PROJECT PLAN

Mirai

Team Alpha

Derek Williamson

Tyler Carey

Anna Pikus

Anthony Nguyen

Table of Contents

Document Change Control

Revision Number	Date of Issue	Author(s)	Brief Description of Change
v1.0	02/03/2020	Derek Williamson, Tyler Carey, Anna Pikus, Anthony Nguyen	Initial publishing of the Project Plan for Mirai

1. Project Overview

1.1 Purpose, Scope, and Objectives

- The purpose of this project, Mirai, is to provide a solution to the troubles of day-to-day organization of tasks, events, and lists through an intuitive online dashboard in the form of a web application. Compatibility with smaller mobile devices (such as smartphones) is out-of-scope and will not be estimated as a portion of the work effort for the project. The primary objective for the team in developing the application is to create a web application that is accessible from both PCs and tablets.
- In order for this project to be considered feature-complete and deliverable, the following requirements must be fulfilled:
  - All front- and back-end functionality described in this document should be code-complete,
  - All code implementations should be unit, integration, and functionally tested where possible, and
  - All mitigatable and resolvable risks have been mitigated and resolved.
- This project, along with a feature-complete application, aims to achieve the following objectives:
  - Provide an intuitive “desktop”-like interface for users to plan and organize their day-to-day tasks,
  - Allow users to use the application through an existing Google account, and
  - Store user information in a secure environment.
- The full, living collection of requirements for this project can be found on the team’s Kanban board, found on Trello (https://mistad.net/4901/alpha). Requirements for both project deliverables and functionality are laid out as stories and tasks in the board’s swim lanes. Because the team follow’s an Agile workflow, specific functional requirements for the application are subject to frequent change. Creating a static document to hold these requirements would severely limit the flexibility of the developers’ creativity and workflow.

1.2 Assumptions, Constraints and Risks

- In order for Mirai to reach deliverability by the desired delivery date, the following assumptions have been made:
  - Google and Amazon resources, APIs, and SDKs are all feature-complete, secure, stable, and well-tested.
  - Additional deliverables will not be required outside of the already defined list of project deliverables detailed in section 1.3.

- Future work commitment remains consistent with current projections.
- Resource dependencies will either remain in the free tier or remain within the budget of credits provided by the University of North Texas.

- In remain prepared, the team has outlined the following constraints and risks on the project:
  - Developers are constrained by their work and class schedules and risk under-commitment when compared to what's been planned.
  - The application is dependent on the stability of the third-party web services that host, persist, and maintain our application's environment and is at risk of either a security or availability compromise.
  - The team has no dedicated quality assurance testers to offload the work of ensuring that the product meets requirements and follows testing standards.
  - Application deployment is dependent on the stability of the chosen CI/CD tool, Travis CI.

1.3 Project Deliverables

- As required to satisfy the full terms of the project, the following deliverables must be produced and completed by the end of the project (4/23/2020):
  - Project Plan
  - System Requirements Specification
  - Design Specification
- These deliverables will be submitted as PDFs for device-to-device consistency.
- All contents are for internal use only and may be viewed by any individual in the University.

1.4 Schedule and Budget Summary

- Details on the schedule and budget summary can be found in sections 3.5.1 and 3.5.4, *Work Breakdown Structure* and *Budget Allocation*, respectively.

1.5 Definitions and Acronyms

- GitHub – a Git repository hosting service with many additional features to aid in the control and collaboration of a project.
- Travis CI – a cloud hosted continuous integration service used to build and test software projects hosted at GitHub.
- Node.js – an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a browser.

