets all the needs and requirements of the client, is usable, is delivered in a suitable time frame, and meets the budget of the client.

The establishment and maintenance of project infrastructure and other such product plans will be documented in Google Docs and will follow the UML. Review of design documents will occur weekly and, if needed, weekly meetings will be conducted regarding such review and other such administrative necessities. Work products and other such deliverables will be generated a week or weekend before the deadline to ensure proper testing before deployment. As of the first iteration of the software project management plan, major milestones to be achieved include: creation of a design document and system architecture, creating a baseline application where accounts can be made, creation of the database system to house relevant information, allowing users to input their own data, deployment of the first usable prototype, allowing users to connect their bank account, data visualization and other such data-related features, creation of a goal system and interactive play, and the creation of a system that handles giving appropriate tips to users. More deliverables and milestones will be to be determined by a later release. Approval of documents and work will be decided by the team before moving on to future stages.

7.2 Methods, Tools, and Techniques

The specification, design, development, configuration, integration, deliverance, modification, and maintenance of the project deliverables will follow the UML. The software project schedule will be documented and tracked through Microsoft Project. Documentation will be written and shared with developers in Google Docs and Github and will follow the UML. The development methodology is object-oriented and will be implemented with a combination of HTML, CSS, and Bootstrap to create a website front-end for the user. The backend of the software project will be determined in a later release; for the time being, it will be either made by using Node.js or Flask. The resulting development environment state will be shared among all developers through GitHub. Every developer will push either directly to add changes to the GitHub repository or, if a major change is made (for example the addition of a major milestone), will instead create a branch of the repository and request a pull request for other developers to review.

7.3 Infrastructure Plan

Communication for the software project development and maintenance will be conducted virtually through Discord and Zoom. The development environment state will be shared among the developers through GitHub. There will be no required IDE (excluding required testing tools to be determined at a later date) or text editor for developing code; developers are free to use what they are most comfortable with while developing the product.

During the beginning of the development process, the software project will be manually tested instead of using automated testing through manual creation of accounts and manual usage of features in prototypes by the developers. When prototypes reach a state that is more usable where features are more defined and usable, automated testing and continuous integration tools will be used. These tools may include using software testing tools such as Selenium or Travis CI, of which will be decided during the development process when prototypes are more developed and developers can see which tool is more useful for the current phase of development.

7.4 Product Acceptance and Migration Plan

Quality assurance and acceptance of deliverables will be constantly monitored and evaluated throughout multiple phases of the project depending on what deliverable they are. Deliverables will be constantly evaluated and compared against a checklist of desired results to see whether it meets an acceptable standard. For example, the acceptable standard for a prototype will be whether it meets the client's needs and requirements. Prototypes of the software will first be evaluated by the Software Quality group, testing for non-functional and functional requirements. If the prototype is accepted by the Software Quality group, it shall then be tested by the client to test for non-functional and functional requirements and pass on any questions or suggestions that will be passed to the development team in order to produce another prototype and deliverable; if deemed not acceptable the prototype will go back to the development phase and include any missing requirements and any suggestions made by the Software Quality group. Prototypes will also be tested for being user-friendly; some questions or possible qualities to check for are: is it clear how to navigate through the software, is the user-interface too lack-luster, are text too small, etc. Towards the final prototype, the product will also be tested with various browsers and browser sizes to give users more availability or accommodations depending on their workspace and system requirements.

Deliverables such as design documents or system architecture design will be evaluated by the team to figure out whether the document is readable, is presentable, contains the necessary and detailed information, and is understandable. Developers add any missing details or additional necessary information to the deliverables to reach these qualifications.

Testing and inspecting documents for product assurance will be mainly done manually or by people unless deemed necessary. Prototypes will be tested and inspected by people as well as by tools such as Cucumber and Selenium (the exact tools to be determined in a later release of the software project management plan or during development phase).

8. SUPPORTING PROCESSES PLAN

This section contains plans for supporting processes that cover the development life cycle of the development project. These plans include, but are not limited to, configuration management, software qualification (verification and validation), documentation, quality assurance, reviews, audits, problem tracking and resolution, and management. Plans for supporting processes will be developed to a level consistent with other sections and subsections of the project plan. Specifically, roles, responsibilities, authorities, schedule, budgets, resource requirements, risk factors, and work products for each supporting process will be specified.

8.1 Configuration Management Plan

Configuration identification, control, status accounting, evaluation, and release management consistent with Polytechnic University standards and policies will be achieved using a Github repository, which would allow for the storage and sharing of all documents. The baselining of work products will be managed through Github’s version control, which would allow for different versions of the product to be revisited and used as a logical basis for comparison. Logging and analysis of change requests will be also managed by Github’s version control—in this case, the change requests are the commits and pushes made by the developers from their local machines to a working branch. This procedure of not pushing directly to the main/master branch is necessary for change control, as only changes that are finalized, fully tested, and approved will be merged to the main/master branch. Regarding developer notification of when baselines are established or modified: developers will be briefed during team meetings on the goals for subsequent baselines and will be sent email notifications from Github every time the repository has been modified. Furthermore, developers will respect the GitHub workflow—they will pull before sending their changes to the remote repository to ensure that they have the newest version of the baseline(s).

