**Municipality Permit Chatbot System (MPCS)**

**Project Plan (PP)**

**Revision 1.2**

**23 July 2020**

**Solicitation Phase**

**Prepared by**:

Rusty Baker

Sepribo Taylor-Harry

Subhash Gandhi Vallala

Joshua Piersol

Matthew Slaymaker

Nathaniel Muesing

SWEN 670

University of Maryland Global Campus

PAGE INTENTIONALLY LEFT BLANK

Project Plan Approvals

|  |  |  |
| --- | --- | --- |
| Name | Signature | Date |
| Approved by:  Dr. Mir Assadullah |  |  |
| Approved by:  "Stakeholder" |  |  |
| Approved by:  "Project Manager" |  |  |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Name | Description of Change(s) |
| 1.0 | 11 June 2020 | Rusty Baker | Initial Release |
| 1.1 | 2 July 2020 | Rusty Baker, Matthew Slaymaker | Incorporated Stakeholder feedback in addition to added in the Test Plan associated with the Project. |
| 1.2 | 23 July 2020 | Rusty Baker | Updated document revisioning schema to comply with other documents. |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Introduction 1](#_Toc44582424)

[Project Description 1](#_Toc44582425)

[Customer/User 1](#_Toc44582426)

[Contracting 2](#_Toc44582427)

[Referenced Documents 2](#_Toc44582428)

[Re-approval Criteria 3](#_Toc44582429)

[Project Assignment 3](#_Toc44582430)

[Project Scope 3](#_Toc44582431)

[Project Assumptions 4](#_Toc44582432)

[Project Constraints 4](#_Toc44582433)

[Project Organization/Staffing 5](#_Toc44582434)

[Roles and Responsibilities 6](#_Toc44582435)

[Project Tools 8](#_Toc44582436)

[Project Facilities 10](#_Toc44582437)

[Project Documents Storage 10](#_Toc44582438)

[Distribution Statement and Security Classification Markings for Work Products 10](#_Toc44582439)

[Knowledge/Skills Needed 10](#_Toc44582440)

[Work Breakdown Structure 11](#_Toc44582441)

[Project Schedule 12](#_Toc44582442)

[High-Level Schedule 12](#_Toc44582443)

[Project Integrated Master Schedule (IMS) 13](#_Toc44582444)

[Estimates for Size, Effort, and Cost 13](#_Toc44582445)

[Project Budget 16](#_Toc44582446)

[Metrics Approach 16](#_Toc44582447)

[Deliverables 17](#_Toc44582448)

[Description 18](#_Toc44582449)

[Communication 19](#_Toc44582450)

[Reviews and Meetings 19](#_Toc44582451)

[Risk Analysis 20](#_Toc44582452)

[Risk management Process 20](#_Toc44582453)

[Current program risks 21](#_Toc44582454)

[Organization of the project 24](#_Toc44582455)

[Documentation 26](#_Toc44582456)

[Major Product Tasks 26](#_Toc44582457)

[Project Software Development Process 27](#_Toc44582458)

[*Scrum Team* 28](#_Toc44582459)

[Product Owner. 28](#_Toc44582460)

[Scrum Master. 29](#_Toc44582461)

[Development Team. 29](#_Toc44582462)

[*Scrum Activities* 30](#_Toc44582463)

[Release Planning. 30](#_Toc44582464)

[Sprint. 30](#_Toc44582465)

[Sprint Planning Meeting. 31](#_Toc44582466)

[Sprint Retrospective. 32](#_Toc44582467)

[*Scrum Artifacts* 32](#_Toc44582468)

[Product Backlog. 33](#_Toc44582469)

[Product Burn-Down. 34](#_Toc44582470)

[Sprint Backlog. 35](#_Toc44582471)

[Unit Test. 35](#_Toc44582472)

[Software Integration. 36](#_Toc44582473)

[*Software Engineering Elements* 36](#_Toc44582474)

[Programming Languages. 36](#_Toc44582475)

[Coding Guidelines. 36](#_Toc44582476)

[Code Reviews. 36](#_Toc44582477)

[*Software Configuration Management* 37](#_Toc44582478)

[*Build Strategy* 37](#_Toc44582479)

[Software Test Plan 38](#_Toc44582480)

[Test Plan Identifier 38](#_Toc44582481)

[Introduction 38](#_Toc44582482)

[Background 41](#_Toc44582483)

[Objectives of the Test Plan 41](#_Toc44582484)

[Objectives of UAT 41](#_Toc44582485)

[References 42](#_Toc44582486)

[Test Items 42](#_Toc44582487)

[Features to Be Tested 42](#_Toc44582488)

[Features Not To Be Tested 43](#_Toc44582489)

[Approach 43](#_Toc44582490)

[Develop Tests 44](#_Toc44582491)

[Prepare to Test 45](#_Toc44582492)

[Run Tests 45](#_Toc44582493)

[Review Test Results 46](#_Toc44582494)

[Test Results and defects 46](#_Toc44582495)

[Item Pass/Fail Criteria 47](#_Toc44582496)

[Suspension Criteria and Resumption Requirements 48](#_Toc44582497)

[Test Deliverables 48](#_Toc44582498)

[Testing Tasks 49](#_Toc44582499)

[Environmental Needs 49](#_Toc44582500)

[Responsibilities 49](#_Toc44582501)

[Staffing and Training Needs 50](#_Toc44582502)

[Schedule 51](#_Toc44582503)

[Risks and Contingencies 52](#_Toc44582504)

[Appendix A: Acronyms and Abbreviations 54](#_Toc44582505)

[Appendix B: IMS/WBS 55](#_Toc44582506)

List of Tables

[Table 1 *Referenced Documents* 2](#_Toc44582758)

[Table 2 *MPCS Roles and Responsibilities* 6](#_Toc44582759)

[Table 3 *Project Tools* 8](#_Toc44582760)

[Table 4 *Project Possible Working Days* 13](#_Toc44582761)

[Table 5 *Project-Specific Core Metrics* 17](#_Toc44582762)

[Table 6 *Deliverables* 18](#_Toc44582763)

[Table 7 *Summary of Reviews and Meetings* 19](#_Toc44582764)

[Table 8 *Current Identified Project Risks* 22](#_Toc44582765)

[Table 9 *Testable Municipality Permit Chatbot System items corresponding version numbers* 42](#_Toc44582766)

[Table 10 *Testable MPCS Items within Chatbot* 42](#_Toc44582767)

[Table 11 *Testable Business processes for City Official Interface System* 43](#_Toc44582768)

[Table 12 *Issue Priority Ratings* 46](#_Toc44582769)

[Table 13 *Test Responsibilities* 50](#_Toc44582770)

[Table 14 *Sprint 4 Test Schedule* 51](#_Toc44582771)

[Table 15 *Sprint 5 Test Schedule* 51](#_Toc44582772)

[Table 16 *Sprint 6 Test Schedule* 52](#_Toc44582773)

[Table 17 *Acronyms and Abbreviations* 54](#_Toc44582774)

List of Figures

[*Figure 1.* Chatbot System Organizational Diagram 5](#_Toc44582865)

[*Figure 2.* Chatbot High-Level Schedule. The Image was derived from IMS Timeline View. 12](#_Toc44582866)

[*Figure 3.* Chatbot Staffing profile allocated hours based on timeline and effort 14](#_Toc44582867)

[*Figure 4.* Chatbot Total Project Labor Hours 14](#_Toc44582868)

[*Figure 5.* Cross-functional team member salaries and hourly rates data 15](#_Toc44582869)

[*Figure 6.* The total estimated cost of the project 15](#_Toc44582870)

[*Figure 7.* Risk management process overview. Reprinted from 5 Steps to Any Effective Risk Management. 20](#_Toc44582871)

[*Figure 8*. Risk Assessment Matrix. 21](#_Toc44582872)

[*Figure 9.* Current Risks Identified in the Risk Assessment Matrix 24](#_Toc44582873)

[*Figure 10.* Scrum Process 28](#_Toc44582874)

[*Figure 11:* Product Burn-Down Char 35](#_Toc44582875)

[*Figure 12.* City Official Interface Use Case Diagram 39](#_Toc44582876)

[*Figure 13*. City Resident Chatbot Use Case Diagram 40](#_Toc44582877)

Introduction

This Project Plan (PP) documents the details required to execute and control the Municipal Permit Chatbot System (MPCS) software project successfully. Moreover, this document provides details regarding facilitating communication among stakeholders and establishing the Project's commitments (e.g., deliverables). Furthermore, this document identifies information specific to the project's document planning, assumptions, decisions, scope, and schedule baselines. Throughout this document, "project" refers to the MPCS.

Project Description

The purpose of the MPCS is to design, develop, and deliver a reliable and functional software application that assists end-users in navigating the local city website (e.g., Pasadena) by answering questions and providing information when inquired. Questions provided by the end-user are administered via a popup website chatbox. The MPCS convincingly acknowledges and interfaces with the end-user comparable to a human customer representative by determining whether to provide a direct answer or reference search results from within the website. Furthermore, the MPCS is linked to other applications within the website to provide accurate information to the end-user. The MPCS will also contain options that will be customizable by its stakeholders (e.g., City Officials) through a downloadable client. Each stage of the project will require approval from key stakeholders to transition to the subsequent phase to ensure the project has adequate coverage in meeting the stakeholder's performance expectations.

## Customer/User

The customer for the project is the University of Maryland Global Campus (UMGC). The end-user of this system are web users who access ( e.g., Web Traffic) the local city website and interact with the system. In addition to city officials who modify the MPCS through the downloadable modification client

## Contracting

The project contractual agreement is established as the course syllabus document within the SWEN 670 UMGC website. This document outlines the stakeholder's high-level requirements in addition to establishes the project milestones and deliverables. This project's primary workforce is graduate students enrolled within the SWEN 670 course within the spring 2020 semester. All procurement items affiliated with the project will acquire the program manager's approval.

## Referenced Documents

The documents, specifications, and standards listed in the table below are referenced in this plan.

Table 1

*Referenced Documents*

| Identifier | Title | Location |
| --- | --- | --- |
| CSPP-001 | The Scrum Guide™, The Definitive Guide to Scrum: The Rules of the Game | <http://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100> |
| CSPP-002 | Software Development Coding Standards | TBD |
| ISO- 31000:2018(E) | Risk Management | <https://www.iso.org/obp/ui#iso:std:iso:31000:ed-2:v1:en> |

## Re-approval Criteria

The following list depicts the criteria for when this Project Plan (PP) is obligated to change and increase its current revision status. Furthermore, this document must be re-approved by the program team's internal members, key stakeholders, ultimately to be evaluated by Quality Assurance (QA) for concurrence as was arranged for the original plan:

a) Addition or deletion of one or more products committed to being delivered

b) A change in the project's defined process

c) A change in customer requirements that affect the project's progression or plans for accomplishing development, testing, or delivery of the final product.

# Project Assignment

The subsections below provide insight into the project's resources in terms of project organization/staffing, tools, facilities, deliverables, and scope.

## Project Scope

Project execution will occur once the project has its initial meeting between team members and stakeholders on the date specified in sections 2.9 and 2.10. Once executing, the team will formally begin their initial iteration of requirements analysis while practicing the Agile Scrum methodology, as stated in Section 4. The project will be committed to satisfying the following high-level capabilities, as notated from the introductory meeting with the stakeholder:

1. Develop a Chatbot System that assists city residents with municipal website navigation
2. Develop a Chatbot System that guides city residents with current city regulations (e.g., Restrictions & Zoning) and application procedures (e.g., Permits)
3. Incorporate open-source computer software with the development of the Chatbot software application.
4. Incorporate an open-source repository infrastructure with the utilization of Github
5. Provide City residents with the ability to facilitate citizens applying for permits
6. Incorporate city codes, restrictions, and zones based on city official input.

