CPSC 3720 Fall 2020

Angel Tracker v.1.0

Thomas Richardson
Stephen Rau
Tommy DeLair
Jordan Leibel

Oct 16, 2020

Table of Contents

Introduction

Improve your company's time management and productivity using Angel Tracker™!

Angel Tracker™, developed by Irk, will be the new bleeding edge web-based issue tracking system for all professionals in collaborative work settings. Angel Tracker™ will manage and organize issues organizations face when producing and maintaining a product. You will be sure to love the sleek and intuitive interface that is easy to learn and use.

Below you will find more information regarding what is to come from Angel Tracker™:

Proposal - Description of the Angel Tracker™ web service application.

Design - Planned design of the REST endpoints of our issue tracker.

Project Management - Which includes or team Organization, Risk management strategies, and overall project Timeline.

Proposal

The Objective of this Application is to create standalone software that allows for multiple users to track issues within a given software. This software will have the features listed below;

Issues:

Create - creates a new issue

Get - retrieves an issues from server

List all - lists all issues

Assign - assigns an issue to a user

Delete - deletes an issue

Set title - sets a title of a issue

Update title - renames the title of a issue

Set status - sets the status of an issue to

Update status - updates the status of a issue

Component - what system components does this affect

OS - what operating system does this issue affect

Priority - how important is the issue

Search - find using a single attribute

Comments:

Add to an issue - to add additional information to an issue

Get a comment from an issue - to retrieve a single comment from a given issue.

Get all of an issue's comments - to retrieve all comments associated with an issue.

Update a comment of an issue - to change a comment for a given issue

Delete a comment from an issue - to delete a given comment from an issue.

Users:

Create / Add - To create a new user.

List all - to list all users.

Remove a user - to delete a user.

Design

For our Issue tracker we will be using a combination of C++ as our frontend and JSON data files as our back end. C++ functions will make calls between our client and server applications. This will allow for easy integration and portability between various OS due to the large user base and libraries for the language. While JSON will be restricted to our server file and serve as data storage. This allows for straightforward design, with the absence of the need for database management and a query language like SQL. Listed in Appendix A is a table of the various functions, REST endpoints, and JSON response that our software will use.

Project Management

Project Management. Provide a description of how you will complete the project, and address any foreseeable problems. This section will have two subsections:

Team Organization.

Stephen Rau - Scrum Master

Thomas Richardson - UI Specialist

Tommy DeLair - MiddleWare Specialist

Jordan Leibel - Database Specialist

Risk Management

Noted below are a list of possible risks that could prevent the team from completing the project. Below is a list of anticipated Risks and action plans to remedy possible situations.

Risk: The team planned a project that is too large (i.e. “eyes bigger than stomach”).

Plan: Break the project into sizable chunks or sprints. Which will be managed by scrum master. Keep track of progress and determine if the deadline will be met or not. If we think the deadline will not be met, determine what features should be prioritized and what features might need to be dropped in order to meet the deadline.

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

Plan: Keep a dedicated timeline with long term, and short term objectives.

Risk: Major changes to design are needed during implementation.

Plan: Set feature list defined by product owner allows for the assumption that major feature changes will not be an issue.

Risk: Addition or loss of team member (i.e. someone dropped the course, a new person joins the team).

Plan: Have the whole team aware of all parts in case of dropped classes, so portioned workload can be re-divided and absorbed by the group. If someone joins the team have a meeting or a few meetings where we explain our project. Re-distribute tasks among the team members to get them an adequate amount of work.

Risk: Unproductive team member(s)

Plan: Communicate among team members and motivate each other. Find out why the member is unproductive. Find a way to re-distribute tasks so that everyone stays productive.

Risk: Team member(s) lacking expected background
Plan: Each team member will be assigned to tasks that cater to his strengths.

Risk: Illness or unanticipated life events (e.g. death of family member)

Plan: In the event of serious illness or unanticipated life events team something will assess whether the project can still be completed unscheduled and if not we will appeal to the reasonableness of those incharge of such deadlines.

Risk: Inexperience with new tools

Plan: Ask for help from someone with more experience with the tools. Work together to find needed information about the tools and find examples to work off of to try and gain a deeper understanding of the tools.

Risk: Learning curve for tools steeper than expected

Plan: Keep a detailed log of helpful explanatory videos for use as reference material.

Risk: Tools don’t work together in an integrated way

Plan: Keep contingencies for streamlining development incase of incompatibility.

Project Schedule

An initial project plan showing which features will be completed in which sprint. A table, set of tables, Or bullet lists are acceptable.

Sprint 1 Plan:
- Project Proposal

Sprint 2 Plan:
- Issue(s): create, get, list all, update, delete, set title, update title, set status issue

Sprint 3 Plan:
- issue(s) - assign
- Comment(s): Add to an issue, get a comment from an issue, get all of an issue's comments, update a comment of an issue
- User(s): Create / Add, List all, Remove a user

Sprint 4 Plan:
- Find using a single attribute
- Optional javascript interface
- Optional find using two attributes
- Optional find using any number of attributes

Appendix A:

Use Case	REST endpoint	JSON Response
Create Issue	**/IssueService/Create? Identification number=<string>, title=<string>, Description=<string>, Status=<string>, Assigned=<string>, Comment=<string>**	**{"result": "New Issue Created"}**
Get Issue	**/IssueService/GetIssue?IssueID= <num>**	**{"result": "<string>"}**
List All Issues	**/IssueService/getAllIssue?**	**{"result": "<string>"}**
Assign Issue	**/IssueService/AssignIssue?IssueI D=<Num>,UserID=<Num>**	**{"result": "Issue Assigned"}**
Update Issue	**/IssueService/UpdateIssue?IssueI**	**{"result": "Issue Updated"}**

	D=<Num>	
Delete Issue	**/IssueService/DeleteIssue?IssueID=<Num>**	**{“result”: “Issue Deleted”}**
Set title for an Issue	**/IssueService/SetTitle?IssueID=<Num>,Title=<String>**	**{“result”: “Issue Title Set”}**
Update an Issues Title	**/IssueService/UpdateTitle?IssueID=<Num>,Title=<String>**	**{“result”: “Tile Updated”}**
Set Status of an Issue	**/IssueService/SetStatus?IssueID=<Num>,Status=<String>**	**{“result”: “Issue Status Set”}**
Update status of an Issue	**/IssueService/UpdateStatus?IssueID=<Num>,Status=<String>**	**{“result”: “Issue Status Updated”}**
Add to an issue	**/IssueService/AddToIssue?IssueID=<Num>**	**{“result”: “Added to issue”}**
Get a comment from an issue	**/IssueService/GetComment?IssueID=<ID>,CommentNum=<Num>**	**{“result”: “<string>”}**
Get all of an issues’s comments	**/IssueService/AllComments?IssueID=<Num>**	**{“result”: “<string>”}**
Update a comment of an issue	**/IssueService/UpdateComment?IssueID=<Num>,CommentID=<Num>,Comment=<String>**	**{“result”: “Comment Updated”}**
Delete a comment from an issue	**/IssueService/DeleteComment?IssueID=<Num>,CommentID=<Num>**	**{“result”: “Comment Deleted”}**
Create / Add user	**/IssueService/AddUser?<UsersName>**	**{“result”: “New User Added”}**
List All Users	**/IssueService/AllUsers?**	**{“result”: “New Issue Created”}**
Remove a user	**/IssueService/RemoveUser?UserID=<Num>**	**{“result”: “User Deleted”}**