8.2 Qualification (Verification and Validation) Plan

The scope of the qualification work activities by the quality assurance team is large and interwoven with the development team. While developmental activities are independent from qualification activities to an extent, there will be a periodic formal line of communication opened between the development and the quality assurance teams to allow for feedback. For example, to ensure that the system conforms to its specification and meets expectations of the system customer, the quality assurance team will conduct milestone reviews at each stage of the software process with the development team. Similarly, peer reviews will also be periodically performed with the development team, where members collaborate to find bugs in the program. All teams will continuously reference the software requirements specification when reviewing the software. Additionally, a set of responsibilities to ensure specification conformity must also be followed—which includes ensuring traceability of all documents. Prototyping, simulation, and modeling will also be conducted as deemed necessary—whenever the feasibility of design decisions and customer's requirements must be checked. Furthermore, the quality assurance team is also responsible for testing, demonstration, analysis, and inspection: the bulk of which will be accomplished through the findings of automated test tools, like Selenium or Travis CI, to run regression tests. These test tools will be utilized for component and system testing, which will largely target security assessment. While the quality assurance team is also responsible for customer testing, these tests will instead involve analysis of the users' needs based on feedback from sample customers.

8.3 Documentation (Library) Plan

System engineers will be responsible for internal work products like architectural descriptions, design specifications, interface specifications, and traceability metrics. Developers will be responsible for deliverable work products like source code, executable code, and configuration libraries. The quality assurance team will be responsible for regression test scenarios/scripts with the help of developers. Technical writers will be responsible for manuals, and on-line help and documentation. The corresponding documents will be generated by the respective teams, reviewed by the quality assurance team, and, if approved, will be sent to the control board for a final review. All documentation, including the code, will be uploaded on Github, and one other cloud-based service, like Google Drive. Additionally, local copies of all documentation will be stored on several local repositories, like the Git local repository.

8.4 Quality Assurance Plan

The quality assurance team will fully understand the software process, the software product as specified in the requirements specification, the software project plan, supporting plans, and any standards, policies, or guidelines to which the process or the product must adhere. Based on these understandings, the quality assurance team will conduct periodic analysis, inspections, reviews, audits, and assessments on all teams. For example, to ensure that the system conforms to its specification and meets expectations of the system customer, the quality assurance team will conduct milestone reviews at each stage of the software process with the development team. Additionally, the quality assurance team will create a test plan, test scenarios with expected outputs, execute the tests, and report any defects back to the development team for fixes.

8.5 Review and Audits

Project reviews and audits will be conducted on a periodic basis and involve checking the quality of project deliverables by checking software, its documentation, and records of the process to find violations of standards as well as errors and omissions. The different types of reviews as well as the schedule, resources, methods, and procedures in conducting them are as follows:

1. Program inspections and developer peer reviews involve a variety of team members with different backgrounds working together to find bugs in the program. Tasks such as incomplete versions of the system and representations such as UML models will be delegated. A line-by-line review of the program source code will be performed, and a checklist of common programming errors will be referenced to help find bugs.

2. Formal project technical reviews are conducted when there is an imminent problem with the project and negotiation, or when risk mitigation actions fail. In this scenario, an alternative approach to allow for the continuation of the project will be found—which includes whether or not the new approach is still aligned with the customer's goals. If a solution can not be found, the ultimate decision of this review may be to cancel the project.

3. Walkthroughs are a part of the review meeting phase and consists of the author of the program or document will go over it with the review team, with one team member sharing the review, and another formally recording all decisions made. Additionally, all actions agreed during the review should be signed by the review chair.

4. Management reviews are a part of the post-review activities phase and will be conducted if it is determined that the problems discovered need more resources.

5. Audits will be performed through the use of auditing systems placed strategically in code to detect any terms of a license.

8.6 Problem Resolution Plans

Developers, configuration management, change control, and qualification will work together to achieve problem resolution. Problems, whether discovered by the teams themselves or by another team dedicated to reviews like the quality assurance team, must be recorded in an issue tracking system, like Slack or Google Drive, and discussed in periodic meetings. A team will be given responsibility for fixing the defect, and each defect will have a unique identifying number. Qualifications tests to model the solution's feasibility will be conducted and the team assigned to fixing the defect will give updates every time they work on it as well as when they finish fixing it. Upon this fix, there will be a final review of the solution by the quality assurance team and the board before the change is approved to be merged into the main repository in Github by change control.

8.7 Environment Management Plans

Developers will use the same development and testing environments with the same configurations. The development environment will be shared among developers through GitHub whereas there will be no required IDE or text editor; developers are free to use what they are most comfortable with but the files and code used must match what is in the GitHub repository. There is no required OS or any required virtual machines/OS virtualization such as Docker. Required API keys and other such necessities will be shared in a .env file that will not be housed in the GitHub repository; this will be shared among developers by other means so that API keys are not public. Information that needs to be stored (such as account data) will be stored in a cloud database system. Developers will have access to and be able to add or modify any Makefiles or similar files for automated testing tools the team chooses to use.