- JavaScript – the object-orientated programming language that Mirai is programmed in.
- MongoDB – a cross-platform document-oriented database program.
- ElasticBeanstalk – an auto-scaling, load-balanced virtual server that executes and maintains the life of any deployed application without the need for developers to configure an operating system.
- Sass – (Syntactically Awesome Style Sheets) a style sheet language that is an extension of CSS and enables you to employ variables, nested rules, inline imports and more.
- AWS – (Amazon Web Services) a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs on a metered pay-as-you-go basis.
- GCP – (Google Cloud Platform) a suite of cloud computing services offered by Google that runs on the same infrastructure that Google uses internally for its end-user products.
- ESLint – a static code analysis tool for identifying problematic patterns found in JavaScript code.
- Agile – a method of software development where the requirements and solutions of a project evolve through the collaborative effort of self-organizing and cross-functional teams and their end-user.
- SWOT Analysis – a strategic planning technique used to help a person or organization identify strengths, weaknesses, opportunities, and threats related to a business competition or project planning.
- HTML – (Hypertext Markup Language) the standard markup language for documents designed to be displayed in a web browser.
- CSS – (Cascading Style Sheets) a style sheet language used for describing the presentation of a document written in a markup language like HTML.
- Bash – a Unix shell and command language used by the application's deployment and setup scripts.

2. Project Organization

2.1 External Interfaces

- Boundaries between internal and external interfaces of this project are defined by the means of interaction between the application and the service itself. Any interaction that requires network access to external IP addresses is considered an external interface. Mirai's architecture requires the following external interfaces (information tentative):
  - AWS ElasticBeanstalk – hosts the application in a secure cloud environment.
  - AWS S3 – retains files in a secure cloud bucket for storage.
  - MongoDB – retains all of the application's data and metadata.
  - GCP Compute Engine – hosts the MongoDB database program instance.
  - Google Sign-In – handles all Google authentication and authorization.

2.2 Roles and Responsibilities

- During the development of Mirai, the team must participate in the following major work activities:
  - Backlog Grooming – team members must aid in the creation and reviewal of requirements and stories for the project.
  - Development – team members must contribute to the codebase of the application in a manner that extends or repairs functionality.
  - Testing – team members must write and peer-review test cases for the functionality they add or modify within the application's source.
  - Stand-up – team members must update their team daily on any activities they worked on in the previous work day and any activities they plan to commit to on the current work day.
  - Documentation – team members must contribute to the authoring of any deliverables or documentation required for the completion of the project.
- As team lead and project manager, Derek Williamson is responsible for ensuring these work activities are being committed to by the team (Tyler Carey, Anna Pikus, and Anthony Nguyen).

3. Managerial Process Plans

3.1 Start-up Plan

3.1.1 Estimates

To estimate the cost requirements for the project, a function point analysis is used and functions are broken down into three categories of complexity: simple, medium, and complex. The following list of functions is not comprehensive. As the project progresses, new functions will be added to the list.

No.	Description	Complexity
1	Integrate MongoDB	Medium
2	Integrate Travis CI	Simple
3	Integrate S3	Simple
4	Design a CloudFormation template for all necessary AWS resources	Medium
5	Draft creative visuals of the application	Medium
6	Author development documentation	Medium
7	Integrate Google Sign-In	Medium
8	Create the landing page	Simple
9	Create the dashboard page	Complex
10	Create the profile page	Medium
11	Create the sign-in page	Medium
12	Create the preferences page	Simple
13	Integrate push notifications	Simple
14	Integrate email notifications	Medium
15	Create the card functionality	Complex
16	Create the folder functionality	Complex

3.1.2 Estimated Build Effort

Function/Program	Effort	Number of Units	Total Build Effort

Simple Use Cases	2 Person Days	5	10
Medium Use Cases	4 Person Days	8	32
Complex Use Cases	7 Person Days	3	21
Total		16	63

3.1.3 Phase Effort Estimate

Phase	Person Days
Design and Analysis	10
Build and Integration	63
Testing	7
Training	5
Deployment	2
Total	**87**

Due to varying levels of skills among staff and new function additions, re-estimates will be performed on a regular basis as each task is completed. New estimates will be calculated based upon how long a team member takes to complete their assigned task and how many new functions are added.

3.2 Staffing

The staff will be comprised of current University of North Texas students. The duration in which the staff will be required is four months (February - May). Each student will have a designated role based upon experience and skills. The technical skills needed to complete the project will be required in all phases of development and testing. The managerial skills needed will be required for the duration of the project.