### Project Assumptions

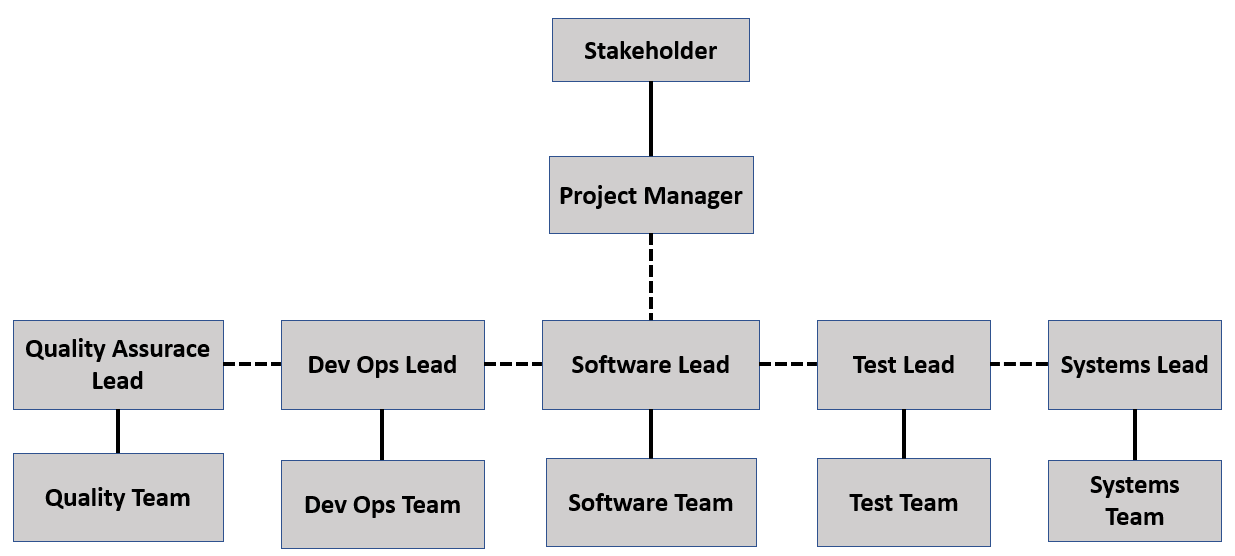
* The project will have all required procurement items provided by that of the stakeholder in order to complete the project, both human and material.
* Project team members will have the supplies needed to perform their tasks adequately, from specific equipment and software down to electricity throughout working hours.
* Personnel costs will not alter during the project lifecycle.
* The overall expense of day-to-day operations will not increase.
* Access to the UMGC will be in working condition through the project life cycle.
* Stakeholders will provide direction and responses to the project team promptly.
* The scope of the project will not change throughout the life cycle.
* The timeline associated with milestones and deliverables, as notated within the course syllabus, is permanent.

### Project Constraints

* Upon milestone completion, formal consent is mandatory from the stakeholder before moving ahead in the Project.
* Presentation material will be supplied with each milestone by that of the project team.
* Software development will not be able to begin until the stakeholder has approved all requirements.

## Project Organization/Staffing

The project is structured to support a concise chain-of-command and providing apparent resolutions in the decision process. The project's top-level organizational structure is represented in Figure 1, where solid lines depict the reporting direction, and dotted lines symbolize support direction. The responsibilities associated with each functional role are defined in complete detail in the subsequent section 2.2.1.



*Figure 1.* Chatbot System Organizational Diagram

### Roles and Responsibilities

The tables below list describe the roles and responsibilities of the project. Furthermore, the below table provides insight into the assigned individual for each role in addition to their responsibility. Throughout the project life cycle, the assignee is subject to change within the below table.

Table 2

*MPCS Roles and Responsibilities*

| Role | Assignee | Responsibility |
| --- | --- | --- |
| Stakeholder | Professor Assadullah | Stakeholders contribute to the Project's success by providing information and oversight in the Project's requirements and design. Moreover, stakeholders participate in program milestone reviews where overall Project status is reported requiring stakeholder approval to move to the next phase of the Project. Furthermore, stakeholders can review project documentation comparable to management plans, technical data, and program status reports, including metrics and risk tracking data. |
| Project Manager | Rusty Baker | Project Manager's primary responsibility is to track project activities in day to day operations in addition to planning project resources. Furthermore, the Project Manager is responsible for the management of the WBS, monitoring the schedule status of the project. The project manager acts as an interface between that of the Project Team and the Stakeholder, providing program status. |
| Quality Assurance (QA) Lead | [Subhash Gandhi](javascript://) Vallala | Quality Assurance (QA) Lead is responsible for leading the quality team in monitoring, inspecting, and proposing actions in the correction of documentation or process to improve a deliverable or internal component of the project. Furthermore, QA is responsible for conducting internal audits, ensuring the team follows the proper process and adhering to the program standards (e.g., Verification). |
| Dev Ops Lead | Sepribo [Taylor-Harry](javascript://) | The DevOps Lead is responsible for interfacing with the DevOps team by providing direction and guidance. Additionally, the DevOps Lead is responsible for developing methods that facilitate team collaboration and provide the team with the proper process of becoming an efficient process. Furthermore, DevOps is responsible for developing ways in which processes can be automated or streamlined, ultimately reducing waste within a program. |
| Software Lead | Nathaniel [Muesing](javascript://) | The Software Lead's primary responsibility is to manage the software development team in addition to provide leadership in overall software design and practices. Furthermore, the software lead is responsible for aligning with the functional team members the overall software development status. A primary task of the Software Lead is to design and develop a Software Project Plan. |
| Test Lead | Matthew Slaymaker | The Test Lead is responsible for the management of the test team in addition to ensuring that the end product is validated based upon the specified requirements. Furthermore, the Test Lead will provide an overall status to where test documents lie in addition to how tests are being conducted via a Test plan. |
| Systems Lead | Joshua Piersol | The Systems Lead is responsible for the management of the System engineers within the program and working closely with engineering professionals to maintain hardware and software efficiency. Furthermore, the systems lead coordinates with all cross-functional members to coordinate the best output when the product is under design, development, or testing. Another critical aspect of the Systems Lead is to guide software developers in with the overall design, ultimately to ensure that the system strictly performs as specified by that of the requirements. |
| Scrum Master | Nathaniel [Muesing](javascript://) | The Scrum Master facilitates Scrum meetings and acts as a Scrum team leader to help the team execute the Scrum process. The Scrum Master works with the entire Scrum team to evolve the Definition of Done, which serves as a comprehensive checklist of necessary activities that ensure the delivery of full features. The Scrum Master also coordinates with the Development Team in order to find and implement the technical practices needed to reach "done" at the end of each Sprint. Other responsibilities of the Scrum Master include removing impediments to the team's progress and fostering self-organization amongst the team. |
| Product Owner | Rusty Baker | The Product Owner is responsible for deciding what tasks will be accomplished through the selection and refinement of Product Backlog items (further discussed in Section 4.3.1.3.1, Product Backlog). Additionally, the Product Owner manages the workflow for the team. The Product Owner maintains the Product Backlog and ensures team awareness of the backlog and tasking priorities. Furthermore, the Product Owner makes an ordered, prioritized list of Product Backlog Items (PBIs), identifies necessary items and logs deferrable tasks. |

## Project Tools

The tables below list and describe the tools and software environments used for this project. All tools listed are current and available for use.

Table 3

*Project Tools*

| Tool Name | Tool Function |
| --- | --- |
| **Discord** | **Function**: Collaborative Project workspace environment that supports Project data storage, collaboration, role assignment, role permissions, and metrics  **Use**: Discord is used to share and communicate amongst cross-functional team members rapidly.  **Artifacts**:   1. Documentation 2. Team Collaboration and Document Sharing 3. Lessons Learned 4. Work Breakdown Structure (WBS) 5. Schedules 6. Action Items 7. Stored Data Items 8. Peer Reviews (PRs) 9. Risks   **Used** **by**: Project Team |
| **UMGC SWEN 670 LEO Course Classroom** | **Function**: Workspace environment that supports the submission of stakeholder deliverables.  **Use**: Project platform used for submitting deliverables. Moreover, this platform provides direct communication with Stakeholders.  **Artifacts**:   1. Deliverables 2. Stakeholder Communication 3. Project Scope   **Used** **by**: Project Team |
| **Microsoft Office** | **Function**: Toolset that includes Microsoft Word, Excel, PowerPoint, Project, Visio, Access, and Outlook  **Use**: The Office toolset is used for creating and processing documentation, spreadsheets, presentations, drawings, databases, and email/file transfers.  **Artifacts**:   1. Formal Documents and Reports 2. Risk Management Workbook 3. Presentations 4. Project WBS and Schedules 5. Organization Charts 6. Audits 7. Code Review Reports   **Used** **by**: Project Team |
| **ArgoUML, app.diagrams.net** | **Function:** Design and diagraming tools to create ERDs and UI designs  **Use:** These tools will allow the visualization of entity and class relationships. Prospective Chatbot window formatting and styling can be prototyped before committing to code.  **Artifacts:**   1. ERDs 2. UML designs |
| **IntelliJ, Visual Studio Code** | **Function**: Software integrated development environment  **Use**: This integrated development environment is used to edit and compile source code modules into executable object code modules for use on the Windows Operating System (OS). This tool will ensure that source code is written in legal constructs. It will also be used to assist the developer with debugging software and locating defects.  **Artifacts**: Software Item (SI) Modules and Test Software  **Used** **by**: Software Engineering Team, Test Team |
| **Github** | **Function**:Software Version Control System (SVCS)  **Use**: All source code will be stored in a GIT Repository. The tool will be used for version control of the software during development.  **Artifacts**:Source Code  **Used by**: Software Development Team |

## Project Facilities

There are no concrete project facilities associated with this project. The team is currently working in a virtual environment; however, overall project collaboration is done through the software application known as Discord in addition to Github.

## Project Documents Storage

All Project configuration items will be stored within Github, ultimately to be supplied to the Stakeholder via UMGC SWEN 670 LEO Classroom, as noted in Table 2 within Section 2.3. Furthermore, the Github platform will store all project data and programmatic items, including the WBS, status reporting documentation, plans, and reports. Drafts will be worked on collaboratively among the MPCS team through the Discord environment, each functional area (e.g., Software, QA, and DevOps) will have its link (e.g., Text Channel) promoting team collaboration specific to that function allowing team members to share documentation. Baseline documentation throughout the project will start at "001" and will rev up plus one for the next revision (e.g., 002).

## Distribution Statement and Security Classification Markings for Work Products

All deliverable products for this project will be intended as "Open Source" products.

## Knowledge/Skills Needed

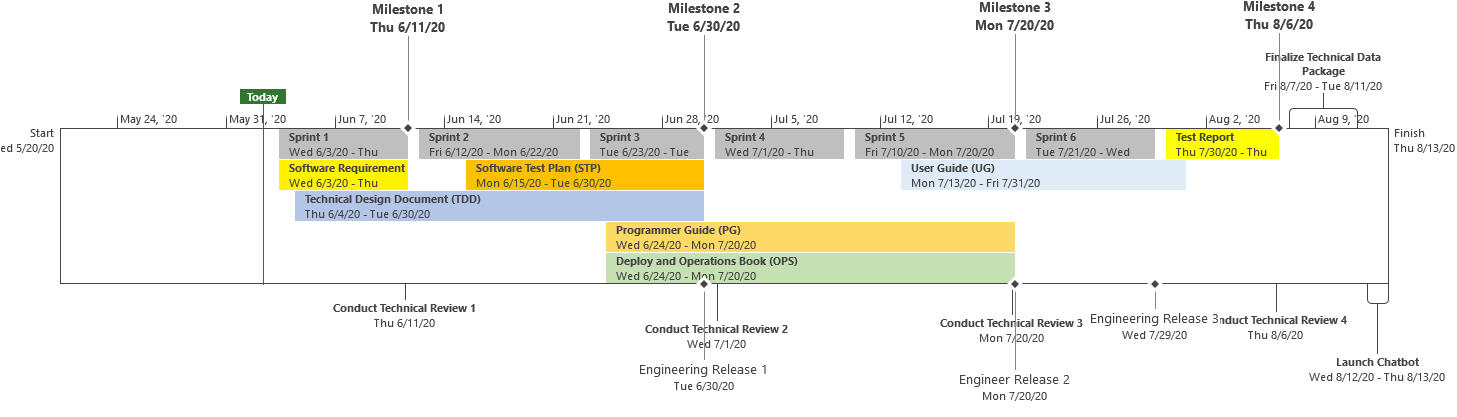
Project team members are expected to have project-specific knowledge and skills in their areas of expertise, including the relevant tools summarized in Section 2.3 of this document. Furthermore, all functional leads are required to have a basic understanding of the Agile Software Development Life Cycle (SDLC) methodology. Functional leads are required to promote collaboration amongst all team members, ensuring that the project maintains a healthy environment for all Team members. There will be no external training required for this project.