8.8 Process Improvement Plan

Every week developers will meet to assess possible areas of improvement. Meetings will begin with discussing any problems members have during development and peers will share ideas on how to solve the problems. Afterward, a code review will be conducted by a group to make sure all developers know how the code functions as well as find any potential problems and any areas for improvement, whether it be to make the code more readable or optimize queries. Each problem will be examined carefully by all members to find the root cause of the problem. A discussion will be held regarding potential solutions and the best solution will be agreed upon by all members and implemented promptly.

9. ADDITIONAL PLANS

To be specified in a later release.

10. INDEX

11. RATIONALE

None.

12. NOTES

None.

13. APPENDICES

13.1 Schedule Tracking

Artifact or Deliverable	Who (Individual and Team)	Estimated	Actual	Difference
SRS	Amanda Lin	10 hours	10 hours	0 hours
	Gordon Lei	5 hours	5 hours	0 hours
	Jason Li	5 hours	5 hours	0 hours
	Jay Kang	5 hours	5 hours	0 hours
	Summary for entire team	25 hours	25 hours	0 hours

Artifact or Deliverable	Who (Individual and Team)	Estimated	Actual	Difference
SPMP	Amanda Lin	5 hours	7 hours	-2 hours
	Gordon Lei	5 hours	3 hours	+2 hours
	Jason Li	5 hours	3 hours	+2 hours
	Jay Kang	5 hours	5 hours	0 hours
	Summary for entire team	20 hours	18 hours	+2 hours

Cumulative

Who (Individual or Team)	Estimated	Actual	Difference
Amanda Lin	15 hours	17 hours	-2 hours
Gordon Lei	10 hours	8 hours	+2 hours
Jason Li	10 hours	8 hours	+2 hours
Jay Kang	10 hours	10 hours	0 hours

Summary for entire team	45 hours	43 hours	+2 hours

13.2 Defect Tracking

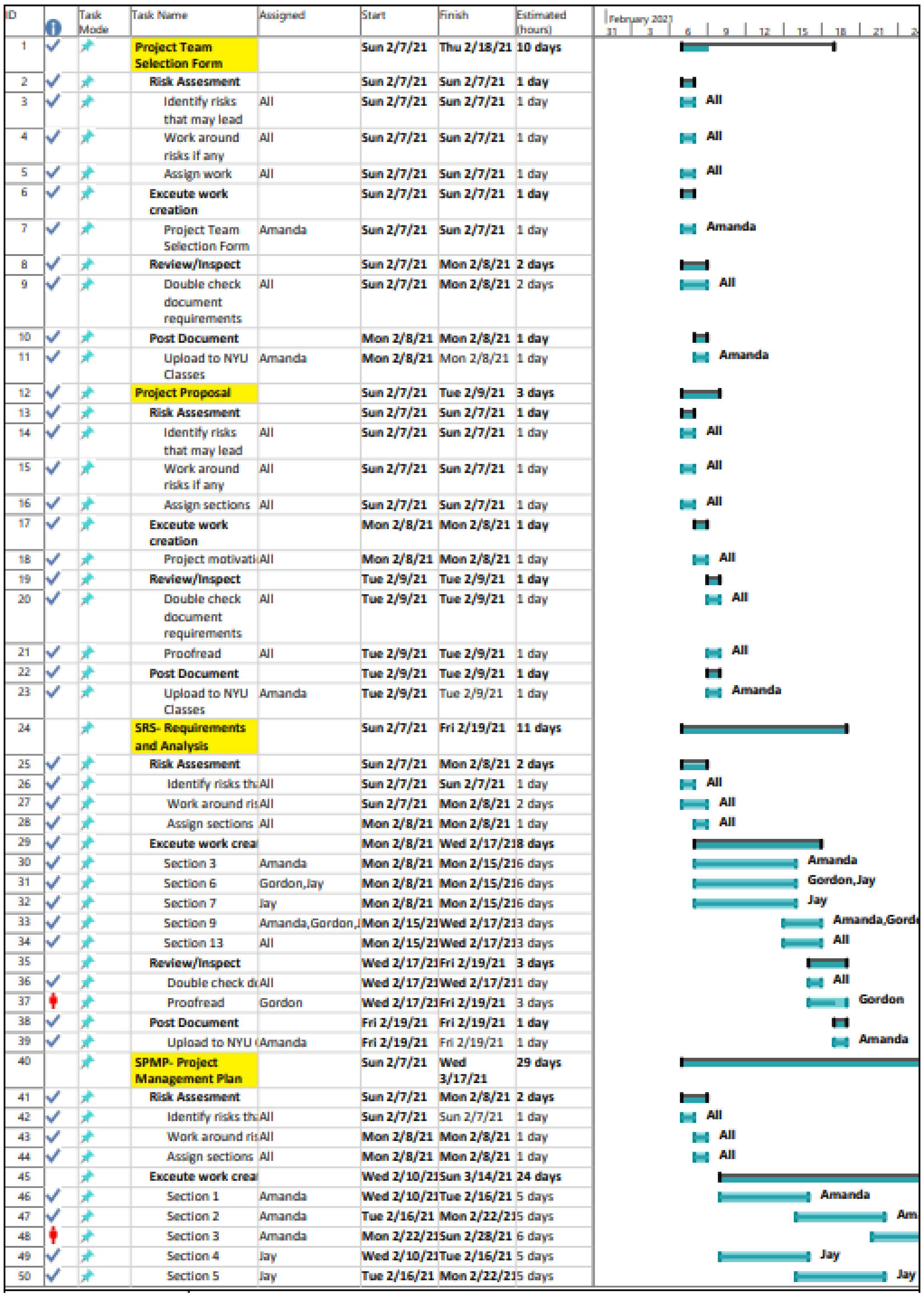
Artifact or Deliverable	Who (Individual and Team)	Estimated	Actual	Difference
SRS	Amanda Lin	40	16	-24
	Gordon Lei	20	2	-18
	Jason Li	20	10	-10
	Jay Kang	20	1	-19
	Summary for entire team	100	29	-71