3.2.1 People by Role

Role	Number Required
Project Leader	1

Developer	1
Developer	1
Developer	1
Total	4

3.2.2 People by Skill and Experience

Area	Total #	Minimum Months Experience Needed
Javascript	4	3
MongoDB	1	3
HTML/CSS	4	3
NodeJS	4	3
AWS	4	3
GCP	2	3

3.2.3 Resource Acquisition

All necessary resources will be available at the start of the project. Any additional resources above budget will be requested through the University of North Texas Computer Science Department. Software will be obtained through various websites. Hardware will be provided by the staff or University of North Texas.

Resource	Contact	Responsibility to Obtain
Software	Derek Williamson	All staff
Hardware	University of North Texas	All staff
Meeting locations	Collin Higher Education Center	All staff
Training	Anna Pikus	All staff

3.3 Project Staff Training

All staff will be trained in the below areas. Training will be completed through online tutorials videos and written manuals or through staff already trained.

Training Area	Duration	Waiver Criteria
GitHub (technical)	2 days	If already trained
Javascript (technical)	5 days	If already trained
MongoDB (technical)	3 days	If already trained
HTML/CSS (technical)	4 days	If already trained
AWS (technical)	5 days	If already trained
NodeJS (technical)	5 days	If already trained
Visual Studio Code (technical)	1 day	If already trained
Testing (technical)	5 days	If already trained
Project Management (managerial)	2 hours	If already trained
Configuration Management (managerial)	2 hours	If already trained
Group Review (managerial)	1 hour	If already trained

