

CPSC 3720 SPRING 2020

White Water Reporting W.W.R.

Euphrates



Michael Wynnchuk, Cole Anderson, Christian Walker

October 16th, 2020

TABLE OF CONTENTS

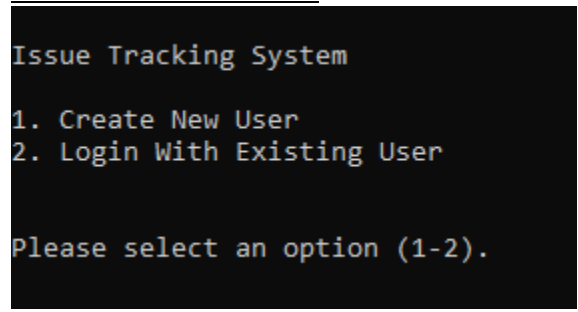
Introduction	3
Proposal	3
Prototype Images.....	3
Who uses the issue tracking software?	5
What does the user want to do with the software?	5
What is the user's goal?.....	6
What steps does the user take to accomplish a particular task?	6
How the website should respond to an action?	7
Project Management	8
Team Organization.....	8
Risk Management	8
Requirements/Design/Estimation	8
People	9
Learning & Tools	10
Project Schedule	10
Software Design:.....	11
Design and Design Rationale	11

INTRODUCTION

You wouldn't want to be stranded on a river with a tear in your raft without a patch, would you? The same goes for software without a patch either. Look no further than White Water Reporting, an issue tracking system that is guaranteed to keep your software afloat. Michael, Christian and Cole are here to serve you with special features such as and ability to search based on attributes, linking your issue to your operating system and the ability to collaborate with your fellow rafters and vote on the importance of an issue. The following outlines how we are going to make sure you won't be stranded without a paddle the next time you have an issue on the river of life. Our team is ready to keep your progress flowing regardless of the dam's you may encounter. Below you will find prototypes, feature set, how we will manage the team and design of the project at hand.

PROPOSAL

PROTOTYPE IMAGES

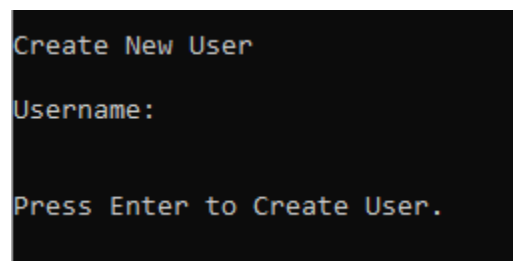


```
Issue Tracking System

1. Create New User
2. Login With Existing User

Please select an option (1-2).
```

(Login Page)

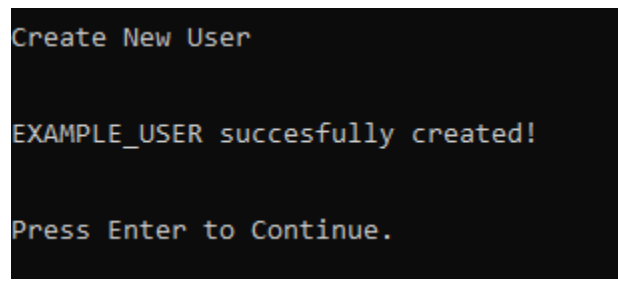


```
Create New User

Username:

Press Enter to Create User.
```

(Signup Page)



```
Create New User

EXAMPLE_USER succesfully created!

Press Enter to Continue.
```

(Successful Signup)

```
Create New User
```

```
That username is already taken. Please try another.
```

```
Username:
```

```
Press Enter to Create User.
```

(Unsuccessful Signup)

```
Issue Tracking System - Home
```

```
Issues:
```

1. Create New Issue
2. View Issue by ID
3. Search Issues by Attribute (OS, Issue Type, Title, Status)
4. View All Issues
5. Update an Issue
6. Delete an Issue
7. Update Issue Title
8. Update Issue Status

```
Users:
```

1. List All Users
2. Delete Account

(Home Page)

```
Issue Tracker System - Issue ID: 1

Title: [Issue Title]
Status: [Issue Status]
Assigned to User ID: [Assignment ID]

Description: [Issue Description]
.....
.....
.....
.....
Comments:

[Issue Comment]: [User ID]
[Issue Comment]: [User ID]
[Issue Comment]: [User ID]

1. Add comment

2. Delete comment
3. Update Issue
4. Return to Home

What would you like to do?
```

(Issue Page)

WHO USES THE ISSUE TRACKING SOFTWARE?