## Work Breakdown Structure

The initial Project Work Breakdown Structure (WBS) will be provided thirty days before the project kick-off date. The WBS will be maintained within the Integrated Master Schedule (IMS). The WBS will be baselined with the IMS, and both items will be monitored for modifications weekly. A copy of the IMS can be found in the documents channel within Discord. Furthermore, an overview of the WBS is outlined in 'Appendix B'.

## Project Schedule

### High-Level Schedule



*Figure 2.* Chatbot High-Level Schedule. The Image was derived from IMS Timeline View.

Figure 2 represents a high-level schedule of the Project. The start of the Project is based on the stakeholder meeting held on 20 May 2020. From this start date, the team fully established methods of communication and conducted an initial kick-off meeting. The current outlook of the schedule consists of starting the initial sprint one iteration, completing with a deliverable of milestone one. The following milestones are subject to be delivered after completion of sprints three and five, where the final milestone will be provided after completion of a Formal Qualification Test (FQT) event and generation of the test report. Figure 2 also outlines technical reviews (e.g., stakeholder meetings) where the project's overall status is presented. During this meeting, that of the stakeholder will assess the approval or disapproval of milestone completion. Furthermore, software development and documentation will be conducted throughout the six sprint iterations. It should be noted that the current portrayed high-level schedule is subject to change; however, the overall representation will be supplied to stakeholders with each milestone and provided status.

### Project Integrated Master Schedule (IMS)

The initial Project Integrated Master Schedule (IMS) was developed in association with the project plan. Furthermore, the initial version can be viewed within 'Appendix A'. Once the IMS is baselined, the project manager will ensure that the project status and tasks are updated weekly. Moreover, the IMS will outline the high functional level of the work for each deliverable (down to level four) due to having a short timeline for project performance. Workdays associated with the IMS are Monday to Friday, representing a standard IMS that would ideally be used in a standard project.

## Estimates for Size, Effort, and Cost

Size and effort have been characterized in the following section by incorporating the high-level schedule, and to how many days planning, development, documentation, and testing will take to meet program completeness of fulfilling the customer requirements as notated in the 'Project Scope' section. Furthermore, overall costs and effort are characterized by information derived from the IMS in addition to dates of expected completion supplied by that of the stakeholder. Figure 3 outlines the start and end dates as mandated by the contract and stakeholder in addition to the total amount of workdays possible and weeks.

Table 4

*Project Possible Working Days*

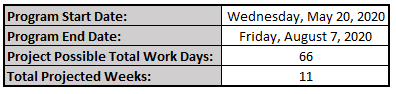
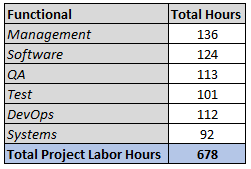


Table 4 characterizes the overall expected amount of effort needed in the form of hours needed for each cross-functional area. The amount of effort has been characterized based on project scope and the staffing profile needed to administer the work as outlined in the high-level schedule (Figure 2) with the possible amount of workdays, as shown in Figure 3. Figure 4 displays the total amount of effort in the form of labor hours needed for the project. The work hours for each cross-functional role was determined to be a total of eight hours per week maximum. Based on this characteristic, the allotted work value presents a realistic and achievable amount of hours for each cross-functional member.

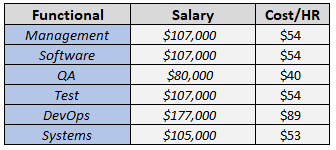


*Figure 3.* Chatbot Staffing profile allocated hours based on timeline and effort



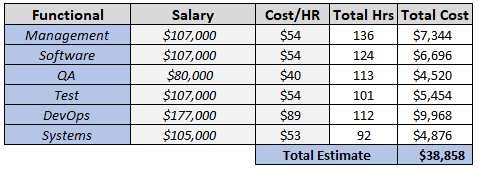
*Figure 4.* Chatbot Total Project Labor Hours

Figure 5 portrays the salary values in addition to the hourly rate of each cross functional team member. The dollar value for each cross functional role was done through market research using the website "https://www.indeed.com/salaries".



*Figure 5.* Cross-functional team member salaries and hourly rates data adapted from "<https://www.indeed.com/salaries>."

Figure 6 represents the total estimated cost of the project. The formulation of the total estimate was done by multiplying the total work hours by that of the hourly costs (derived from Figures 4 and 5). A summation of all cross functionals would generate the total project's total estimate. The overall total estimate is labor only, for there are no materials or travel associated with this project.



*Figure 6.* The total estimated cost of the project

Project Budget

Once initial and incremental funds have been obtained from the stakeholder, overall project budget information will be refreshed during the project's execution to indicate the current status of current funding. Furthermore, the overall program budget will be tracked by that of the project manager throughout weekly status meetings. An actual budget and estimated budget value will be displayed in weekly status reporting to monitor and track the overall programs Cost Variance (CV). In the generation of this initial project plan, no funds have been received by that of the stakeholder; however, it can be presumed that the total estimate offer will be approved and delivered. The initial project will be developed, utilizing free, open-source software. Software that requires funds when work is scaled up will be noted throughout the project to stakeholders. The scale will be determined on end-user volume. The current approved budget and related documentation are stored in Discord within the Documents on the channel section.

Metrics Approach

The following section provides insight into the metrics that will be used in the status reporting of the overall project. To ensure that funds are being properly spent and the project is on schedule, a weekly status report will be supplied every Sunday. This weekly report will use Earned Value Metric (EVM) status reporting characteristics in addition to an overall synopsis of the current project's health. Table 5 outlines the metrics used in EVM, where the projects projected budgeted cost and planned schedule are compared to that of the current schedule work done and actual cost. Furthermore, in calculating the projects health status, the program manager will aim and aspire to fulfill the following items: Proactively maintain project Schedule, Cost/Effort/Size, and Requirements Volatility to assure that Project commitments are being met

Table 5

*Project-Specific Core Metrics*

| Core Metrics | Project Usage |
| --- | --- |
| Weekly – Schedule Variance (SV) | SV = Budgeted Cost of Work Performed (BCWP) – Budgeted Cost of Work Scheduled (BCWS) |
| Weekly – Cost Variance (CV) | CV = Budgeted Cost of Work Performed (BCWP) – Actual Cost of Work Performed (ACWP) |
| Weekly – Schedule Performance Index (SPI) | SPI = BCWP/BCWS |
| Weekly – Cost Performance Index (CPI) | CPI = BCWP/ACWP |
| Weekly – Percent Spent | Percent Spent = (ACWP/ BCWS) \*100 |
| Weekly – Percent Complete | Percent Complete = (BCWP/BAC)\*100 |
| Weekly – Percent Scheduled | Percent Scheduled = (BCWS/BAC)\*100 |
| Weekly – Estimate at Completion (EAC) | EAC = ACWP + [(BAC-BCWP)/(CPI)] |

Deliverables

Project deliverables were identified in the approved and outlined within the contract, as mentioned in the 'Contracting' section. Furthermore, all deliverable media will be packaged and supplied to the stakeholder via the UMGC SWEN 670 LEO Course Classroom, as mention in the 'Project Tools' section. Table 6 portrays the outlined Project deliverables that are required by that of the stakeholder. The expected delivery dates for these deliverables are outlined in the IMS and can be viewed within the 'Project Schedule' section.

Table 6

*Deliverables*

| Contract Deliverable # | Name | Description |
| --- | --- | --- |
| CDN-1 | Project Plan | This document provides the overall structure of how the program will be managed. |
| CDN-2 | Software Requirement Specification | The SRS document outlines what the software will do in addition to outline what the expected software performance is. Ideally, this document outlines the functionality of the product needed in order to fulfill all stakeholder's needs. |
| CDN-3 | Technical Design Document | A TDD describes a resolution to a given problem that is correctly specified in the form of design for a software program or feature. This document is primarily used to communicate the work's technical details to members of the team. |
| CDN-4 | Software Test Plan | This document is a plan that outlines the overall approach in which software will be tested. Furthermore, this document outlines the formal testing approaches or testing activities within the validation phase in testing software. |
| CDN-5 | Programmer Guide | This document is designed with the characteristic of guiding new developers with how the software should be developed (e.g., Coding Standards, Branching Schema, Repository setup). |
| CDN-6 | Deployment and Operations Book | This document outlines the software compilation process and the procedures and operations that the system administrator or operator carries out. |
| CDN-7 | User Guide | This document outlines the procedures and processes of how the user will use the software application. This document is similar to that of an operators manual, providing the user with input into the methods |
| CDN-8 | Test Report | This document outlines the overall test results (e.g., output) from the formal test event, as outlined in the Software Test Plan (STP). Furthermore, this document is developed when the project exit criteria specified in the Evaluation and Reporting Stage Output Criteria are fulfilled. |

Communication

The Project utilizes the following communication methods during the Project lifecycle:

1. Instant messaging characteristics supplied through the use of the Discord.
2. Email exchanges via personal email addresses
3. Teleconferences are conducted via voice over internet protocol (VOIP) calls through the Discord software application.
4. Teleconferences will also be conducted using the tool Microsoft Teams.
5. Telephone discussions through personal telephones.

Reviews and Meetings

Project team informal meetings, meetings with other teams (e.g., DevOps), and formal stakeholder reviews are summarized in Table 7. Furthermore, Table 7 outlines the overall entrance and exit criteria for each meeting that is to be expected in order to facilitate communication and decrease out of the scoped conversation.

Table 7

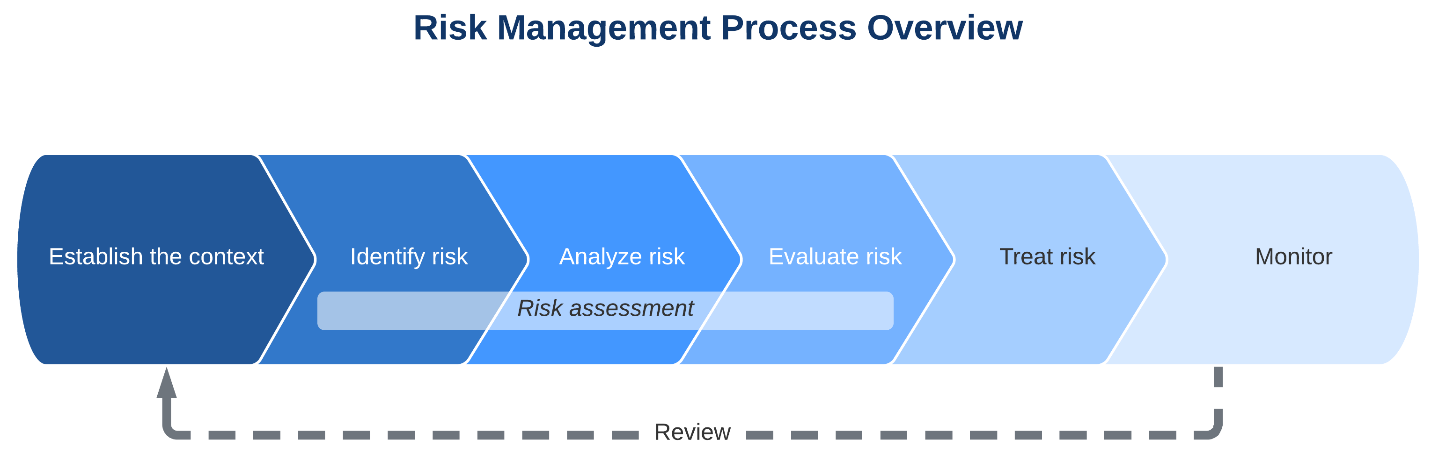
*Summary of Reviews and Meetings*

| Meeting/Review Type | Entrance Criteria | Exit Criteria | Conducted | Frequency |
| --- | --- | --- | --- | --- |
| Internal Team Meeting | Schedule status, issues, risks, team status | Action Items, Corrective Actions | Email, web-portal (e.g., Discord), Teleconference | Weekly |
| Cross-Functional Meetings with DevOps Team | Presentation Material, Action Items | Action Items, Corrective Actions | Email, Microsoft Teams | Occasional |
| Stakeholder  Formal Review | Technical Work Products, Presentation Material | Action Items, Corrective Actions, Guidance for documentation | Teleconference (e.g., Microsoft Teams) | Refer to Project schedule for the planned date |

Risk Analysis

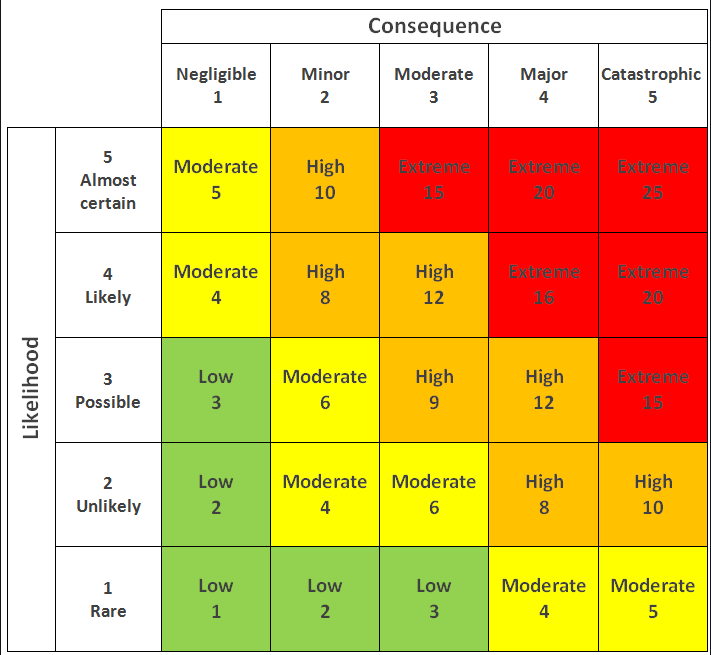
Risk management Process

The Project will follow the risk management approach, as depicted in the following Figure 7. Upon establishing the content of the Project, the team will identify any potential risks that may impact the overall program's success or baseline schedule. Once risks are identified, they will be tracked within internal status meetings, which will be stored within the Discord platform. Furthermore, overall project risks will be tracked in meetings with the stakeholder, specifically during milestone reviews.