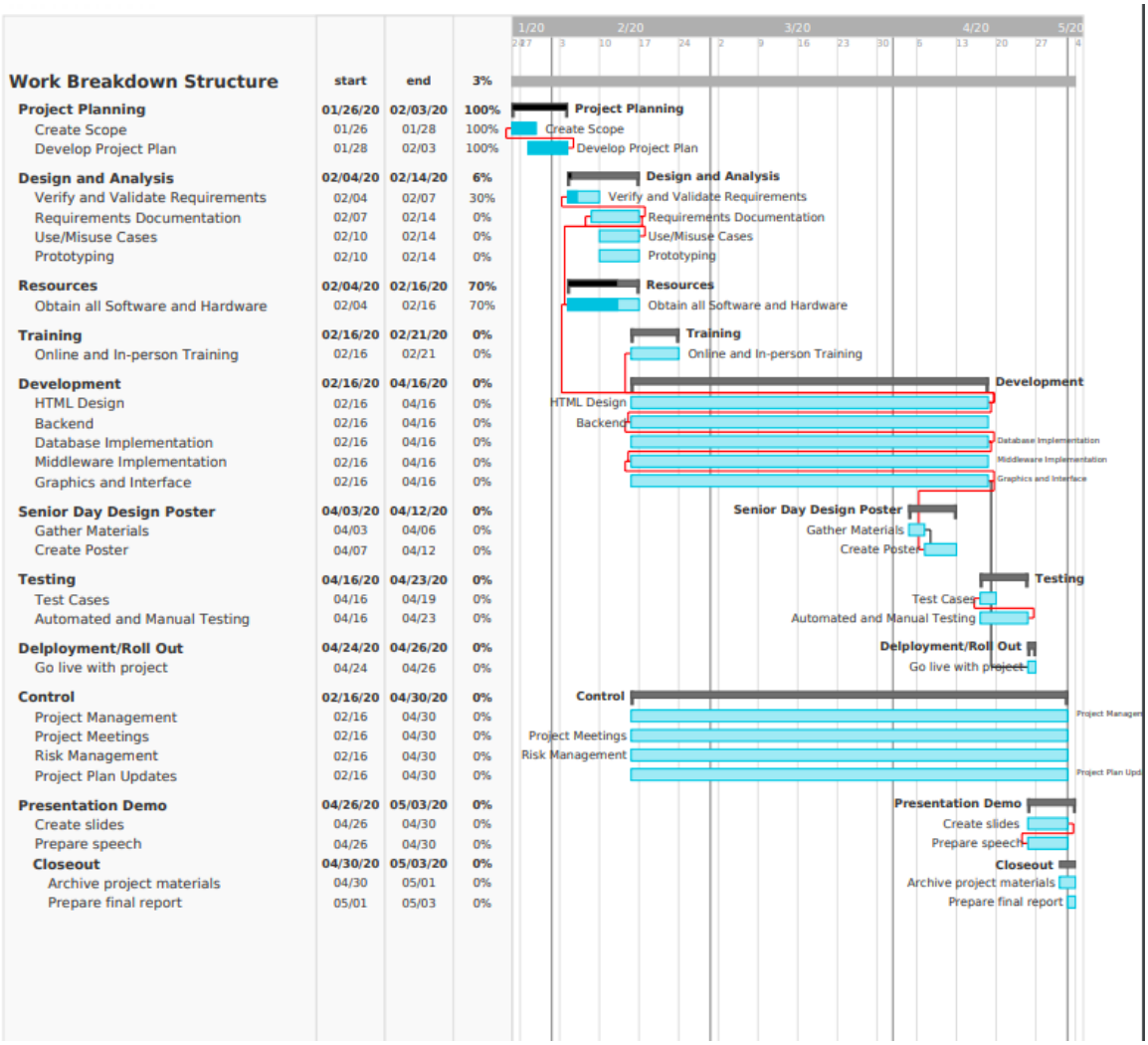
3.4 Work Plan

3.4.1 Work Breakdown Structure

No.	Task	Start Date	End Date
1.0	**Project Planning**	**1/26/2020**	**2/3/2020**
1.0.1	Create Scope Statement		
1.0.2	Develop Project Plan		
1.1	**Design and Analysis**	**2/4/2020**	**2/16/2020**
1.1.1	Verify and Validate Requirements		
1.1.2	Requirements Documentation		

1.1.3	Use/Misuse Cases		
1.1.4	Prototyping		
1.2	**Obtain all Software and Hardware**	**2/4/2020**	**2/16/2020**
1.3	**Training**	**2/16/2020**	**2/21/2020**
1.4	**Development**	**2/16/2020**	**4/16/2020**
1.4.1	HTML Design		
1.4.2	Backend		
1.4.2.1	Database implementation		
1.4.2.2	Middleware implementation		
1.4.3	Graphics and Interface		
1.5	**Senior Design Day Poster**	**4/3/2020**	**4/12/2020**
1.6	**Testing**	**4/16/2020**	**4/23/2020**
1.6.1	Test Cases		
1.6.2	Automated and Manual Testing		
1.7	**Deployment/Roll Out**	**4/24/2020**	**4/26/2020**
1.8	**Control**	**2/16/2020**	**4/30/2020**
1.8.1	Project Management		
1.8.2	Project Meetings		
1.8.3	Risk Management		
1.8.4	Project Plan Updates		
1.9	**Presentation Demo**	**4/26/2020**	**4/30/2020**
1.10	**Closeout**	**4/30/2020**	**5/3/2020**

3.4.2 Schedule Allocation



3.4.3 Resource Allocation

No.	Task	Human Resources	Hardware and Software	Skill Level
1.0	**Project Planning**	**All staff**		**Low**
1.0.1	Create Scope Statement	All staff	Google Docs	Low
1.0.2	Develop Project Plan	All staff	Google Docs	Low
1.1	**Design and Analysis**	**All staff**		
1.1.1	Verify and Validate Requirements	All staff		Low

