

# Trackinator

---



Knick-Knack

Tyler Justinen-Teite

Victor Besson

Liane Goritz

Ryan Knight

Oct 16, 2020

## TABLE OF CONTENTS

Introduction	2
Proposal	2
Project Management	4
Risk Management	4
Team Organization	5
Software Design	6
Design	6
Design Rationale	6
Appendices	8

## INTRODUCTION

Our project proposal is broken down into 3 main sections, proposal, project management and design. First we will go over how we intend to create a REST based ticketing service, using the REST API and C++. It will be important, as we are all working remotely, to keep up to date with our documentation so that it is easier to figure out what others code is doing, this will also help write the user manual, or any other required documentation, as required.

In order to understand how our issue tracking system will be exactly just that, we will explain the basic principles of an issue tracking system. Firstly, these types of systems are essentially just small text files. These are typically used to track progress and/or problems with the current iteration of a project. Each issue will be defined by the title, a description, who it's assigned to, and the priority of the issue. To allow input on issues in the software, users will be allowed to add comments on issues.

## PROPOSAL

Knick-Knack is going to create a ticket management system. The primary use case of the system will be a database that allows users to track development and maintenance of their own system. We will be aiming to create a very simple UI, and a backend with no bells-and-whistles. Our system will take in a ticket request, and assign it to a user who will complete the work required for the issue.

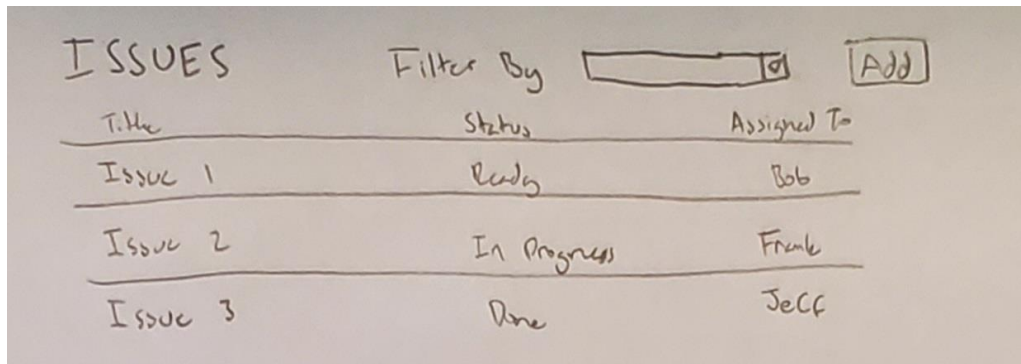
To make our tracking system unique we have decided to implement ticket **PRIORITY**, **REPORTER**, and **OPERATING-SYSTEM**. Priority means you can assign how important a ticket is (i.e. 1-5), the reporter allows you to see who created the ticket, and the operating system allows you to see what system the ticket is in reference to.

The searching feature that we will be adding is the ability to search by an **ATTRIBUTE**. This will allow a user of the application to find an issue by matching an attribute to the search term.

Prototypes and images:

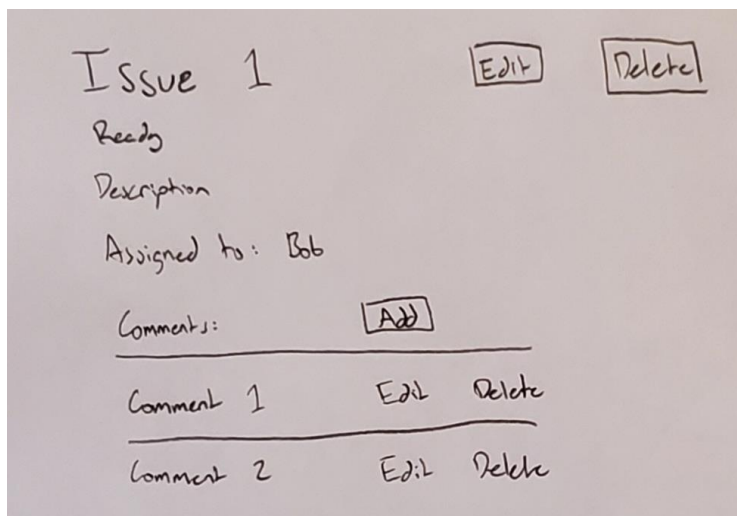
The following are mockups of the pages that will be used by the view layer of the application.