*Figure 7.* Risk management process overview. Reprinted from 5 Steps to Any Effective Risk Management Process by Lucidchart blog. (n.d.). Retrieved from https://www.lucidchart.com/blog/risk-management-process. Copyright n.d. by Lucidchart.

Throughout the project life cycle, the project team will identify risks in which a risk assessment matrix will be used to define risk level by examining the category of probability or likelihood against the category of consequence severity, as shown in Figure 8. Project risks will be evaluated and reviewed during monthly status meetings, where the project team will distribute risks and planned mitigations for key stakeholders to review.



*Figure 8*. Risk Assessment Matrix. Reprinted from Consequence vs. Likelihood by G. Kubra Kaya (n.d.) Retrieved from https://www.researchgate.net/figure/A-standard-risk-matrix\_fig7\_323570642. Copyright n.d. by G. Kubra Kaya

Current program risks

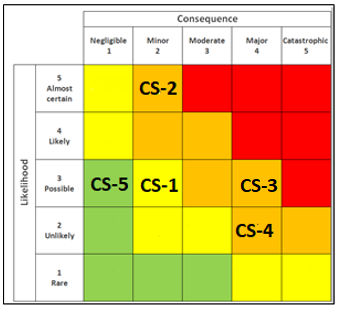
Table 10 summarizes the current identified project risks, detailing an overall risk description with a mitigation task that will be monitored throughout the life cycle of the project. Risks within the table have been identified with a risk number that will be assessed and identified within a risk assessment matrix, as shown in Figure 9.

Table 8

*Current Identified Project Risks*

| Risk Number | Description | Mitigation |
| --- | --- | --- |
| CS -1 | **Title**: Insufficient resources  **Description**:  Throughout the project, it is possible that the staffing profile might not be met, or the development environment and tools are not obtained and maintained.  **Impact**:  Overall, software development will impact on a day-to-day basis. | * Explore a different alternative to securing resources, including the hiring of new employees. * Hire quality engineers * Engage in change management while collaborating with crucial Stakeholders regarding rescheduling or reprioritizing work. |
| CS - 2 | **Title**: Low Velocity of the team  **Description**:  There is a possibility that the overall productivity of the software developers will be low for several months.  **Impact**:  The development will not be completed within the scheduled time and will have to push schedule to the right. | * Involve scrum master and team management to identify issues and act as a facilitator to resolve team impedances. * Implement a status reporting process that holds accountability on software developers and escalates items to senior management if needed. |
| CS - 3 | **Title**: Team communication  **Description**:  For instance, team communication will diminish, resulting in "Silo" development or a lack of transparency in terms of performance and tasks.  **Impact**:  Lack of communication can affect overall team performance and productivity, resulting in a slip in the schedule. | * Proactively manage communication * Establish meetings in which the team can communicate and align on tasks * Ensure all project status information is correct and up to date. * Create action items and follow up on action items when needed. |
| CS - 4 | **Title**: Poorly defined requirements  **Description**:  If the overall system performance is not fully defined, then there lies the possibility that the overall requirements are not fully defined.  **Impact**:  Reduced requirements can impact the project's development phase and test phase, ultimately impacting the schedule. | * Develop clear, complete, detailed, cohesive, attainable, and testable requirements that are agreed to by all players. * Conduct requirement review boards with the customer where the concurrence is provided on software requirement specifications (SRS). * Conduct Requirement working groups while developing requirements. |
| CS - 5 | **Title**: Poor software quality  **Description**:  Improper verification of overall process due to a lack of quality control members.  **Impact**:  Poor software quality will lead to schedule impacts due to software bugs and improper tracing. | * Create a quality control plan that will outline the project's quality efforts. * Ensure that QA members are validating requirements and design specifications. * Ensure that QA members and Team members are having walk-through inspections when needed. * Incorporate QA members on all software code reviews. |

The below image Figure 9 depicts a risk assessment matrix that contains the above list project risks outlined in Table 10. Placement of each risk was done by assigning the risks a likelihood value based on the probability of occurrence throughout the project. Moreover, a corresponding consequence value will be identified and assigned to risks within the matrix, indicating the severity and impact of the risk on the schedule.



*Figure 9.* Current Risks Identified in the Risk Assessment Matrix, adapted from Consequence vs. Likelihood by G. Kubra Kaya (n.d.) Retrieved from https://www.researchgate.net/figure/A-standard-risk-matrix\_fig7\_323570642. Copyright n.d. by G. Kubra Kaya

Organization of the project

The project will be developed using the Agile software development values and principles, as defined in the 'Software Development Process' section of this document. Agile software development encompasses several software development approaches under which requirements and resolutions unfold through the collaborative effort of self-organizing, cross-functional teams, and stakeholder involvement. Agile promotes adaptive planning, evolutionary development, early delivery, and [continual improvement](https://en.wikipedia.org/wiki/Continual_improvement_process), and it stimulates a rapid and adaptable response to improving project efficiency. The specific Agile methodology that will be employed within the project is known as Scrum, consisting of a multifaceted set of development principles as defined within this document in the 'Software Development Process' section. Furthermore, this methodology incorporates the concept of team collaboration and team transparency in communication. The organization of the project will be focused on promoting team transparency, where each cross-functional has value-added in addition to being empowered in the decision-making process of the program. Furthermore, collaboration amongst team members, specifically cross-functional leads, will center around the development of a list outlining the team's effort of work for each member in what is known as the "Product Backlog."

Scrum incorporates a concept known as a "Sprint," which is a single iteration of distributed software increments. Sprint timelines are typically one month; however, for this project, the team will exercise weekly Sprints consisting of seven days of execution. At the end of each iteration, an examination is made of the Product Backlog, where priorities are shifted based upon stakeholder feedback, and software capability functionality is further defined. This methodology empowers administration with the ability to identify problems earlier at the development stage.

All development tools are listed in the 'Project Tools' section within this document.

Product features will be recognized, measured, designated, and prioritized. Software engineering release increments will be defined, and dates will be portrayed within the high-level schedule and the IMS. In order to track the Product Backlog, the team will incorporate the use of JIRA, providing the team with an environment in which effort dedicated to documentation or development can be managed, refined, or traced. Furthermore, software development will be incorporated into a series of engineering release increments where the product will be designed, implemented, integrated, and tested.

Software development will be divided into a series of engineering releases. Furthermore, each increment will contain code specific to different system components, ultimately providing the desired functionality. Each new increment will integrate code from previous increments in addition to the new code. Each release will be approximately three months in duration. The Test Team will conduct internal tests on each release to ensure that the software development effort addresses the designated requirements. Additionally, at the end of the development cycle, the software will a Formal Qualification Test (FQT) for acceptance testing.

Documentation

The following documents will be developed during the software design and development phase of the project. Furthermore, these documents will be like "living documents" that will be progressively elaborated throughout the development cycle and baselined following the Project schedule:

* Software Requirements Specification (SRS)
* Technical Design Documents(TDD)
* Software Test Plan (STP)

Major Product Tasks

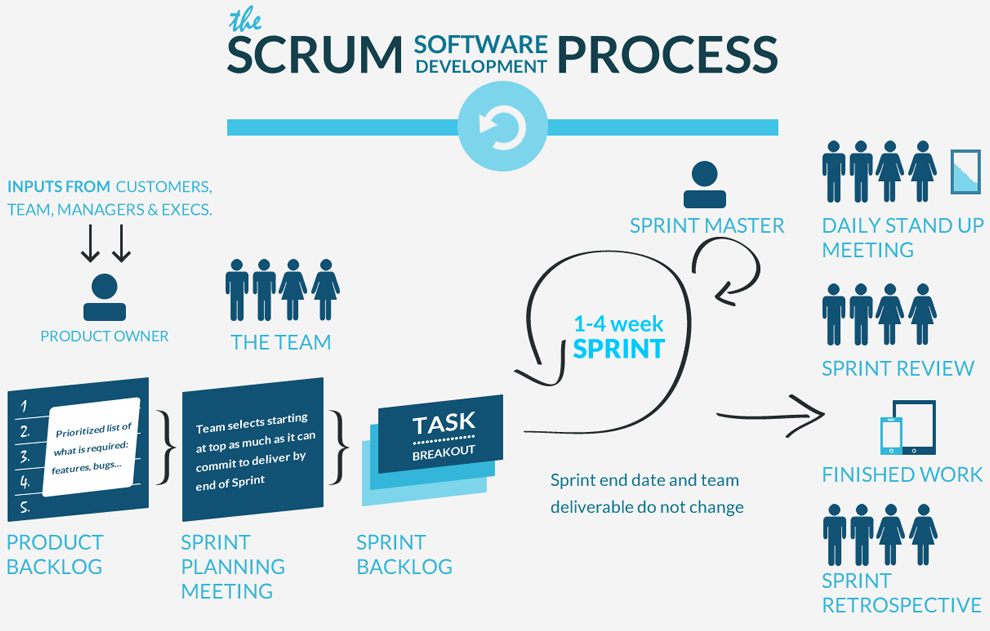
The following major product tasks will be accomplished throughout the project:

* Identification of software requirements specific to that of the MPCS.
* The team will investigate the reusability of the previous semester developed software.
* Design and develop a product that ensures all requirements are met.
* Develop a testing framework specific to that of the MPCS that can be tested via automation.
* Create reliable and accurate documentation necessary for the project.
* Integrate the software and test the product in a deployable environment via the DevOps team's infrastructure.

Project Software Development Process

Selecting the appropriate development methodology (Waterfall, Agile, etc.) for a project is a confluence of several factors: project complexity, requirement stability, project duration, project size, team member locations, and external stakeholder engagement. The Project will use an Agile software development approach to encourage proper task coordination between functional areas and to produce working products that meet Project objectives for incremental releases. Agile software development is employed to practice sustainable code development, conduct regular code testing, and produce incremental releases in order to induce product stability. The Agile approach is responsive to change and reduces risks by providing the capability to demonstrate working software at regular intervals to the customer/stakeholders.

The specific methodology of Agile that this project will incorporate is the Scrum methodology. Scrum is centered around teamwork, specifically undergoing development as a "Scrum Team. Furthermore, Scrum's overall process and methodology incorporate three primary roles, six main events, and five artifacts, which are further defined in the following subsections. Figure 10 portrays the Scrum software development process, which outlines the core concepts of software development for this project.



*Figure 10.* Scrum Process, Adapted from Image "GSA Tech Guides" by n.d. Retrieved from https://tech.gsa.gov/guides/popular\_approaches/

The following subsections will outline the core components and key members of the Scrum software development process, as outlined in Figure 10. Furthermore, the Scrum Team, Product Owner, Scrum Master, and Product Owner roles derived from Figure 10 will be discussed.

