CPSC 3720 SPRING 2020

C-Track:

An Issue Tracking System Project Proposal

CZAR

Group C

William Hughson, Dustin Ward, Don Castillo

October 16th, 2020

TABLE OF CONTENTS

INTRODUCTION

What is an Issue Tracking System?

Nowadays, people rely on issue tracking systems to manage their workflow and get an overall view of their productivity. An **issue tracking system** is a web application or software package that maintains a list of issues (in a form of ticket), sometimes sorted based on their priorities, that are to be resolved. From a software development standpoint, it keeps track of what the bugs are and their current statuses, whether they are fixed or not by developers. It is also where the project manager or the client would create an issue ticket for feature requests. The issue tracking system is also used by individuals as part of their time management and productivity routine. From a business standpoint, it allows customers to report any problems in using the products or services to the business owners. Jira, Bugzilla, and Trello are just some of the popular issue tracking systems.

An issue tracking system is designed to illuminate all aspects associated with issues that obstruct a system from operating efficiently. With any system comes end-users that operate the system day in and day out. Without an issue tracking system if that end-user discovers a problem, he will have to contact a support line with information about the issues, then the support line might have to contact someone else and some information could be lost in translation. What happens if the support line is closed for the day? The end-user, unfortunately, will have to wait until it reopens and relay the information from many hours ago. Now with an issue tracking system, all this information can be imported the moment it happens with clear concise instructions.

Introducing C-Track

Team Czar has been tasked to develop and provide documentation for an issue tracking system which we call **C-Track** (Czar Track). Why choose team Czar for developing, maintaining, and providing documentation for an issue tracking system like C-Track? Team Czar is the proper choice for a multitude of reasons. We pride ourselves on developing software that is cost and hardware efficient, user-friendly, and most importantly gets the job done. Furthermore, each member of Czar has extensive customer service training which is critical when translating business needs to a functional system.

The C-Track application will offer a variety of features designed to help businesses and individuals customize their issue tickets and make the team workflow and customer support manageable.

This report was written to clearly outline our plan for developing C-Track. This includes the proposal section, the project management section, and the design section. The **proposal section** provides a description of what C-Track is and how it is expected to function based on different use cases. The **project management section** provides an insight on who the members of the team are, how the team will proceed in developing and completing the C-Track application, and how the team will address foreseeable problems. Lastly, the **design section** describes the planned backend design of C-Track.

PROPOSAL

C-track is a state-of-the-art issue tracking system that allows individuals to track, view, solve, and update problems in real time. C-track enables individuals the ability to report issues with a title and a short description of the problem as well as how it occurred. Furthermore, C-track allows developers the ability to view, update, track, and delete everything associated with an issue. This includes, but is not limited to, assigning and setting the status of an issue. In addition, developers have the ability to create, add, delete, and list users in the system. C-track also comes with bonus features that allows developers to view the type of issue, track the reporter of the issue, create user groups, and find issues that match multiple fields or parameters. See Appendix A for a detailed use case of C-track.

PROJECT MANAGEMENT

This section details who the team members are, how the team will complete the project, and how the team will address any foreseeable problems the team may encounter throughout the software development process. This section includes the **team organization section** which introduces the members of the Czar team, the risk **management section** which describes the team's plan of action when issues related to requirements, project design, and human resource arise, and the **project schedule section** which shows what features of the project will be completed in which sprint.

Team Organization

This section introduces the members of team Czar. The team is composed of B.S. Computer Science students at the University of Lethbridge:

1. William Hughson
2. Dustin Ward
3. Don Castillo

The goal of this project is not only to develop the issue tracking system, C-Track, but also for the team to gain more experience in agile software development and apply good software engineering practices as we go through the sprints. Every member of the team is expected to fill the roles of software developer and tester.

Risk Management

Any successful team-oriented project requires, but are not limited to, communication, organization, patience, leadership, and risk management. Risk management is crucial because not having a plan for unexpected incidents could become catastrophic. During the construction of the issue tracking system team Czar has acknowledged several roadblocks that can hinder or derail the completion of this project.

1. Requirements, Design, and Estimation

A. **The team planned a project that is too large.**

The team recognizes that understanding the requirement of the project is necessary in order to define the scope of the project and prevent scope creep,

which is defined as the continuous and uncontrolled growth of the project. We do not want to add features that are not required in the course or not expected by the instructor or marker. To prevent confusion among developers, the team will conduct a regular standup meeting where each member will report or update the progress of their tasks to the other members of the team. This ensures that no duplication of tasks or overlapping responsibilities happen. The summary of the meeting will be reported to the marker or instructor on MS Teams.

If the team realizes that the project has grown, we will revisit the project guidelines. We will scrap the features that are not in the original plan and move on to features that are required.

B. The team underestimated how long parts of the project would take.

Team Czar is committed to make many features available as early as possible through the use of agile methodology. We expect that features will be partially added incrementally for every sprint, and be fully functional in the subsequent sprints. To make this possible, the team will develop the easiest feature to implement first and leave the challenging ones at the end of the development process. This ensures that there is a working prototype after each sprint.

The team will make use of the issue tracking system available on Gitlab to keep track of the team's productivity. Each member will be assigned his own tasks and report its progress by changing the task ticket on the issue tracking board. During the stand-up meeting, the team will review what has been done and create new tickets for the next sprint.

C. Major changes to design are needed during implementation

Before the actual implementation, the team will come up with a design plan that would guide all the developers in developing the software. If unsure, the team will consult the marker or the instructor to review the design.