Artifact or Deliverable	Who (Individual and Team)	Estimated	Actual	Difference
SPMP	Amanda Lin	10	5	-5
	Gordon Lei	10	3	-7
	Jason Li	10	3	-7
	Jay Kang	10	5	-5
	Summary for entire team	40	16	-24

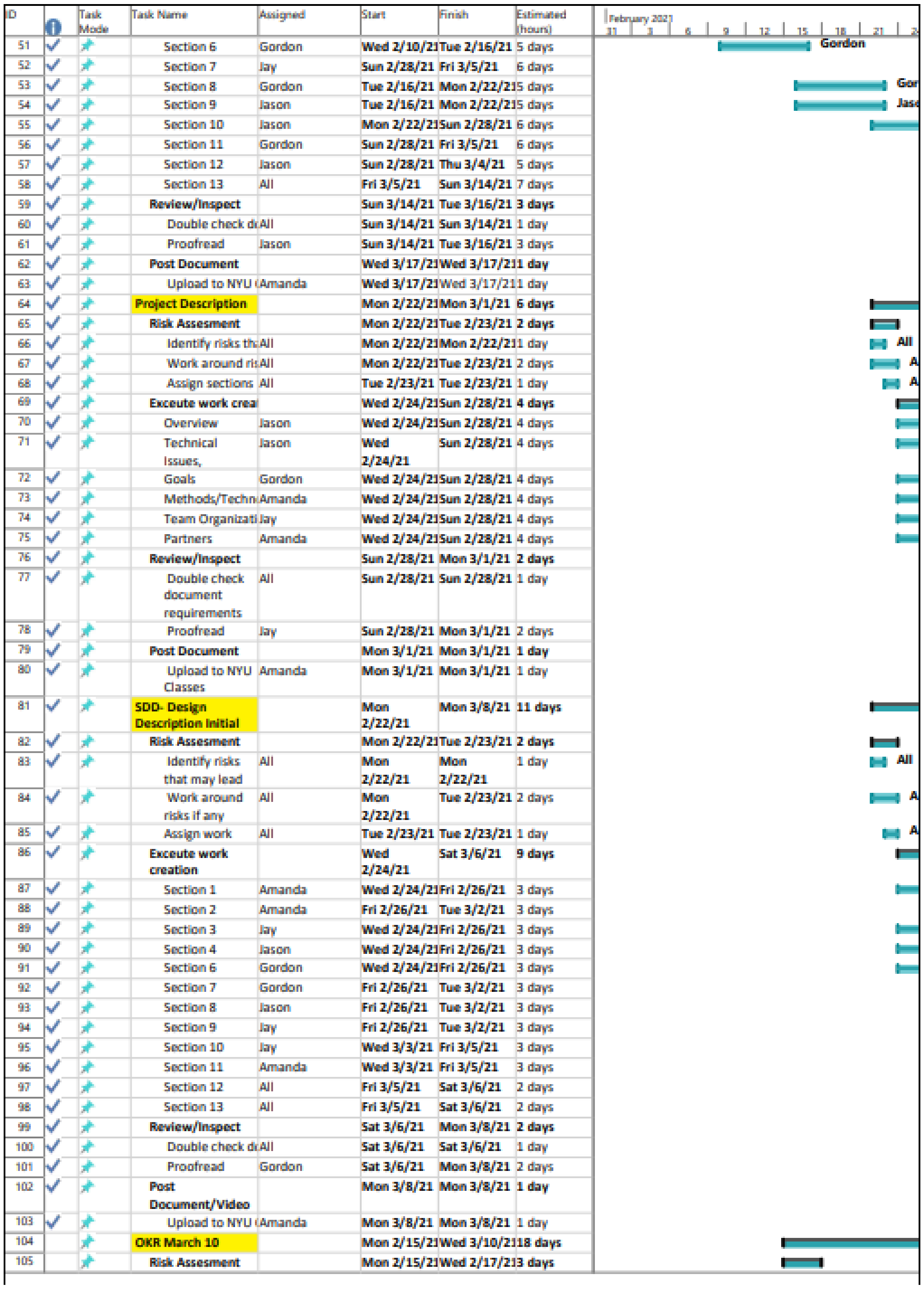
Cumulative

Who (Individual or Team)	Estimated	Actual	Difference
Amanda Lin	50	21	-29
Gordon Lei	30	5	-25
Jason Li	30	13	-17
Jay Kang	30	6	-24
Summary for entire team	130	45	-85