1.1.2	Requirements Documentation	All staff		Medium
1.1.3	Use/Misuse Cases	All staff		Low
1.1.4	Prototyping	All staff		Medium
1.2	**Obtain all Software and Hardware**	**All staff**		**Medium**
1.3	**Training**	**All staff**		**Low**
1.4	**Development**	**All staff**		
1.4.1	HTML Design	All staff		High
1.4.2	Backend	All staff	NodeJS	High
1.4.2.1	Database implementation	All staff	MongoDB	High
1.4.2.2	Middleware implementation	All staff	AWS, GCP	High
1.4.3	Graphics and Interface	All staff	Sass	High
1.5	**Senior Design Day Poster**	**All staff**		**Medium**
1.6	**Testing**	**All staff**		
1.6.1	Test Cases	All staff		Low
1.6.2	Automated and Manual Testing	All staff	ESLint	Medium
1.7	**Deployment/Roll Out**	**All staff**		**Medium**
1.8	**Control**	**All staff**		
1.8.1	Project Management	All staff		Low
1.8.2	Project Meetings	All staff		Low
1.8.3	Risk Management	All staff		Low
1.8.4	Project Plan Updates	All staff		Low
1.9	**Presentation Demo**	**All staff**		**Medium**
1.10	**Closeout**	**All staff**		**Medium**

3.4.4 Budget Allocation

Human Resources Cost

Project Leader	0 USD
Team Members	0 USD
Total	**0 USD**

Software and Hardware Cost

Personal Computers/Laptops	0 USD
Raspberry Pi (optional)	0 USD
MongoDB	0 USD
AWS	0 USD
Visual Studio Code	0 USD
Total	**0 USD**

Managerial Costs

Training	0 USD
Meeting rooms	0 USD
Slack	0 USD
Trello	0 USD
Total	**0 USD**

3.5 Project Tracking Plan

3.5.1 Requirements Management

Any changes made to requirements will need to be reviewed by team members and approved by the project manager. Approvals or rejections will be made based upon time, cost, and resource availability. If a requirement is approved, WBS time and tasks, and scope statement will be updated to accompany the change.

3.5.2 Schedule Control

The project will follow the WBS schedule. Trello will be used to track and measure progress of tasks and milestones. If tasks are completed late or new tasks are added, corrective action will be taken. This may include:

- Reorganizing the schedule
- Changing/removing requirements
- Adding additional team members to the task
- Obtaining new resources

The quality of work completed at each milestone will be measured by:

- Conformance to requirements
- Completeness
- Consistency
- Reviews passed

The scope of work completed at each milestone will be measured by:

- Level of difficulty
- Skill level needed to complete
- Time it will take to complete

3.5.3 Budget Control

The cost of work completed will be measured by the time and money it took to complete each WBS task. Cost reporting will be completed every other week and will be tracked through TeamGantt and Trello.

If the actual cost becomes more than the budgeted cost, corrective action will be taken. This may include:

- Contacting the CS Department for help with costs
- Re-allocating resources
- Removing extra features
- Finding new resources

3.5.4 Quality Control

To measure and control quality of work, group reviews will be held at the end of each completed task in the WBS. In addition, during the development process, any code that is pushed will be required to be reviewed by at least one team member. If there are any issues in quality, the reviewer will communicate what needs to be fixed and corrective action will be taken. The project manager will review the final revised product.

3.5.5 Reporting

All staff will be in constant communication throughout each week. Due to a limited time frame to complete the project, it is important that requirements, schedule, and cost statuses are transparent.

The methods for communication include:

- Slack
- Trello
- Google Docs

Two types of reports should be constructed:

- Weekly status report detailing accomplishments on the week
- Biweekly status report detailing cost and schedule status

3.5.6 Project Metrics

Mirai will both self-track metrics in the application's database and integrate the Google Analytics API to collect and retain project metrics.
Information regarding the processing of metrics are defined in the following bullets:

- Mirai will record the following metrics: visits, first-time dashboard interactions, accounts registered, total successful sign-in attempts, and online user count during peak hours.
- Most of the aforementioned metrics will be recorded on occurence, except for the online user count which will be recorded daily at noon, 3 PM, and 6 PM.
- The application will validate incoming occurrences by running checks using both third-party integrations and internal session checks (e.g., user agent comparison, incoming URL, etc.) before passing metrics to the reporting service. If all checks pass, the reporting service will record the data to the appropriate metric persisters.