Any possible changes to the design of the project will be discussed thoroughly in a team meeting. If the changes are really necessary and there is enough time to do it, then they will be committed to a separate branch. If the changes work as intended, they will be merged into the main branch after they are tested. If the changes are unsuccessful, we will abandon it.

2. People

A. Addition or loss of team member.

Each member of the team is expected to contribute to the project. If a member decides to leave, he has to inform the team immediately about his decision to leave and provide an update to the current status of his assigned task. If someone will replace him, he also has to brief the new member about the pending tasks that are assigned to him. The new member will also be oriented to the current situation of the team during the meeting. Moreover, the rest of the team will help and guide the new member until he is fully integrated.

B. Unproductive team member(s).

This project requires the participation of all the members of the team. If a member is not doing his assigned task, the rest of the team will ask the member of the reason why he is slacking off. If the member is not adept or lacks the skills in completing the tasks he will be helped or assigned to other tasks like video presentation, documentation, or testing. If the reason of the member is lack of

time or commitment with the other classes, this will be reported to the marker or the instructor.

C. Team member(s) lacking expected background.

As mentioned above, the rest of the team will help or guide the team member who may have difficulty doing his assigned task because of lack of skills or background knowledge. In other circumstances, he may be assigned to other nontechnical tasks such as documentation and presentation. At this point, it is expected that the members of the team have some background knowledge or practical experience in team-based software development which they gained from CPSC 2720 - Practical Software Development.

D. Illness or unanticipated life events.

Software developers are humans, not robots. Sometimes there are events that are out of the team's control. If any of the members is experiencing issues related to family, health, or another personal predicament, he will be asked to inform the marker or instructor about that. The personal issues that the member may have could affect the overall productivity of the team. This is not an ideal situation since there are deadlines that we have to meet.

3. Learning and Tools

A. Inexperience with new tools.

There is no assurance that all the members of the team will use the same tools recommended in class since everyone will be working virtually. It is presumed that each member will be using some tools based on his preference like text editor and terminal. Microsoft Teams will be the primary communication

channel of the team where stand up meetings will happen and where sprint deliverables will be submitted. The team believes that there would be no problem regarding the inexperience of any member with any tools. If a member encounters a technical problem during the project development, he is encouraged to ask questions to the team, the marker, or the instructor.

B. Learning curve for tools steeper than expected.

In this project, we will be using some tools that might not be familiar with the team like REST APIs, JSON file format, and other libraries We will be researching some of these online or ask the marker or instructor for help. Because of tight deadlines, we will try our best to get familiar with these required tools as fast as possible.

C. Tools don’t work together in an integrated way.

The good thing about the project is that, as a team, we will be following the test-driven development principle and continuous integration practices in software engineering. Every time, members commit and push changes to the remote repository, it will give us an idea where possible bugs come from or whether the implementations made by a member is not compatible due to the tools used. If an integration issue arises, we will revert to the previous commits and try to fix the issue from there.

Project Schedule

This section outlines the deliverables or features that will be completed in each sprint.

Timeline	Scheduled Items
Sprint #1 (v1.0)	● Proposal Document ● Endpoint Design ● Process Planning
Sprint #2 (v1.1)	● Model Design & Implementation ○ Issue ○ Comment ○ User ● Start Initial API Requests ○ Get familiar with RESTbed ○ Simple GET requests
Sprint #3 (v1.2)	● Finish Initial API Requests ○ GET requests ○ Using hard coded data ● Start Updating/Deleting data ○ POST requests ○ DELETE requests ○ PUT requests
Sprint #4 (v1.3)	● Finish Updating/Deleting data ○ POST requests ○ DELETE requests ○ PUT requests ● Interface Design ○ CLI output ○ Possible Web-Interface ○ User Input ● Finalizing Documentation

DESIGN

This section shows the planned design of the REST endpoints. Each endpoint services a particular use case mentioned in the proposal section.

Use Case	REST Endpoint	JSON Response
GET the IDs of issues from the start index to the end index. (Useful for pagination) Could also just return the entire Issue instead of ID. Also, an optional parameter to only get issues of a certain type	/issues?start=<idx>,end=<idx> >, type=<issueType>	{“issues”: [“<issueID>”, …] }
GET the info for a specific issue. Provide the issue ID. The data returned will change as we add or remove features.	/issues/{id}	{“title”: “<string>”, “body”: “<string>”, “type”: “<string>”, “reporter”: “<userID>” Etc.. }
POST a new issue. Doesn’t have to return much. Maybe just a status code based on if the issue was created properly or not. This endpoint could also be used for editing an issue	/issues	{“status”: <statusCode> }

GET all comments for a certain issue. I don't think we will need to store these separately like issues. We could also implement pagination, but that might be too much work. Should also **POST** a new comment to the issue.	/issues/{id}/comments	{"comments": [{"id": "<commentID>", "body": "<string>", "user": "<userID>", Etc… }, {*next comment*}, …]}
GET a specific comment by ID. Can also be used to edit a comment	/issues/{id}/comments/{id}	{"id": "<commentID>", "body": "<string>", "user": "<userID>", Etc… }
GET an array of all user IDs. Optional parameter for searching by group (user/tester/etc) Should also **POST** a new user	/users?group=<groupType>	{"users": ["<userID>", …] }
GET a user by ID. Should also be able to update their info	/users/{id}	{"id": "<userID>", "name": "<string>", "picture": "<imageURL>" "group": <string>, Etc… }

APPENDICES

A. C-Track Use Case Diagram