13.3 Gantt Chart/Microsoft Project Schedule

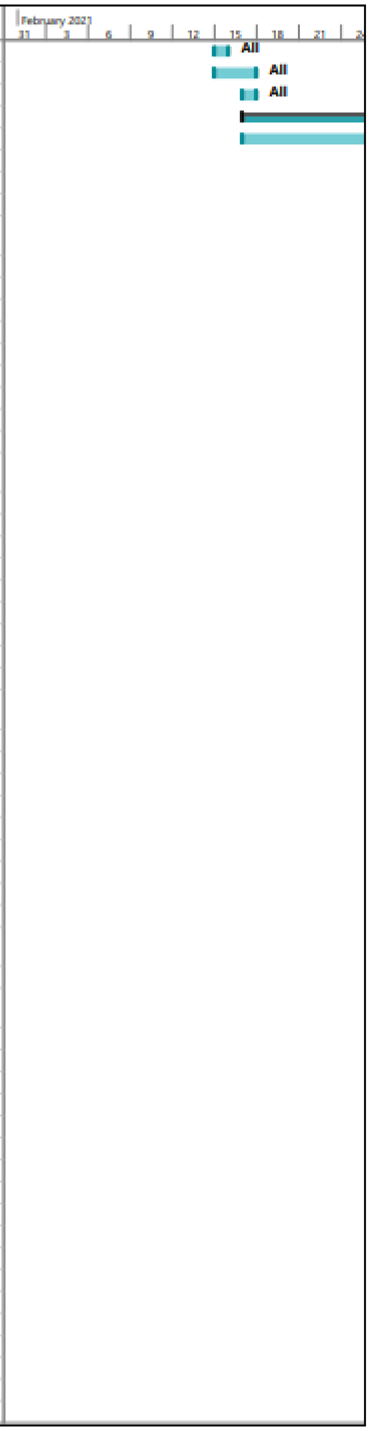


ID		Task Mode	Task Name	Assigned	Start	Finish	Estimated (hours)
1	✓	📌	**Project Team Selection Form**		**Sun 2/7/21**	**Thu 2/18/21**	**10 days**
2	✓	📌	**Risk Assesment**		**Sun 2/7/21**	**Sun 2/7/21**	**1 day**
3	✓	📌	Identify risks that may lead	All	**Sun 2/7/21**	**Sun 2/7/21**	1 day
4	✓	📌	Work around risks if any	All	**Sun 2/7/21**	**Sun 2/7/21**	1 day
5	✓	📌	Assign work	All	**Sun 2/7/21**	**Sun 2/7/21**	1 day
6	✓	📌	**Exceute work creation**		**Sun 2/7/21**	**Sun 2/7/21**	**1 day**
7	✓	📌	Project Team Selection Form	Amanda	**Sun 2/7/21**	**Sun 2/7/21**	1 day
8	✓	📌	**Review/Inspect**		**Sun 2/7/21**	**Mon 2/8/21**	**2 days**
9	✓	📌	Double check document requirements	All	**Sun 2/7/21**	**Mon 2/8/21**	2 days
10	✓	📌	**Post Document**		**Mon 2/8/21**	**Mon 2/8/21**	**1 day**
11	✓	📌	Upload to NYU Classes	Amanda	**Mon 2/8/21**	Mon 2/8/21	1 day
12	✓	📌	**Project Proposal**		**Sun 2/7/21**	**Tue 2/9/21**	**3 days**
13	✓	📌	**Risk Assesment**		**Sun 2/7/21**	**Sun 2/7/21**	**1 day**
14	✓	📌	Identify risks that may lead	All	**Sun 2/7/21**	**Sun 2/7/21**	1 day
15	✓	📌	Work around risks if any	All	**Sun 2/7/21**	**Sun 2/7/21**	1 day
16	✓	📌	Assign sections	All	**Sun 2/7/21**	**Sun 2/7/21**	1 day
17	✓	📌	**Exceute work creation**		**Mon 2/8/21**	**Mon 2/8/21**	**1 day**
18	✓	📌	Project motivati	All	**Mon 2/8/21**	**Mon 2/8/21**	1 day
19	✓	📌	**Review/Inspect**		**Tue 2/9/21**	**Tue 2/9/21**	**1 day**
20	✓	📌	Double check document requirements	All	**Tue 2/9/21**	**Tue 2/9/21**	1 day
21	✓	📌	Proofread	All	**Tue 2/9/21**	**Tue 2/9/21**	1 day
22	✓	📌	**Post Document**		**Tue 2/9/21**	**Tue 2/9/21**	**1 day**
23	✓	📌	Upload to NYU Classes	Amanda	**Tue 2/9/21**	Tue 2/9/21	1 day