*Scrum Team*

All cross-functional members, aside from the stakeholder listed in Table 2 within the 'Roles and Responsibilities' section, will make up this project's "Scrum Team." Also, due to having limited resources in terms of personnel for this project, several team members will wear two hats. For instance, the software lead (Nathan Muesing) will be the primary developer, software lead, and scrum master.

Product Owner.

The Product Owner is responsible for deciding what tasks will be accomplished through the selection and refinement of Product Backlog items (further discussed in the 'Product Backlog; section). For this project, the Product Owner will be the Project Manager (Rusty Baker), for the manager will have the capability of managing the flow of the team while communicating requirements to the stakeholder. The Product Owner maintains the Product Backlog and ensures team awareness of the backlog and tasking priorities. Furthermore, the Product Owner makes an ordered, prioritized list of Product Backlog Items (PBIs), which outlines the key steps of effort for the project.

Scrum Master.

The Scrum Master facilitates Scrum meetings and acts as a Scrum team leader to help the team execute the Scrum process. For this project, the Scrum Master will be the Software Lead (Nathan Muesing). The primary role of the Scrum Master is to remove impedances from the Scrum Team so that the team can be as productive as possible.

Development Team.

The Development Team consists of software developers and testers performing the product iteration of development, testing, and delivery/release. Software developers for this project will consist of two individuals who is Subhash Gandhi Vallala (QA lead) and then Nathan Muesing (Software Lead). These two individuals possess a vast amount of logic when it comes to development. The testers for this project would consist of Mathew Slaymaker (Test Lead) and Sepribo Taylor-Harry (DevOps Lead). Due to having an Agile infrastructure, the roles of software developer and tester can change; however, the Cross-Functional position assigned to the individual will not change. All Scrum Team members must collaborate and work together in terms of software development and testing in order to produce an MPCS.

*Scrum Activities*

A Scrum team performs several vital activities during the life cycle of the Project in order to uphold transparency within the team and with the stakeholders. As such, the Scrum team provides status at the six time-boxed events: Release Planning Sprint, Sprint Planning Meeting, Daily Scrum, Sprint Review Meeting, and Sprint Retrospective. These events are described in the subsections below.

Release Planning.

Release planning occurs on the first day of every Sprint and remains focused on the long-term, strategic goals for the Project. The release plan starts with the prioritized and estimated Product Backlog, and it establishes the date for the release and the number and length of the Sprints. Release planning, which also involves estimating new features, concludes with a general design of each Sprint's features for the release.

The Release Planning Meeting establishes the goals for the release (such as, "What are we going to build?" and "Why are we building it?"), the risks of the release, and the overall features and functionality that will be contained in the release. Initially, the PBIs will be assigned to each release based on the Product Backlog order.

Sprint.

The team transforms its plan into working software through Sprints, the core of Scrum. In addition to daily development work, Scrum Sprints consist of the Sprint Planning Meeting, the daily Scrum, the Sprint Review, and the Sprint Retrospective Meeting. The outcome of a Sprint is to produce potentially shippable functionality. Each Sprint will be four weeks in length and will build upon the functionality from previous Sprints. Any PBIs not completed during their allocated Sprint will be incorporated into a subsequent Sprint.

Backlog grooming (or grooming the Product Backlog) is an activity that occurs during each Sprint. The PBIs are reviewed and revised as the Scrum team gains domain knowledge. The Product Owner monitors the Sprint Burn-Down to ensure that PBIs are in order of priority and include sufficient detail to use in the upcoming Sprint. At the end of each Sprint, the Product Owner statuses the Sprint Burn-Down to Project management. Additional burn-down information can be found in Section 4.3.1.3.2, Product Burn-Down.

Sprint Planning Meeting.

The purpose of the Sprint Planning Meeting is to plan the work to be performed during each Sprint. The Development Team selects the PBIs to be completed during the upcoming Sprint based on the Release Planning and PBI priority. The Sprint Planning Meeting will be devoted to defining the two aspects of each Sprint: the "What" and the "How."

The "What" aspect determines the activities of the upcoming Sprint. The Development Team selects the work to be completed after reviewing the prioritized list of unimplemented PBIs, existing issues, and bugs found during testing. The team takes into account any critical dependencies, the latest increment of software, defects discovered during testing, the capacity of the team, and any planned interruptions (such as milestone reviews, team member vacations, national holidays, etc.).

The "How" aspect determines how the selected activities will be accomplished. The Development Team plans how to implement the selected PBIs and build the functionality into a "done"increment. The team develops the design after the Sprint goals are entirely understood. All user stories selected for implementation will have the identified work divided into detailed tasks, including hourly estimates. Issues selected for implementation will be examined to determine if the issue description is sufficient for implementation. Tasks will be developed for issues that require additional definition or when that issue may require two or more tasks to support the development effort. All development tasks will follow the traditional steps in a software development life cycle: designing and coding software, unit testing, producing documentation, conducting peer reviews and creating/updating installers. These tasks will include all of the work necessary to call a PBI "done" and suitable for delivery.

Sprint Retrospective.

The Sprint Retrospective is conducted after the Sprint Review and before the next Sprint Planning Meeting. This is the final meeting of a Sprint; The Sprint Retrospective is used to assess the actual Scrum process rather than the product under development.

The Sprint Retrospective provides an opportunity for the Scrum team to achieve the following actions:

* Discuss the successes and obstacles of the last Sprint.
* Provide feedback about improving the Scrum process for future Sprints.
* Record lessons-learned on the Project's SharePoint site.

*Scrum Artifacts*

Scrum contains five principle artifacts: Product Backlog, Product Burn-Down, Sprint Backlog, Sprint Burn-Down, and Potentially Shippable Increments (PSIs). Product and Sprint Backlogs are used in planning and execution; Product and Sprint Burn-Downs assist in tracking progress, and PSIs comprise a grouping of tasks for each Sprint. Together, this simple set of artifacts enables Scrum teams to accomplish a great deal of work with minimal overhead.

Product Backlog.

The Product Backlog is a prioritized list of features, bugs, constraints, technical work, and knowledge acquisition required to develop and deliver an operable software system successfully. PBIs are derived from customer requirements and captured as user stories. Software requirements are derived from user stories and issues and are used as the basis for FQT.

A user story is a short, simple description of the requirement/feature as told from the perspective of the person desiring the capability. These user stories form the basis of the Product Backlog, an essential artifact of Scrum. The user stories will be documented in JIRA and used during development.

The PBIs are divided into releases of three Sprints per release. The first Sprint will be planned in detail, and the additional Sprints will be planned at the start of each release. The Product Owner orders the PBIs based on considerations of risk, business value, dependencies, and date needed. The features added to the Backlog are written in the user story format with common fields displayed in Table 4.4.1.3.1-1. The Product Backlog is the "What" that will be built, and it is sorted in the relative order in which it should be built.

Issues are captured from testing or user reports of fielded operational discrepancies. Issues have an extensive list of required data; issues follow the detailed workflow of a baselined product within JIRA. JIRA enforces the collection of certain required data items before moving the issue into the next workflow state.

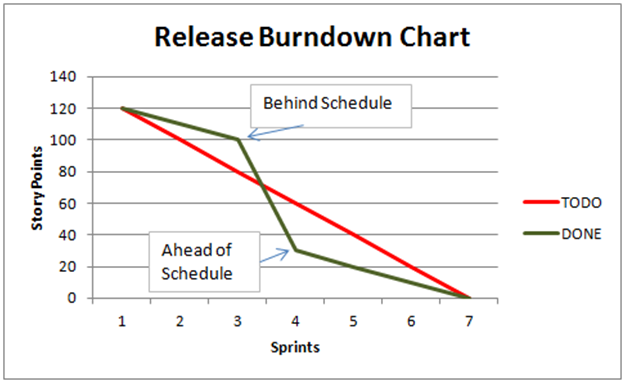
The standard JIRA data fields are listed in Table 11 below.

| Table 9  *JIRA Fields* | |
| --- | --- |
| Custom and Required Fields | |
| Estimated Hours | Proposed Solution |
| Estimated SLOC | Acceptance Criteria |
| Actual Hours | Story Points |
| Actual SLOC |  |
| Custom and Not Required Fields | |
| SRS ID | Fixed Build Version |
| Resolution Notes | Fixed Release Version |
| Code Review Completed | Documentation Updated |
| Unit Test Written | Safety Significance Marked |
| Acceptance Testing Passed |  |
| Not Custom and Required Fields | |
| Workaround |  |
| Not Custom and Not Required Fields | |
| Resolution | Block Notes |

The Product Owner is ultimately responsible for ordering the user stories and issues in the Backlog for the Development Team. The Product Backlog contains rough estimates of development effort; these values are stated in story points using a rounded Fibonacci sequence. These estimates help gauge the timeline and may influence the order of the PBIs. Issues typically address a narrowly focused development issue. Each estimate will be measured in story points, which will be developed using complexity and duration estimates.

Product Burn-Down.

The sum of the remaining effort on the Product Backlog is shown in a Product Burn-Down chart. The team uses this chart to gauge how much effort is remaining before and after each Sprint. The Product Burn-Down chart is submitted to the Product Owner after each Sprint. Figure 11 shows a burn-down chart example.



*Figure 11:* Product Burn-Down Chart, Adapted from Image "Burndown Chart" by K. Amit, Retrieved from https://worldofagile.com/blog/burn-down-chart/

Sprint Backlog.

The Sprint Backlog contains tasks created by the team to turn a set of PBIs for one Sprint into a PSI. Potentially Shippable Increment for more information. The Sprint Backlog consists of items selected from the Product Backlog, as well as any tasks the team must perform to turn those items into working functionality.

Unit Test.

Unit tests will be developed in conjunction with software development. Unit tests will be automated and run at the time of development on-demand or by using the server for each engineering release. The unit tests will be developed using a unit testing framework, ultimately to run in a Jenkins server environment established by that of the DevOps team. The server will execute the Unit Tests that have been developed, ultimately to provide a report on what failures were produced.

The unit test process utilizes our continuous integration tool, Jenkins. The automation scripts along with the server aid the software development team by supporting:

* Enables Developers to find and solve defects in a code base rapidly and to automate testing of their builds
* Compiles Code
* Builds Unit Tests
* Executes Unit Tests
* Captures Unit Test Results

Software Integration.

The Jenkins continuous integration server will be set up to perform nightly builds from the head revision of Subversion automatically. The server will run all the automated unit tests and the code coverage tool, when appropriate, in conjunction with performing the builds.

*Software Engineering Elements*

Programming Languages.

The software will be implemented using SQL, JAVA, and C#.

Coding Guidelines.

All developers will be required to follow the coding standards outlined in the Chatbot Developer Guidance Document.

Code Reviews.

Code reviews will be conducted internally and then followed up with the DevOps team. The QA cross-functional lead will ensure that the software developed meets the outlined coding guidelines. Upon completion of the code review, the team will "Commit" the software to a Git Repository, where the DevOps team will further review the software code.

*Software Configuration Management*

During the development phase, all source code will be controlled via a versioning number. Developers will check in their updated source code regularly. Check-ins will typically be done at the PBI level. When checking in code, a developer will be required to enter change-set comments and associate the check-in to a PBI. The change-set comments should provide enough detail so that anyone viewing the change history can easily discern the changes.

*Build Strategy*

An automated build and test tool will be used to detect issues with the software. The build and test will be conducted by the DevOps team, which will potentially run nightly builds on a Jenkins Server. If a problem is detected during the nightly build/test, it should be the highest priority of the Software Development Team to correct the problem. However, problems detected during the nightly build/test should be infrequent because the developers are building and testing before committing the code.