Issues Page:



Title	Status	Assigned To
Issue 1	Ready	Bob
Issue 2	In Progress	Frank
Issue 3	Done	Jeff

View/Edit Issue Page:



Issue 1

Ready

Description

Assigned to: Bob

Comments:

Comment 1	Edit	Delete
Comment 2	Edit	Delete

Create Issue Page:

CREATE ISSUE

Title:

Description:

Assign:

Status:

Users Page:

username	f name	l name	
Bob B	Bob	Bobberson	Delete
FrankF	Frank	Frankerson	Delete

## PROJECT MANAGEMENT

### RISK MANAGEMENT

#### I. Requirements/Design/Estimation

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

Cut down the advanced bits or anything that is not required.

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

Increase work time on the project. Or finish what was needed in the next sprint if able.

##### C. Major changes to design are needed during implementation

Assess the need for change and attempt to address the design changes quickly.

## II. People

### A. Addition or loss of team member

If a member was added, catch the new member up on where the team is currently at. If a member was lost, increase workload between members equally.

### B. Unproductive team members(s)

First step - Ask them what they want to do.

Second step - Give them a task to do.

Third step - Tell them politely that they need to put in more effort/work.

Final Step - potentially remove member(s).

### C. Team member(s) lacking expected background

If they are unable to learn what is needed of them to be able to do equal work, they must be removed.

### D. Illness or unanticipated life events

Distribute the work that was needed between the rest of the team equally.

## III. Learning & Tools

### A. Inexperience with new tools

We formed a team such that we can delegate everything we don't know to someone else in the group.

### B. Learning curve for tools steeper than expected

If the tools are too complicated they aren't worth using.

### C. Tools don't work together in an integrated way

If the tools don't work together, they're too complicated and are not worth using.

## PROJECT SCHEDULE

### Sprint 1

- ☐ Create Issues
- ☐ Get Issues
- ☐ List All Issues
- ☐ Assign Issues
- ☐ Update Issues
- ☐ Set Title
- ☐ Update Title

- ☐ Set Status
- ☐ Update Status
- ☐ Set Priority
- ☐ OS That The Issue Affects

#### Sprint 2

- ☐ Add Comment To An Issue
- ☐ Get A Comment From An Issue
- ☐ Get All Of An Issue's Comments
- ☐ Update A Comment Of An Issue
- ☐ Delete A Comment From An Issue

#### Sprint 3

- ☐ Create/Add An User
- ☐ List All Users
- ☐ Remove A User
- ☐ Searching A Single Attribute
- ☐ Reporter For Issue

## SOFTWARE DESIGN

### DESIGN

The software will be developed for use as a command line/web application. The user of the program will be able to perform operations on the data in the application by using the provided interface to communicate with the REST endpoints listed in the table in appendix 1.0. The interface (or the view) will send data in a URI to the REST endpoint where the application will perform operations on the issues, users, and comments stored in the backend. The backend will then return a code that indicates the status of the operation (successful, failed) as well as information that the console can display regarding the application. The format of the data that is returned from the server is in JSON, allowing a great degree of flexibility for the view to interact with the user.

### Design Rationale

The REST endpoints provide a way to separate the view from the controller of the application, allowing the interface to be built completely separate from the business logic itself. Such a design allows a variety of views to be created, such as web interfaces and standalone applications, and have them access and control the same data. The codes that get returned by the REST server allow the views to know the status of the data and what to show the user after a JSON response is returned from the server.





## APPENDICES

### 1.0: Table of REST Endpoints

Use Case	REST Endpoint	JSON Response
Create Issue	/issue/create?title=_&description=_&assign=_&status=_&priority=_	{"code":_,"id":_}
Get Issue	/issue/get?id=_	{"code":_,"id":_,"title":_,"desc":_,"os":_,"assign":_,"priority":_}
Edit Issue	/issue/edit?id=_&title=_&description=_&assign=_&status=_&priority=_	{"code":_,"id":_}
Delete Issue	/issue/delete?id=_	{"code":_}
Get all Issues	/issue/list	{"issues":[{"title":_,"status":_,"assign":_}]}
Set OS for Issue	/issue/setos?os=_&id=_	{"code":_}
Add Comment to Issue	/comment/add?issueid=_&comment=_&user=_	{"code":_,"commentid":_}
Get a comment	/comment/get?id=_	{"code":_,"comment":_,"user":_}
Get all comments for an issue	/comment/getall?issueid=_	{"code":_,"comments":[{"user":_,"comment":_}]}
Update comment	/comment/edit?id=_&comment=_	{"code":_}
Delete comment	/comment/delete?id=_	{"code":_}
Add user	/user/add?fname=_&lname=_&username=_	{"code":_,"username":_}
List all users	/user/list	{"code":_,"users":[{"username":_,"fname":_,"lname":_}]}
Remove user	/user/remove?username=_	{"code":_}

Add reporter	/issue/reporter?username=_&issueid=_	{"code":_}
Search by attribute	/issue/search?attribute=?&search=_	{"issues":[{"title":_,"status":_,"assign":_}]}