24		📌	**SRS- Requirements and Analysis**		**Sun 2/7/21**	**Fri 2/19/21**	**11 days**
25	✓	📌	**Risk Assesment**		**Sun 2/7/21**	**Mon 2/8/21**	**2 days**
26	✓	📌	Identify risks th	All	**Sun 2/7/21**	**Sun 2/7/21**	1 day
27	✓	📌	Work around ris	All	**Sun 2/7/21**	**Mon 2/8/21**	2 days
28	✓	📌	Assign sections	All	**Mon 2/8/21**	**Mon 2/8/21**	1 day
29	✓	📌	**Exceute work crea**		**Mon 2/8/21**	**Wed 2/17/2**	**8 days**
30	✓	📌	Section 3	Amanda	**Mon 2/8/21**	**Mon 2/15/2**	6 days
31	✓	📌	Section 6	Gordon,Jay	**Mon 2/8/21**	**Mon 2/15/2**	6 days
32	✓	📌	Section 7	Jay	**Mon 2/8/21**	**Mon 2/15/2**	6 days
33	✓	📌	Section 9	Amanda,Gordon,J	**Mon 2/15/2**	**Wed 2/17/2**	3 days
34	✓	📌	Section 13	All	**Mon 2/15/2**	**Wed 2/17/2**	3 days
35		📌	**Review/Inspect**		**Wed 2/17/2**	**Fri 2/19/21**	**3 days**
36	✓	📌	Double check d	All	**Wed 2/17/2**	**Wed 2/17/2**	1 day
37	●	📌	Proofread	Gordon	**Wed 2/17/2**	**Fri 2/19/21**	3 days
38	✓	📌	**Post Document**		**Fri 2/19/21**	**Fri 2/19/21**	**1 day**
39	✓	📌	Upload to NYU (	Amanda	**Fri 2/19/21**	Fri 2/19/21	1 day
40		📌	**SPMP- Project Management Plan**		**Sun 2/7/21**	**Wed 3/17/21**	**29 days**
41	✓	📌	**Risk Assesment**		**Sun 2/7/21**	**Mon 2/8/21**	**2 days**
42	✓	📌	Identify risks th	All	**Sun 2/7/21**	Sun 2/7/21	1 day
43	✓	📌	Work around ris	All	**Mon 2/8/21**	**Mon 2/8/21**	1 day
44	✓	📌	Assign sections	All	**Mon 2/8/21**	**Mon 2/8/21**	1 day
45		📌	**Exceute work crea**		**Wed 2/10/2**	**Sun 3/14/21**	**24 days**
46	✓	📌	Section 1	Amanda	**Wed 2/10/2**	**Tue 2/16/21**	5 days
47	✓	📌	Section 2	Amanda	**Tue 2/16/21**	**Mon 2/22/2**	5 days
48	●	📌	Section 3	Amanda	**Mon 2/22/2**	**Sun 2/28/21**	6 days
49	✓	📌	Section 4	Jay	**Wed 2/10/2**	**Tue 2/16/21**	5 days
50	✓	📌	Section 5	Jay	**Tue 2/16/21**	**Mon 2/22/2**	5 days