3.6 Risk Management Plan

In order to identify potential risks, the staff will:

- Meet in person or on slack to discuss and record risks
- Review lessons learned from past projects
- Perform a SWOT analysis
- Review major phases of project and risks associated with each
- Consider project scope, time, and money

To analyze and prioritize the risks, the staff will:

- Give each risk a percentage of likelihood
- Give each risk a number that represents the impact the risk would have 1(low)-5(high)
- Multiply the two numbers together for the risk score
- Prioritize by ranking the risk scores from highest to lowest

Risks with high scores will require a planned response and action. Risks with median scores will be individually assessed to determine if a plan is needed. Risks with low scores will be left unplanned, but will be assessed throughout the

lifetime of the project to determine if planning may be needed later. All risks will be reassessed and scores may be adjusted as each phase of the project is completed. The risks will be created and tracked through Project Online available to anyone with a shared link.

To manage the risks, the staff will either:

- Avoid the risk
- Transfer the risk
- Create a contingency plan which will consist of all risks, their preventive measures, and ways to restore the project
- Accept the risk

Potential Risk Examples

Risk	Impact(1-5)
Team member drops the class	5
Sudden growth in requirements	4
Productivity Issues	3

3.7 Project Close-out Plan

To close out the project, the staff will:

- Confirm all WBS tasks are completed
- Document any tasks that are not complete for legitimate reasons
- Confirm all requirements have been met
- Ensure all Trello tasks are at 100% completion
- Conduct final project evaluation. This will be split up and each team member will document their findings
- Return all borrowed or rented equipment, software, or services
- Perform final risk assessment
- Conduct lessons learn session
- Perform analysis of project objectives achieved
- Congratulate and thank all stakeholders and team members
- Archive the project on GitHub

To prepare for the final report, the staff will document any lessons learned throughout the project and create a brief analysis as each objective is achieved. Any communication after project completion will be through school email or Slack.

4. Technical Process Plans

4.1 Process Model

- Information will be shared among team members in accordance with configuration management in section 5.1.1. Team members will use Slack to communicate their progress and Trello to display the status of the project. GitHub will be used to consolidate code between all members.
- Status reports documenting the progress of Mirai will be submitted every week, along with a detailed document at the end of each sprint (Sprint 0, Sprint 1, and Sprint 2) which states what has been accomplished during the previous sprint and what will be the focus of the next sprint.
- Reviews of Mirai will be conducted after every major pull request implementing a new feature in the development branch. Members who implement code for a pull request will not be allowed to review their own code for the pull request.
- Approval will mainly consist of reviews conducted during major pull requests but will also occur when shipping code from the development branch to the production branch of Mirai.
- The major milestones of Mirai will be as follows:
    - The first major milestone will be to complete the software requirements of the project. This will consist of completing the Project Plan to outline the objectives of the project and the System Requirements Specification to outline the features and behaviors of the Mirai application.
    - The second major milestone will be to set up the framework of Mirai. This will consist of setting up all connections between the MongoDB database, the AWS Elastic Beanstalk instance, and the Google Cloud Computing instance. This will also include setting up user authentication to the website.
    - The third major milestone will be to develop the main functions of the application. This will be the preliminary implementation of the application and will focus on ensuring server-side code behaves as intended.
    - The fourth major milestone will be to develop the user interface of the application. This will focus on development of the client-side application and ensuring client-side code behaves as intended.
    - The fifth major milestone will be ensuring client-side and server-side code runs together smoothly through extensive testing. This will consist of unit and integration testing between all modules of the application. The sixth major milestone will be closing the project and delivering the project. This will consist of confirming all work conforms to the features and behaviors outlined in the System Requirements

Specification, ensuring all remaining balances are paid, getting formal acceptance of the project, and handing off the completed product.