Software Test Plan

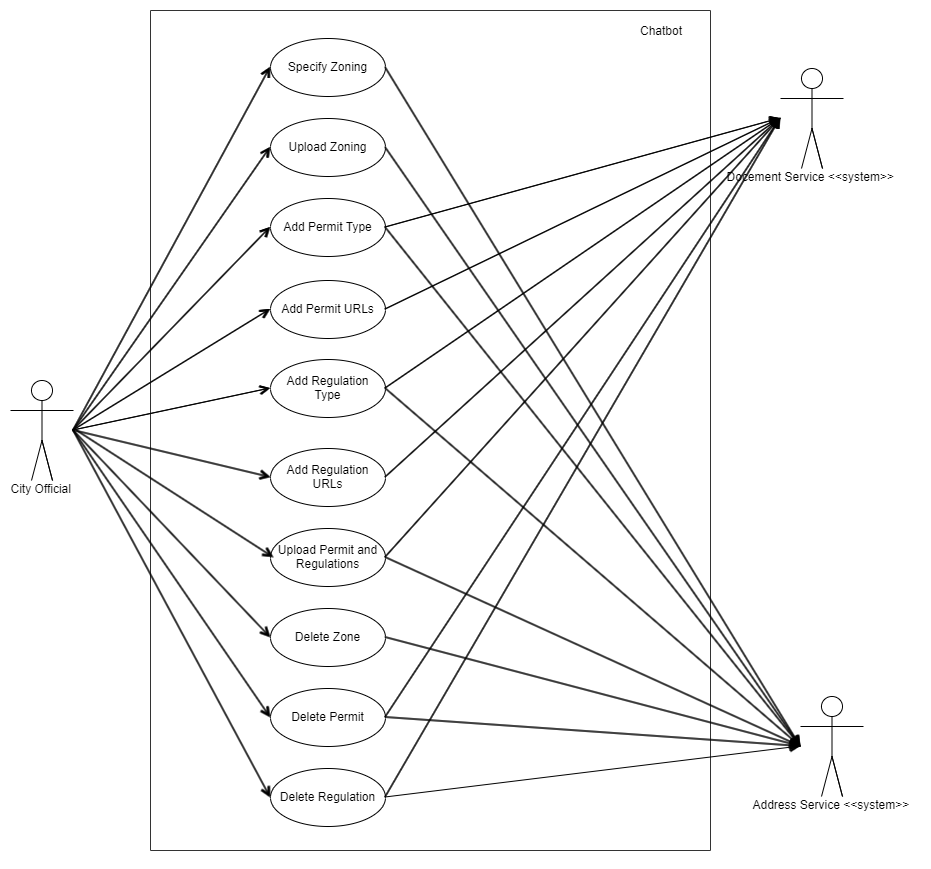
## Test Plan Identifier

User Acceptance Test Plan (UATP).

## Introduction

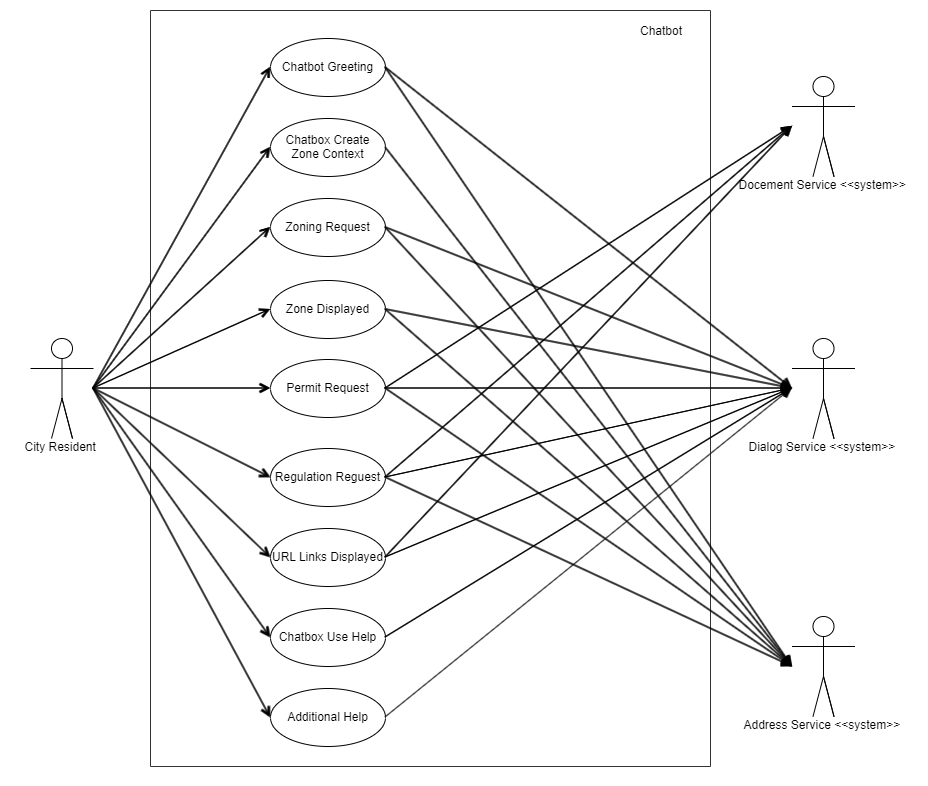
The Municipality Permit Chatbot system will consist of a Chatbot application that city residents will interact with, as well as a downloadable component that allows city officials to customize the offerings of the Chatbot. The system is composed of two primary subsystems, the City Resident Chatbot and the City Official Interface. The purpose of the City Resident Chatbot subsystem is to facilitate access to the features by residents, while the City Official Interface is to facilitate access to the features of the system by city officials.

The Municipality, Permit Chatbot system, is a software application that assists end-users in navigating the local city website (e.g., Pasadena) by answering questions and providing information when inquired. Questions provided by the end-user are administered via a popup website chatbox. The Chatbot convincingly acknowledges and interfaces with the end-user comparable to a human customer representative by determining whether to provide a direct answer or reference search results from within the website. Furthermore, the Chatbot is linked to other applications within the website to provide accurate information to the end-user. The Chatbot System (CS) will also contain options that will be customizable by its Stakeholders (e.g., City Officials) through a downloadable client.



*Figure 12.* City Official Interface Use Case Diagram

Figure 12 depicts the Use Case Diagram of the Chatbot system that has the City Official as the primary actor, providing a depiction of the system's expected behavior. The secondary actors here include the Document Service that will handle all the creation and storage of Permits and Regulations and the Address Service that will handle all the resident's address and zones.



*Figure 13*. City Resident Chatbot Use Case Diagram

Figure 13 depicts the Use Case Diagram of the Chatbot system that has the City Residents as the primary actor, providing a depiction of the system's expected behavior. The secondary actors here include the Document Service that will handle all the fetching of Permits and Regulations, the Dialog Service that will handle all the conversations with the users, and the Address Service that will handle all the city's address and zones. Furthermore, this document provides insight into the design of the Chatbot system from the end user's perspective, communicating the behavior in the user's terms by specifying all externally visible system behavior. In the following areas reside an overview of the UAT and its proposed deliverables, approaches, and results.

### Background

The UAT is being developed for the testing required for the Municipality Permit Chatbot system. This document is the master development testing plan, which will be utilized for the running of tests of this project. Tests will be developed throughout the project lifecycle based on the specifications outlined in this document.

### Objectives of the Test Plan

The purpose of the UAT Plan is to communicate to the development team and stakeholders a detailed plan for running development tests. The following sections of the UAT will outline the following:

1. The purpose of the UAT.
2. The scope of testing.
3. Detail the testing approach.
4. Define the expected deliverables.
5. Define the evaluation methods of the testing results
6. Estimate risks and proposed mitigations.
7. To assure that all stakeholders agree to the proposed details of the UAT.

### Objectives of UAT

The principal objectives of the UAT are:

1. Assuring that the functionality of deliverables adheres to project specifications.
2. Ensuring that the proposed functionality required of business scenarios has been delivered.

### References

1. Requirements Specification Document for the Municipality Permit Chatbot System.
2. IEEE829 International Testing Standards.

## Test Items

The following items of the Municipality Permit Chatbot System will be tested. The system is composed of two primary subsystems, the City Resident Chatbot and the City Official Interface.

The testable items of the Municipality Permit Chatbot System are as follows:

Table 9

*Testable Municipality Permit Chatbot System items corresponding version numbers*

|  |  |
| --- | --- |
| **Items to be Tested** | **Version Number** |
| **City Resident Chatbot** | 1.0 |
| **City Official Interface** | 1.0 |

## Features to Be Tested

The following features of the Municipality Permit Chatbot System will be tested. The system is composed of two primary subsystems, the City Resident Chatbot and the City Official Interface. The testable business processes of the City Resident Chatbot subsystem as a resident are as follows:

Table 10

*Testable MPCS Items within Chatbot*

|  |  |
| --- | --- |
| **Items to be Tested** | **Version Number** |
| **Chatbot greeting** | 1.0 |
| **Chatbot create zone context** | 1.0 |
| **Zoning request** | 1.0 |
| **Zone displayed** | 1.0 |
| **Permit request** | 1.0 |
| **Regulation request** | 1.0 |
| **URL links Displayed** | 1.0 |
| **Chatbox use help** | 1.0 |
| **Additional help** | 1.0 |

The testable business processes of the City Official Interface subsystem as an official are as follows:

Table 11

*Testable Business processes for City Official Interface System*

| **Items to be Tested** | **Version Number** |
| --- | --- |
| **Specify zoning** | 1.0 |
| **Upload zoning** | 1.0 |
| **Add permit type** | 1.0 |
| **Add permit URLs** | 1.0 |
| **Add regulation type** | 1.0 |
| **Add regulation URLs** | 1.0 |
| **Upload permit and regulations** | 1.0 |
| **Delete zone** | 1.0 |
| **Delete permit** | 1.0 |

## Features Not To Be Tested

Any additions that are not related to zoning, regulations, or permit applications are out of scope for this project.

## Approach

The test approach is broken down into four processes: Develop test, prepare the test, run test, and review test. The following sections outline these processes.

### Develop Tests

Tests will be developed utilizing the following testing tools:

1. JIRA
2. Angular
3. Jenkins

The following types of testing will be performed during System Integration Testing (SIT):

1. Acceptance testing: Development and executing the acceptance tests throughout the Sprint. Acceptance testing will be validated in the build-integration environment.
2. Unit testing: Developed to run automatically at development on-demand or by the server.
3. Functional testing: Performing test cases based on testable requirements
4. Regression testing: Ensuring changes to the system do not introduce new defects.

Test development will involve the following activities:

1. Analysis of requirements: Review that test works towards the completion of requirements.
2. Scenario development: Development of scenarios and required testing technique.
3. Define acceptance criteria: Check if the test meets accepted standards.
4. Construction of test case: Test case is defined based on expected input against expected output in regards to proving acceptance criteria.
5. Writing of test scripts: Writing and creation of test scripts.
6. Test documentation review: Review of all testing documentation for that test.

The following reviews will be conducted by the test team and the QA lead.

1. Test plan review
2. Test case review
3. Test progress & milestone reviews
4. Post-test review

### Prepare to Test

Test preparation will consist of the following three processes:

1. Preparation of test environment: Assuring that hardware and staffing are in place and ready to enable the required test.
2. Preparation of test data: Build all required data files and associated scripts required for the test case.
3. Preparation of testing documentation:Prepare test documentation to describe the entry criteria, procedure, and test reporting.

### Run Tests

The running of tests will involve the following two processes:

1. Running of test: Utilize the created test scripts and processes from the test case to run in the associated development environment.
2. Recording of results: Creation of test log of activities completed, and results. Incident defect severity detailed.

### Review Test Results

When testing is completed, acceptance of the system will be assessed. Incidents will be identified, assessed, and documented. Furthermore, each identified issue will be rank on severity. Test results will be checked against existing scenarios, requirements, and system impact. The quality assurance lead will work with the test team during evaluation. Tests will then be assessed based on their expected inputs and outputs against their actual outputs, whether they have passed or failed each respective test. At that time, defects will be assigned an appropriate level description based on severity.

### Test Results and defects

Defects that are discovered during test execution will be assessed, documented, and assigned an issue priority outlined in Table 13. All P1 issues will be subject to rework and are mandatory to be resolved, whereas P2 and P3 issues will be projected to the stakeholder for approval in resolving.

Table 12

*Issue Priority Ratings*

| Priority Descriptions |
| --- |
| Priority 1 (Critical) – P1 |
| * Prevents the accomplishment of fulfilling project completion of MPCS |
| * An error that produces a complete failure of the software |
| * There is no method of resolving (e.g., System crashes). |
| Priority 2 (Major) – P2 |
| * Components of the software are not working correctly. For instance, the user interface is working on how the information is not getting passed on. |
| * A potential workaround is possible with the presented issue. |
| Priority 3 (Minor) – P3 |
| * Software affected with the anomaly is non-critical functionality in the overall performance of the software. |
| * Desirable enhancements or new features |

## Item Pass/Fail Criteria

For each requirement, business process or system feature to be tested, the tester will execute a set of pre-defined test cases. Each test case will have a series of stimuli and expected responses. As each stimulus is performed, the results are evaluated. If the observed responses are equal to the expected results, a test is designated to have "passed." If the observed results are not equal to the expected results, a test is designated to have "failed."