February 2021
31
3
6
9
12
15
18
21
24
Gordon
Gor
Jaso
All
A

ID	ⓘ	Task Mode	Task Name	Assigned	Start	Finish	Estimated (hours)
51	✓	📌	Section 6	Gordon	Wed 2/10/21	Tue 2/16/21	5 days
52	✓	📌	Section 7	Jay	Sun 2/28/21	Fri 3/5/21	6 days
53	✓	📌	Section 8	Gordon	Tue 2/16/21	Mon 2/22/21	5 days
54	✓	📌	Section 9	Jason	Tue 2/16/21	Mon 2/22/21	5 days
55	✓	📌	Section 10	Jason	Mon 2/22/21	Sun 2/28/21	6 days
56	✓	📌	Section 11	Gordon	Sun 2/28/21	Fri 3/5/21	6 days
57	✓	📌	Section 12	Jason	Sun 2/28/21	Thu 3/4/21	5 days
58	✓	📌	Section 13	All	Fri 3/5/21	Sun 3/14/21	7 days
59	✓	📌	**Review/Inspect**		**Sun 3/14/21**	**Tue 3/16/21**	**3 days**
60	✓	📌	Double check d	All	Sun 3/14/21	Sun 3/14/21	1 day
61	✓	📌	Proofread	Jason	Sun 3/14/21	Tue 3/16/21	3 days
62	✓	📌	**Post Document**		**Wed 3/17/21**	**Wed 3/17/21**	**1 day**
63	✓	📌	Upload to NYU (	Amanda	Wed 3/17/21	Wed 3/17/21	1 day
64	✓	📌	**Project Description**		**Mon 2/22/21**	**Mon 3/1/21**	**6 days**
65	✓	📌	**Risk Assesment**		**Mon 2/22/21**	**Tue 2/23/21**	**2 days**
66	✓	📌	Identify risks th	All	Mon 2/22/21	Mon 2/22/21	1 day
67	✓	📌	Work around ris	All	Mon 2/22/21	Tue 2/23/21	2 days
68	✓	📌	Assign sections	All	Tue 2/23/21	Tue 2/23/21	1 day
69	✓	📌	**Exceute work crea**		**Wed 2/24/21**	**Sun 2/28/21**	**4 days**
70	✓	📌	Overview	Jason	Wed 2/24/21	Sun 2/28/21	4 days
71	✓	📌	Technical Issues,	Jason	Wed 2/24/21	Sun 2/28/21	4 days
72	✓	📌	Goals	Gordon	Wed 2/24/21	Sun 2/28/21	4 days
73	✓	📌	Methods/Techn	Amanda	Wed 2/24/21	Sun 2/28/21	4 days
74	✓	📌	Team Organizati	Jay	Wed 2/24/21	Sun 2/28/21	4 days
75	✓	📌	Partners	Amanda	Wed 2/24/21	Sun 2/28/21	4 days
76	✓	📌	**Review/Inspect**		**Sun 2/28/21**	**Mon 3/1/21**	**2 days**
77	✓	📌	Double check document requirements	All	Sun 2/28/21	Sun 2/28/21	1 day
78	✓	📌	Proofread	Jay	Sun 2/28/21	Mon 3/1/21	2 days
79	✓	📌	**Post Document**		**Mon 3/1/21**	**Mon 3/1/21**	**1 day**
80	✓	📌	Upload to NYU Classes	Amanda	Mon 3/1/21	Mon 3/1/21	1 day
81	✓	📌	**SDD- Design Description Initial**		**Mon 2/22/21**	**Mon 3/8/21**	**11 days**
82	✓	📌	**Risk Assesment**		**Mon 2/22/21**	**Tue 2/23/21**	**2 days**
83	✓	📌	Identify risks that may lead	All	Mon 2/22/21	Mon 2/22/21	1 day
84	✓	📌	Work around risks if any	All	Mon 2/22/21	Tue 2/23/21	2 days
85	✓	📌	Assign work	All	Tue 2/23/21	Tue 2/23/21	1 day
86	✓	📌	**Exceute work creation**		**Wed 2/24/21**	**Sat 3/6/21**	**9 days**
87	✓	📌	Section 1	Amanda	Wed 2/24/21	Fri 2/26/21	3 days
88	✓	📌	Section 2	Amanda	Fri 2/26/21	Tue 3/2/21	3 days
89	✓	📌	Section 3	Jay	Wed 2/24/21	Fri 2/26/21	3 days
90	✓	📌	Section 4	Jason	Wed 2/24/21	Fri 2/26/21	3 days
91	✓	📌	Section 6	Gordon	Wed 2/24/21	Fri 2/26/21	3 days
92	✓	📌	Section 7	Gordon	Fri 2/26/21	Tue 3/2/21	3 days
93	✓	📌	Section 8	Jason	Fri 2/26/21	Tue 3/2/21	3 days
94	✓	📌	Section 9	Jay	Fri 2/26/21	Tue 3/2/21	3 days
95	✓	📌	Section 10	Jay	Wed 3/3/21	Fri 3/5/21	3 days
96	✓	📌	Section 11	Amanda	Wed 3/3/21	Fri 3/5/21	3 days
97	✓	📌	Section 12	All	Fri 3/5/21	Sat 3/6/21	2 days
98	✓	📌	Section 13	All	Fri 3/5/21	Sat 3/6/21	2 days
99	✓	📌	**Review/Inspect**		**Sat 3/6/21**	**Mon 3/8/21**	**2 days**
100	✓	📌	Double check d	All	Sat 3/6/21	Sat 3/6/21	1 day
101	✓	📌	Proofread	Gordon	Sat 3/6/21	Mon 3/8/21	2 days
102	✓	📌	**Post Document/Video**		**Mon 3/8/21**	**Mon 3/8/21**	**1 day**
103	✓	📌	Upload to NYU (	Amanda	Mon 3/8/21	Mon 3/8/21	1 day
104		📌	**OKR March 10**		**Mon 2/15/21**	**Wed 3/10/21**	**18 days**
105		📌	**Risk Assesment**		**Mon 2/15/21**	**Wed 2/17/21**	**3 days**