- Mirai is a web application that helps users organize their day-to-day tasks through a personal, customizable dashboard. This application helps remind users of upcoming and current tasks as well as track their progress and productivity.
- Mirai will be completed with this schedule:

Major Milestone	Scheduled Completion
1	Sprint 0
2	Sprint 0
3	Sprint 1
4	Sprint 2
5	Sprint 2
6	4/26/2020

- The final delivered product of Mirai will be a web application. This web application will be accessible from any device capable of accessing the internet.
- The process model will be reviewed after every sprint to ensure compliance with the requirements.

4.2 Methods, Tools, and Techniques

- Mirai will be developed following the "Agile" development method. This process focuses on iterative releases which will allow the development team to identify and fix defects early.
- The application will be written in a combination of the following programming/markup languages and frameworks: JavaScript (Node.js), HTML, and CSS (Sass).
- The applications "Trello" and "Slack" will be used to manage progress on new features. Team members will provide updates through these applications to ensure on-time deployment of new features.
- GitHub will be used for version control and to manage the development pipeline using pull requests. To accomplish this, pull requests will be used to check new features as they are moved onto the development branch. After a code review, these iterations will be added to the development branch. Once the iterations are checked to ensure proper integration and to ensure they meet expectations and requirements, they will be moved onto the production branch for deployment.
- Travis CI will be used to test and validate code on GitHub.

4.3 Infrastructure

- For development purposes, Mirai will be developed on primarily Windows (and sometimes MacOS) using Visual Studio Code, Bash,

- For the production environment, Mirai's application will be hosted on an AWS ElasticBeanstalk instance with an AWS S3 bucket for storage. Mirai's MongoDB instance will be hosted on a Google Cloud Platform Linux instance.

- Team members will be responsible for upkeep of the local development environment and ensuring all code is compatible with the production environment.

- GitHub will be used to maintain the application codebase to ensure code is maintained across multiple environments.

- Travis CI will be used to test and validate code on GitHub.

4.4 Product Acceptance

- Mirai will be in a user-acceptable position once the application has satisfied all of the functional requirements and tests and a feature-complete version of the application has been accepted and deployed.
- The following functional tests will be required:
    - The application must have a secure login protocol.
    - The application must have a user dashboard page which displays all user events and cards.
    - The application must be able to migrate data from other applications.
    - The application must have a standardized header and footer.
    - The application must be able to support event and card creation/deletion by the user.
    - The application must be able to send push or email notifications for user events and cards.
    - Users must have the ability to manage their events and cards.
- These functional tests will be updated once the Software Requirements Specification has been finalized.
- The Mirai Application must be completed and delivered no later than April 23, 2020.

5. Supporting Process Plans

5.1 Configuration Management

This section will discuss the configuration management plan for the IM/IT project known as “Mirai.” The methods we will use will change based on the different situation we are presented.

- For identifying the configuration templates will be used for reference as well as communication between the team will take place so that everyone on the team knows what the standard for any given task may be.
- To keep control of the configuration discussion will take place constantly throughout the project and the approval system will make sure that people are keep the standard that was originally discussed.
- To keep track of the status and account for everyone’s hard work we will be using an application called “Trello”. This will allow us to have a visual board with each task someone is working on with different columns to see where the task is in terms of completion.
- For evaluation of the project before code can be implemented to the master branch the developing branch needs at least one form of approval by someone who has not worked on that merging branch.
- Once the branch has been approved it may be merged and then deleted as to not have multiple dead branches cluttering the workspace.

The different procedures for the main processes we will go through are:

- For the initial baselining of work products discussion will take place to discuss the new idea and form a plan on how we will start.
- To analyse change requests and log the communication for said requests we will be using either the application “Slack” or through the change request system implemented in our GitHub repository.
- For the change in control board procedures, if a change is necessary to the original plan all members of the team must agree that a change is necessary and then continue further in the development.
- In order to track changes made or ones in progress the use of Trello to see where the task is and also ongoing conversation about the said changes will take place until the change has been completed through the messaging system in Slack.

This project is using multiple automated configuration management tools using the application Slack we have set reminders that will automatically remind everyone on the team multiple times a week to make sure everyone is on task. Such reminders consist of due dates and also reminders to communicate with the team if you are struggling on your task, if you need help or if you are doing fine.