The Test team uses JIRA for reporting, maintaining, tracking, and overall management of the defects on the Municipality Permit Chatbot System.

The assignment and description of defect severity levels will be as follows:

1. Critical: Business objectives or completion of the test case are impacted.
2. High: Defects that prove to be detrimental to the system. Testing likely should not progress to the next build until addressed.
3. Medium: Defects that provide invalid/incorrect information.
4. Low: Defects are aesthetic. Functionality is NOT impacted.
5. Info: An item observed during testing that may require further information. This type of priority could be assigned to a work order for an item encountered that is not clear in the requirements.

## Suspension Criteria and Resumption Requirements

Test criteria will be deemed to be normal when all test cases have been executed, the test will be suspended, and the results documented for the Test Summary Report. A test may be deemed to have run correctly when the testing team and quality lead are satisfied with the results, and defect severity level is assessed before implementation into the system.

If a critical processing unit is discovered to be defective, testing should be suspended until the defects have been fixed or otherwise addressed. When the addressed suspect unit is moved back into the test environment, any previously performed tests that affect the unit should be performed again to ensure new defects were not created as a result of the fix.

## Test Deliverables

The following deliverables are expected as a result of each test:

1. Testing Acceptance Plan: What should be done accomplished and tested on the UAT adhering to acceptance criteria.
2. Test Case: Values of inputs and expected output results from the test.
3. Test Log: Test results from running the test.
4. Incident Report: Observations from the test and documentation of any unexpected results.
5. Incident Report Log Updated: Summary of al incident reports.
6. Test Summary Report: Summary of the test, including all testing data.

## Testing Tasks

The following testing activities must be completed for each specified test:

1. Test plan prepared.
2. Functional specifications documented and delivered to the testing team or lead software developer.
3. Preparation of the testing environment.
4. Running of tests.
5. Summary and documentation of the test.

## Environmental Needs

Tests will be developed utilizing the following testing tools:

1. JIRA
2. Angular
3. Jenkins

Test environmental specifications and needs will be developed in conjunction with or based on the recommendations of the systems design lead or DevOps team.

## Responsibilities

The Test Lead is responsible for facilitating the testing of the system and all subsystems of the project.

The Test Lead will work in conjunction with the primary Systems Development Lead, DevOps, and the remaining team members in the capacity of the Test Team to fulfill the requirements outlined in the UAT. The Quality Assurance Lead will collaborate on signing off on all tests and at the conclusion of each test as well. The test team should understand the expectations of the collective project timeline, as well as the level of quality of the project as defined by the QA Lead and Project Manager.

Table 13

*Test Responsibilities*

|  |  |
| --- | --- |
| **Task** | **Assigned To** |
| **Test Plan Prepared** | Test Lead, Test Team, QA Lead |
| **Functional Specifications Documented** | Test Lead, Test Team, QA Lead |
| **Preparation of Test** | Test Lead, Test Team, Software Lead, Software Team |
| **Running of Test** | Software Lead, Software Team |
| **Summary of Test** | Test Lead, Test Team, Software Lead, QA Lead |

## Staffing and Training Needs

Testing will be done by the Test Lead, Test Team, and Software Lead. Quality assurance responsibilities will be coordinated between all parties and the Quality Assurance Lead.

In order to provide a proper testing environment for the project, the following needs will need to be addressed:

1. All parties involved in testing will need to be trained on the basic operations required of the testing environment.
2. All parties will need familiarization with the required reports and documentation.
3. At least the Test Lead and Software Lead need to be trained in the coordination and distribution of tests to the testing environment.

## Schedule

The remainder of the project will be completed in a series of 3 development sprints running from 7/1/20 through 7/30/20.

Table 14

*Sprint 4 Test Schedule*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sprint 4** | **Sevens days** | **Wed 7/1/20** | **Thu 7/9/20** | **Resident Chatbot** | **City Official Interface** |
| **Chatbot greeting** | 1 | 7/1/20 | 7/2/20 | X |  |
| **Chatbot create zone context** | 1 | 7/2/20 | 7/3/20 | X |  |
| **Zoning request** | 1 | 7/3/20 | 7/4/20 | X |  |
| **Zone displayed** | 1 | 7/4/20 | 7/5/20 | X |  |
| **Specify zoning** | 1 | 7/5/20 | 7/6/20 |  | X |
| **Upload zoning** | 2 | 7/6/20 | 7/8/20 |  | X |

Table 15

*Sprint 5 Test Schedule*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sprint 5** | **Sevens days** | **Wed 7/10/20** | **Thu 7/20/20** | **Resident Chatbot** | **City Official Interface** |
| **Permit request** | 1 | 7/10/20 | 7/11/20 | X |  |
| **Regulation request** | 1 | 7/11/20 | 7/12/20 | X |  |
| **URL links Displayed** | 1 | 7/13/20 | 7/14/20 | X |  |
| **Add permit type** | 1 | 7/15/20 | 7/16/20 |  | X |
| **Add permit URLs** | 1 | 7/16/20 | 7/17/20 |  | X |
| **Add regulation type** | 1 | 7/17/20 | 7/18/20 |  | X |
| **Add regulation URLs** | 1 | 7/18/20 | 7/19/20 |  | X |

Table 16

*Sprint 6 Test Schedule*

| **Sprint 6** | **Sevens days** | **Wed 7/20/20** | **Thu 7/30/20** | **Resident Chatbot** | **City Official Interface** |
| --- | --- | --- | --- | --- | --- |
| **Chatbox use help** | 2 | 7/20/20 | 7/22/20 | X |  |
| **Additional help** | 1 | 7/22/20 | 7/23/20 | X |  |
| **Upload permit and regulations** | 2 | 7/23/20 | 7/25/20 |  | X |
| **Delete zone** | 1 | 7/25/20 | 7/26/20 |  | X |
| **Delete permit** | 1 | 7/26/20 | 7/27/20 |  | X |

## Risks and Contingencies

For each risk identified, each risk and its cause should be identified, the likelihood of it happening should be assessed, as well as its potential impact on the project if it were to occur. A countermeasure will be developed to address each risk.

1. Creation and facilitation of testing with DevOps: Risk level moderate to high. Development environment changes not previously anticipated may be communicated by DevOps during development. To counter this, DevOps should be involved early communicated with regularly to avoid issues that may develop in the development environment. Should problems occur, timeline adjustments will be necessary.
2. Environmental development issues: Risk level low to moderate. The currently planned development environment may not be suitable for all development and testing needs. To counter this, environmental tools have been researched for compatibility with other tools required. Should problems occur, timeline adjustments will be necessary.
3. Time management issues: Risk level moderate to high. The current development cycle may be difficult to complete in the remaining five weeks to complete the system and both required subsystems. To counter this, features may need to be reassessed, and the timeline adjusted as necessary.
4. Acronyms and Abbreviations

The acronyms and abbreviations used in this document are defined in Table A-1 below.

Table 17 *Acronyms and Abbreviations*

| Acronym/Abbreviation | Definition |
| --- | --- |
| ACWP | Actual Cost of Work Performed |
| BAC | Budget At Completion |
| BCWP | Budgeted Cost Work Performed |
| BCWS | Budgeted Cost Work Schedule |
| CPI | Cost Performance Index |
| CS | Chat System |
| CV | Cost Variance |
| EAC | Estimate at Completion |
| EVM | Earned Value Measurement |
| FQT | Formal Qualification Test |
| IMS | Integrated Master Schedule |
| MPCS | Municipality Permit Chatbot System |
| OS | Operating System |
| OPS | Operations |
| PG | Programmer Guide |
| PP | Project Plan |
| PBI | Product Backlog Item |
| PR | Peer Review |
| PSI | Potentially Shippable Increments |
| PM | Program Manager |
| PSI | Potentially Shippable Increments |
| QA | Quality Assurance |
| SDLC | Software Development Life Cycle |
| SLOC | Software Lines Of Code |
| SI | Software Item |
| SPI | Schedule Performance Index |
| SRS | Software Requirement Specification |
| STP | Software Test Plan |
| SV | Schedule Variance |
| SVCS | Software Version Control System |
| SVD | Software Version Description |
| TDD | Technical Design Document |
| TD | Test Description |
| TR | Test Report |
| TRR | Test Readiness Review |
| UG | User Guide |
| UMGC | University of Maryland Global Campus |
| UML | Unified Modeling Language |
| WBS | Work Breakdown Structure |
|  |  |