ID	ⓘ	Task Mode	Task Name	Assigned	Start	Finish	Estimated (hours)
106	●	📌	Identify risks th	All	Mon 2/15/21	Mon 2/15/21	1 day
107	●	📌	Work around ris	All	Mon 2/15/21	Wed 2/17/21	3 days
108	●	📌	Assign work	All	Wed 2/17/21	Wed 2/17/21	1 day
109		📌	**Execute work crea**		Wed 2/17/21	Sat 3/6/21	**14 days**
110	●	📌	OKR	All	Wed 2/17/21	Sat 3/6/21	14 days
111		📌	**Review/Inspect**		Sat 3/6/21	Mon 3/8/21	**2 days**
112	●	📌	Double check d	All	Sat 3/6/21	Sat 3/6/21	1 day
113	●	📌	Proofread	Gordon	Sat 3/6/21	Mon 3/8/21	2 days
114		📌	**Post Document/Video**		Mon 3/8/21	Mon 3/8/21	**1 day**
115	●	📌	Upload to NYU (	Amanda	Mon 3/8/21	Mon 3/8/21	1 day
116		📌	**Front end Developm**		Wed 3/10/21	Wed 5/5/21	**41 days**
117		📌	**Risk Assesment**		Wed 3/10/21	Mon 3/15/21	**4 days**
118	●	📌	Identify risks th	All	Wed 3/10/21	Fri 3/12/21	3 days
119	●	📌	Work around ris	All	Fri 3/12/21	Sun 3/14/21	2 days
120	●	📌	Assign work	All	Sun 3/14/21	Mon 3/15/21	2 days
121		📌	**Execute work crea**		Mon 3/15/21	Wed 4/28/21	**33 days**
122	●	📌	Develop code	Jay,Jason	Mon 3/15/21	Wed 4/28/21	33 days
123		📌	**Review/Inspect**		Wed 4/28/21	Wed 5/5/21	**6 days**
124	●	📌	Double check requirements	All	Wed 4/28/21	Fri 4/30/21	3 days
125	●	📌	Bug fixes	All	Fri 4/30/21	Wed 5/5/21	4 days
126		📌	**Back end developme**		Wed 3/10/21	Wed 5/5/21	**41 days**
127		📌	**Risk Assesment**		Wed 3/10/21	Mon 3/15/21	**4 days**
128	●	📌	Identify risks th	All	Wed 3/10/21	Fri 3/12/21	3 days
129	●	📌	Work around ris	All	Fri 3/12/21	Sun 3/14/21	2 days
130	●	📌	Assign work	All	Sun 3/14/21	Mon 3/15/21	2 days
131		📌	**Execute work crea**		Mon 3/15/21	Wed 4/28/21	**33 days**
132	●	📌	Code back end	Gordon,Amanda	Mon 3/15/21	Wed 4/28/21	33 days
133		📌	**Review/Inspect**		Wed 4/28/21	Wed 5/5/21	**6 days**
134	●	📌	Double check Requirements	All	Wed 4/28/21	Fri 4/30/21	3 days
135	●	📌	Bug fixes	All	Fri 4/30/21	Wed 5/5/21	4 days
136		📌	**Database developme**		Wed 3/10/21	Wed 5/5/21	**41 days**
137		📌	**Risk Assesment**		Wed 3/10/21	Mon 3/15/21	**4 days**
138	●	📌	Identify risks th	All	Wed 3/10/21	Fri 3/12/21	3 days
139	●	📌	Work around ris	All	Fri 3/12/21	Sun 3/14/21	2 days
140	●	📌	Assign work	All	Sun 3/14/21	Mon 3/15/21	2 days
141		📌	**Execute work crea**		Mon 3/15/21	Wed 4/28/21	**33 days**
142	●	📌	Create database	Jay,Gordon	Mon 3/15/21	Wed 4/28/21	33 days
143		📌	**Review/Inspect**		Wed 4/28/21	Wed 5/5/21	**6 days**
144	●	📌	Double check requirements	All	Wed 4/28/21	Fri 4/30/21	3 days
145	●	📌	Bug fixes	All	Fri 4/30/21	Wed 5/5/21	4 days
146		📌	**SDD- Design Description Final**		Thu 4/1/21	Wed 5/5/21	**25 days**
147		📌	**Risk Assesment**		Thu 4/1/21	Sat 4/3/21	**3 days**
148	●	📌	Identify risks th	All	Thu 4/1/21	Thu 4/1/21	1 day
149	●	📌	Work around ri	All	Thu 4/1/21	Sat 4/3/21	3 days
150	●	📌	Assign work	All	Sat 4/3/21	Sat 4/3/21	1 day
151		📌	**Execute work crea**		Sat 4/3/21	Sat 5/1/21	**22 days**
152	●	📌	Section 1	Amanda	Sat 4/3/21	Fri 4/9/21	6 days
153	●	📌	Section 2	Amanda	Fri 4/9/21	Tue 4/13/21	3 days
154	●	📌	Section 3	Jay	Sat 4/3/21	Fri 4/9/21	6 days
155	●	📌	Section 4	Jason	Sat 4/3/21	Fri 4/9/21	6 days
156	●	📌	Section 6	Gordon	Sat 4/3/21	Fri 4/9/21	6 days
157	●	📌	Section 7	Gordon	Fri 4/9/21	Tue 4/13/21	3 days
158	●	📌	Section 8	Jason	Fri 4/9/21	Tue 4/13/21	3 days
159	●	📌	Section 9	Jay	Fri 4/9/21	Tue 4/13/21	3 days
160	●	📌	Section 10	Jay	Tue 4/13/21	Wed 4/21/21	7 days
161	●	📌	Section 11	Amanda	Tue 4/13/21	Wed 4/21/21	7 days
162	●	📌	Section 12	All	Wed 4/21/21	Sat 5/1/21	9 days
163	●	📌	Section 13	All	Wed 4/21/21	Sat 5/1/21	9 days
164		📌	**Review/Inspect**		Sat 5/1/21	Wed 5/5/21	**4 days**
165	●	📌	Double check d	All	Sat 5/1/21	Sat 5/1/21	1 day
166	●	📌	Proofread	Gordon	Sat 5/1/21	Wed 5/5/21	4 days
167		📌	**Post Document/V**		Wed 5/5/21	Wed 5/5/21	**1 day**
168		📌	Upload to NYU	Amanda	Wed 5/5/21	Wed 5/5/21	1 day


February 2021
All
All
All