5.2 Verification and Validation

The verification and validation for the IM/IT project known as “Mirai” are as follows:

- The scope for verifying and validating the project is to make sure that all things implemented are helpful and work towards the end goal that was originally defined or the new altered end product.
- GitHub’s approval system that has been implemented as well as the system to ask for approval in the messaging system Slack will help and organize the verification and validation of the project.
- Each team member can verify and validate any other team members work. If there is a differencing on opinion between the two team members a third or fourth member may discuss the idea and reach a solution. Each team member works independently of the other unless providing help and not waiting on another team member.
- Multiple progress reviews will take place throughout the course of the project, each team member reviews the others work ethic and each team member is in charge of making sure the other team members are staying on topic and staying working. If one team member is stuck or is struggling, others will help assist them. The use of Slack and Trello will provide visual help to see where a task is in terms of completion.
- Before a part of the product is added testing will occur to make sure and provide a secondary layer of validation to the part of the product to make sure that everything is still holding that standard.
- Travis CI has been integrated with GitHub to automatically test and validate each new build of the application.

5.3 Documentation

The plan for the generation of non-deliverable and deliverable project documentation is as follows:

- Google Docs is being used as the platform for creating either deliverable or non-deliverable documentation so that the team may work together and complete the documentation in a timely manner.
- The use of Slack’s messaging system will provide the team with an easy communication tool to organize and complete the documentation.
- The complete list of documents that need to be completed have been placed in a Google Docs folder to stay organized and to provide a template to the team for each of the different documents. Each document can be compiled by any team member and will also need to be approved and reviewed by a different team member or all other team members.

- Due dates and tasks have been setup using the reminders automatically implemented to Slack. The system will remind the team of due dates two days before, one day before, the day of, and the hour before the due date.

5.4 Quality Assurance

The quality assurance plan for the IM/IT project known as “Mirai” is as follows:

- Maintain consistent communication and validation throughout the project’s development. The reviewal and approval system will both ensure that each team member is submitting work that holds that same originally discussed standard. Each process must adhere to the originally discussed or the newly altered level of standard that the team has established.
- During the implementation process, all changes should be validated and approved through the *Reviews and Audits* (section 5.5) process detailed below.
- Each team member is responsible for both their own work and the work of their fellow team members. There is no set priority for who can review what parts of the project. However if a team member is unsure whether the changes they’ve made to any part of the project meets the criteria, the project manager is responsible for assisting and validating their part.

5.5 Reviews and Audits

The schedule and the procedures to be used throughout development of the project are as follows:

- As a team member finishes their contribution, they must communicate create a pull request through GitHub to begin the review process. Once they have the created the pull request, they must send it through the team’s Slack channel for code reviews. Someone else, other than the original author, from the team can then review it. Approvals and change requests will be made through both GitHub and Slack.
- Travis CI oversees any GitHub branch changes and builds new change sets before allowing any pull request merging by enforcing a required set of “checks.”

5.6 Problem Resolution

The procedure for processing problem reports throughout the development of the project are as follows:

- If an issue becomes apparent with a changeset of code during a pull request reviewal, a team member will request changes through GitHub and specify (1) what the change is and (2) where the change should be made. A message should be made in Slack as well to notify the pull requester of the changes.
- All problems should be addressed and resolved in a respectable, friendly manner as to not create a hostile work environment.

5.7 Subcontractor Management

No subcontractors will be used or required for the IM/IT project known as “Mirai”.

5.8 Process Improvement

For determining any improvements that could me made throughout the development process, a group consensus must be established. If everyone agrees that the improvement should be made steps to make that improvement whatever it may be will be taken. Improvement specifications will vary depending on what aspects of the project is needing improving.

6. Additional Plans

Additional plans for the IM/IT project known as “Mirai” are to turn the website application into a smart mirror. This will of course affect the budget and management process as well as the resources. Further explanation and adjustments will be made to the document if time permits this goal to be achieved.