1. IMS/WBS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| WBS | Task Name | Duration | Start | Finish |
| **1** | **Chatbot System** | **62 days** | **Wed 5/20/20** | **Thu 8/13/20** |
| **1.1** | **Milestones** | **40 days** | **Thu 6/11/20** | **Thu 8/6/20** |
| 1.1.1 | Milestone 1 | 0 days | Thu 6/11/20 | Thu 6/11/20 |
| 1.1.2 | Milestone 2 | 0 days | Tue 6/30/20 | Tue 6/30/20 |
| 1.1.3 | Milestone 3 | 0 days | Mon 7/20/20 | Mon 7/20/20 |
| 1.1.4 | Milestone 4 | 0 days | Thu 8/6/20 | Thu 8/6/20 |
| **1.2** | **Deliverables** | **41 days** | **Wed 6/10/20** | **Thu 8/6/20** |
| 1.2.1 | Project Plan | 0 days | Wed 6/10/20 | Wed 6/10/20 |
| 1.2.2 | SRS Document | 0 days | Thu 6/11/20 | Thu 6/11/20 |
| 1.2.3 | Software Test Plan | 0 days | Tue 6/30/20 | Tue 6/30/20 |
| 1.2.4 | Technical Design Document | 0 days | Tue 6/30/20 | Tue 6/30/20 |
| 1.2.5 | Programmer Guide | 0 days | Mon 7/20/20 | Mon 7/20/20 |
| 1.2.6 | Deployment and Operation Book | 0 days | Mon 7/20/20 | Mon 7/20/20 |
| 1.2.7 | User Guide | 0 days | Fri 7/31/20 | Fri 7/31/20 |
| 1.2.8 | Test Reports | 0 days | Thu 8/6/20 | Thu 8/6/20 |
| **1.2.9** | **Software Releases** | **21 days** | **Tue 6/30/20** | **Wed 7/29/20** |
| 1.2.9.1 | Engineering Release 1 | 0 days | Tue 6/30/20 | Tue 6/30/20 |
| 1.2.9.2 | Engineer Release 2 | 0 days | Mon 7/20/20 | Mon 7/20/20 |
| 1.2.9.3 | Engineering Release 3 | 0 days | Wed 7/29/20 | Wed 7/29/20 |
| **1.3** | **Initiation Phase** | **7 days** | **Wed 5/20/20** | **Thu 5/28/20** |
| 1.3.1 | Stakeholder Meeting | 1 day | Wed 5/20/20 | Wed 5/20/20 |
| 1.3.2 | Team Introduction | 4 days | Thu 5/21/20 | Tue 5/26/20 |
| 1.3.3 | Initial Team Meeting | 1 day | Wed 5/27/20 | Wed 5/27/20 |
| 1.3.4 | Project Kick-Off Meeting | 1 day | Thu 5/28/20 | Thu 5/28/20 |
| **1.4** | **Project Planning** | **9 days** | **Fri 5/29/20** | **Wed 6/10/20** |
| **1.4.1** | **Project Plan** | **9 days** | **Fri 5/29/20** | **Wed 6/10/20** |
| 1.4.1.1 | Create Project Plan | 4 days | Fri 5/29/20 | Wed 6/3/20 |
| 1.4.1.2 | Conduct Internal Review | 4 days | Thu 6/4/20 | Tue 6/9/20 |
| 1.4.1.3 | Closeout Internal Review | 1 day | Wed 6/10/20 | Wed 6/10/20 |
| 1.4.1.4 | Submit Project Plan Deliverable | 0 days | Wed 6/10/20 | Wed 6/10/20 |
| **1.5** | **Monitoring and Controlling** | **59 days** | **Wed 5/20/20** | **Mon 8/10/20** |
| **1.5.1** | **Stakeholder Meetings** | **48 days** | **Thu 6/4/20** | **Mon 8/10/20** |
| **1.5.1.1** | **Technical Review #1** | **7 days** | **Thu 6/4/20** | **Fri 6/12/20** |
| 1.5.1.1.1 | Create Draft Presentation | 2 days | Thu 6/4/20 | Fri 6/5/20 |
| 1.5.1.1.2 | Team Review of Presentation | 2 days | Mon 6/8/20 | Tue 6/9/20 |
| 1.5.1.1.3 | Submit Slides to Stakeholder | 1 day | Wed 6/10/20 | Wed 6/10/20 |
| 1.5.1.1.4 | Conduct Technical Review 1 | 1 day | Thu 6/11/20 | Thu 6/11/20 |
| 1.5.1.1.5 | Obtain Stakeholder Approval to Progress | 1 day | Fri 6/12/20 | Fri 6/12/20 |
| **1.5.1.2** | **Technical Review #2** | **7 days** | **Wed 6/24/20** | **Thu 7/2/20** |
| 1.5.1.2.1 | Create Draft Presentation | 2 days | Wed 6/24/20 | Thu 6/25/20 |
| 1.5.1.2.2 | Team Review of Presentation | 2 days | Fri 6/26/20 | Mon 6/29/20 |
| 1.5.1.2.3 | Submit Slides to Stakeholder | 1 day | Tue 6/30/20 | Tue 6/30/20 |
| 1.5.1.2.4 | Conduct Technical Review 2 | 1 day | Wed 7/1/20 | Wed 7/1/20 |
| 1.5.1.2.5 | Obtain Stakeholder Approval to Progress | 1 day | Thu 7/2/20 | Thu 7/2/20 |
| **1.5.1.3** | **Technical Review #3** | **8 days** | **Mon 7/13/20** | **Wed 7/22/20** |
| 1.5.1.3.1 | Create Draft Presentation | 2 days | Mon 7/13/20 | Tue 7/14/20 |
| 1.5.1.3.2 | Team Review of Presentation | 2 days | Wed 7/15/20 | Thu 7/16/20 |
| 1.5.1.3.3 | Submit Slides to Stakeholder | 1 day | Fri 7/17/20 | Fri 7/17/20 |
| 1.5.1.3.4 | Conduct Technical Review 3 | 1 day | Mon 7/20/20 | Mon 7/20/20 |
| 1.5.1.3.5 | Obtain Stakeholder Approval to Progress | 1 day | Wed 7/22/20 | Wed 7/22/20 |
| **1.5.1.4** | **Technical Review #4** | **15 days** | **Tue 7/21/20** | **Mon 8/10/20** |
| 1.5.1.4.1 | Create Draft Presentation | 8 days | Tue 7/21/20 | Thu 7/30/20 |
| 1.5.1.4.2 | Team Review of Presentation | 3 days | Fri 7/31/20 | Tue 8/4/20 |
| 1.5.1.4.3 | Submit Slides to Stakeholder | 1 day | Wed 8/5/20 | Wed 8/5/20 |
| 1.5.1.4.4 | Conduct Technical Review 4 | 1 day | Thu 8/6/20 | Thu 8/6/20 |
| 1.5.1.4.5 | Obtain Stakeholder Approval to Progress | 1 day | Mon 8/10/20 | Mon 8/10/20 |
| **1.5.2** | **Internal Quality Assurance Audits** | **55 days** | **Wed 5/20/20** | **Tue 8/4/20** |
| 1.5.2.1 | Conduct Audit 1 | 7 days | Wed 5/20/20 | Thu 5/28/20 |
| 1.5.2.2 | Conduct Audit 2 | 7 days | Wed 7/1/20 | Thu 7/9/20 |
| 1.5.2.3 | Conduct Audit 3 | 7 days | Fri 7/10/20 | Mon 7/20/20 |
| 1.5.2.4 | Conduct Audit 4 | 7 days | Tue 7/21/20 | Wed 7/29/20 |
| 1.5.2.5 | Conduct Audit 5 | 4 days | Thu 7/30/20 | Tue 8/4/20 |
| **1.6** | **Execution Phase** | **50 days** | **Fri 5/29/20** | **Thu 8/6/20** |
| 1.6.1 | Create Product Backlog | 3 days | Fri 5/29/20 | Tue 6/2/20 |
| **1.6.2** | **Sprints (weekly)** | **41 days** | **Wed 6/3/20** | **Wed 7/29/20** |
| **1.6.2.1** | **Sprint 1** | **7 days** | **Wed 6/3/20** | **Thu 6/11/20** |
| 1.6.2.1.1 | Sprint Planning | 1 day | Wed 6/3/20 | Wed 6/3/20 |
| 1.6.2.1.2 | Sprint Execution | 6 days | Thu 6/4/20 | Thu 6/11/20 |
| 1.6.2.1.3 | Sprint End/Demo | 0 days | Thu 6/11/20 | Thu 6/11/20 |
| **1.6.2.2** | **Sprint 2** | **7 days** | **Fri 6/12/20** | **Mon 6/22/20** |
| 1.6.2.2.1 | Sprint Planning | 1 day | Fri 6/12/20 | Fri 6/12/20 |
| 1.6.2.2.2 | Sprint Execution | 6 days | Mon 6/15/20 | Mon 6/22/20 |
| 1.6.2.2.3 | Sprint End/Demo | 0 days | Mon 6/22/20 | Mon 6/22/20 |
| **1.6.2.3** | **Sprint 3** | **6 days** | **Tue 6/23/20** | **Tue 6/30/20** |
| 1.6.2.3.1 | Sprint Planning | 1 day | Tue 6/23/20 | Tue 6/23/20 |
| 1.6.2.3.2 | Sprint Execution | 5 days | Wed 6/24/20 | Tue 6/30/20 |
| 1.6.2.3.3 | Sprint End/Demo | 0 days | Tue 6/30/20 | Tue 6/30/20 |
| **1.6.2.4** | **Sprint 4** | **7 days** | **Wed 7/1/20** | **Thu 7/9/20** |
| 1.6.2.4.1 | Sprint Planning | 1 day | Wed 7/1/20 | Wed 7/1/20 |
| 1.6.2.4.2 | Sprint Execution | 6 days | Thu 7/2/20 | Thu 7/9/20 |
| 1.6.2.4.3 | Sprint End/Demo | 0 days | Thu 7/9/20 | Thu 7/9/20 |
| **1.6.2.5** | **Sprint 5** | **7 days** | **Fri 7/10/20** | **Mon 7/20/20** |
| 1.6.2.5.1 | Sprint Planning | 1 day | Fri 7/10/20 | Fri 7/10/20 |
| 1.6.2.5.2 | Sprint Execution | 6 days | Mon 7/13/20 | Mon 7/20/20 |
| 1.6.2.5.3 | Sprint End/Demo | 0 days | Mon 7/20/20 | Mon 7/20/20 |
| **1.6.2.6** | **Sprint 6** | **7 days** | **Tue 7/21/20** | **Wed 7/29/20** |
| 1.6.2.6.1 | Sprint Planning | 1 day | Tue 7/21/20 | Tue 7/21/20 |
| 1.6.2.6.2 | Sprint Execution | 6 days | Wed 7/22/20 | Wed 7/29/20 |
| 1.6.2.6.3 | Sprint End/Demo | 0 days | Wed 7/29/20 | Wed 7/29/20 |
| 1.6.3 | Formal Qualification Test | 1 day | Thu 7/30/20 | Thu 7/30/20 |
| **1.6.4** | **Documentation Execution** | **47 days** | **Wed 6/3/20** | **Thu 8/6/20** |
| **1.6.4.1** | **Software Requirement Specification (SRS)** | **7 days** | **Wed 6/3/20** | **Thu 6/11/20** |
| 1.6.4.1.1 | Create Draft SRS Document | 4 days | Wed 6/3/20 | Mon 6/8/20 |
| 1.6.4.1.2 | Conduct Internal Review - SRS | 2 days | Tue 6/9/20 | Wed 6/10/20 |
| 1.6.4.1.3 | Closeout Internal Review- SRS | 1 day | Thu 6/11/20 | Thu 6/11/20 |
| 1.6.4.1.4 | Submit SRS Document to Stakeholder | 0 days | Thu 6/11/20 | Thu 6/11/20 |
| **1.6.4.2** | **Software Test Plan (STP)** | **12 days** | **Mon 6/15/20** | **Tue 6/30/20** |
| 1.6.4.2.1 | Create Initial Draft Test Plan | 7 days | Mon 6/15/20 | Tue 6/23/20 |
| 1.6.4.2.2 | Conduct Internal Review - TP | 3 days | Wed 6/24/20 | Fri 6/26/20 |
| 1.6.4.2.3 | Closeout Internal Review- TP | 1 day | Mon 6/29/20 | Mon 6/29/20 |
| 1.6.4.2.4 | Incorporate Test Plan into Project Plan | 1 day | Tue 6/30/20 | Tue 6/30/20 |
| 1.6.4.2.5 | Submit Test Plan Material to Stakeholder | 0 days | Tue 6/30/20 | Tue 6/30/20 |
| **1.6.4.3** | **Technical Design Document (TDD)** | **19 days** | **Thu 6/4/20** | **Tue 6/30/20** |
| 1.6.4.3.1 | Create Draft TDD Document | 14 days | Thu 6/4/20 | Tue 6/23/20 |
| 1.6.4.3.2 | Conduct Internal Review - TDD | 4 days | Wed 6/24/20 | Mon 6/29/20 |
| 1.6.4.3.3 | Closeout Internal Review- TDD | 1 day | Tue 6/30/20 | Tue 6/30/20 |
| 1.6.4.3.4 | Submit TDD Document to Stakeholder | 0 days | Tue 6/30/20 | Tue 6/30/20 |
| **1.6.4.4** | **Programmer Guide (PG)** | **19 days** | **Wed 6/24/20** | **Mon 7/20/20** |
| 1.6.4.4.1 | Create Draft PG | 14 days | Wed 6/24/20 | Mon 7/13/20 |
| 1.6.4.4.2 | Conduct Internal Review - PG | 4 days | Tue 7/14/20 | Fri 7/17/20 |
| 1.6.4.4.3 | Closeout Internal Review- PG | 1 day | Mon 7/20/20 | Mon 7/20/20 |
| 1.6.4.4.4 | Submit PG Document to Stakeholder | 0 days | Mon 7/20/20 | Mon 7/20/20 |
| **1.6.4.5** | **Deploy and Operations Book (OPS)** | **19 days** | **Wed 6/24/20** | **Mon 7/20/20** |
| 1.6.4.5.1 | Create Draft OPS Document | 13 days | Wed 6/24/20 | Mon 7/13/20 |
| 1.6.4.5.2 | Conduct Internal Review - OPS | 4 days | Tue 7/14/20 | Fri 7/17/20 |
| 1.6.4.5.3 | Closeout Internal Review- OPS | 1 day | Mon 7/20/20 | Mon 7/20/20 |
| 1.6.4.5.4 | Submit OPS Document to Stakeholder | 0 days | Mon 7/20/20 | Mon 7/20/20 |
| **1.6.4.6** | **User Guide (UG)** | **15 days** | **Mon 7/13/20** | **Fri 7/31/20** |
| 1.6.4.6.1 | Create Draft UG Document | 10 days | Mon 7/13/20 | Fri 7/24/20 |
| 1.6.4.6.2 | Conduct Internal Review - UG | 4 days | Mon 7/27/20 | Thu 7/30/20 |
| 1.6.4.6.3 | Closeout Internal Review- UG | 1 day | Fri 7/31/20 | Fri 7/31/20 |
| 1.6.4.6.4 | Submit UG Document to Stakeholder | 0 days | Fri 7/31/20 | Fri 7/31/20 |
| **1.6.4.7** | **Test Report** | **6 days** | **Thu 7/30/20** | **Thu 8/6/20** |
| 1.6.4.7.1 | Create Draft TR Document | 3 days | Thu 7/30/20 | Mon 8/3/20 |
| 1.6.4.7.2 | Conduct Internal Review - TR | 2 days | Tue 8/4/20 | Wed 8/5/20 |
| 1.6.4.7.3 | Closeout Internal Review- TR | 1 day | Thu 8/6/20 | Thu 8/6/20 |
| 1.6.4.7.4 | Submit TR Document to Stakeholder | 0 days | Thu 8/6/20 | Thu 8/6/20 |
| **1.7** | **Project Closeout** | **5 days** | **Fri 8/7/20** | **Thu 8/13/20** |
| 1.7.1 | Finalize Technical Data Package | 3 days | Fri 8/7/20 | Tue 8/11/20 |
| 1.7.2 | Launch Chatbot | 2 days | Wed 8/12/20 | Thu 8/13/20 |