The client uses the issue tracking software. There is only one type of user for this software project so there are no administrators. Therefore, every user created in this software will have the powers of an administrator like being able to delete a created account (be it their own account or other users), delete an issue from other users, etc. Finally, there will be no passwords when the user creates an account, in order to login the user must only input a valid username (which will be case sensitive) to login successfully.

WHAT DOES THE USER WANT TO DO WITH THE SOFTWARE?

The user can do the following with the software:

- Create a user account (providing only a username as a means of logging in)
- Login to created account
- Delete an account
- Create an issue

- Update an issue
- Delete an issue
- Comment on an issue
- Search for an issue (optional feature)
- Operating system attribute (optional feature)
- Attribute/description on how the issue affects the software (optional feature)
- Priority of issue (optional feature)
- Be able to view the options menu and go back to the options menu once their done with using the option they have selected before to select a different command or exit the software
- Update issue title
- Update issue status

WHAT IS THE USER'S GOAL?

Since this software is an issue tracker the user's goal or purpose for using this software will be to create, update, and delete, an issue as its core focus. There are various attributes that issues must have such as its title and status. The optional features that have been implemented to make this issue tracker software unique is the search command for finding an issue that has already been created, being able to set and view the priority of the issue, view and add the operating system attribute.

WHAT STEPS DOES THE USER TAKE TO ACCOMPLISH A PARTICULAR TASK?

Firstly, for the user to use the issue tracking software they must first have an account already setup or create one. Afterwards the user may do any one of the following from the options menu:

- Be able to view the options menu and go back to the options menu once their done with using the option they have selected before to select a different command or exit the software
- Create an issue
- Update an issue
- Delete an issue
- Comment on issue
- Search for an issue (optional feature)

- Operating system attribute (optional feature)
- Attribute/description on how the issue affects the software (optional feature)
- Priority of issue (optional feature)
- Update issue title
- Update issue status

After the user is done with their selected option the user may back out of the program to options menu again or they may close the program.

HOW THE WEBSITE SHOULD RESPOND TO AN ACTION?

Once the application is launched the user will be met with the login screen where the user can either login with a pre-existing account or create an account. The program will then determine if login was successful. If the login is unsuccessful such as no valid user account was selected. However, if the login is successful then the user can proceed to the options menu where they may do any of the following in any order:

- Be able to view the options menu and go back to the options menu once their done with using the option they have selected before to select a different command or exit the software
- Create an issue
- Update an issue
- Delete an issue
- Comment on issue
- Search for an issue (optional feature)
- Operating system attribute (optional feature)
- Attribute/description on how the issue affects the software (optional feature)
- Priority of issue (optional feature)
- Update issue title
- Update issue status

After the user is done with their selected feature in the software, they may back out to the options menu again to select another feature of the software or they

may choose to exit the software. Whenever a change is made be it creating, deleting or in general modifying an issue or an issue attribute a notification message will display mentioning what change has been made.

PROJECT MANAGEMENT

TEAM ORGANIZATION

Roles:

- a. Team Lead – Cole Anderson
- b. Documentation/Scrum Master – Christian Walker
- c. Quality Assurance – Michael Wynnychuk

RISK MANAGEMENT

REQUIREMENTS/DESIGN/ESTIMATION

1. The team planned a project that is too large (i.e. “eyes bigger than stomach”).
 - a. Focus on back-end and main attributes of the project (i.e. the controller and basic features) before moving on to extra features. Keep UI as simple as possible while building up the functionality of the project.
2. The team underestimated how long parts of the project would take.
 - a. Discuss rescheduling project components with team. Prioritize base attributes and put focus on which components need to be completed first. Drop unnecessary features if there is no time.
3. Major changes to design are needed during implementation.
 - a. Consult team before making any major changes as this can be very time consuming and can cause miscommunication between team members. Discuss whether major designs are

worth changing in the scope of the project. Allocate timeframe in order to make design changes within a separate branch of the project repo.

PEOPLE

1. Addition or loss of team member (i.e. someone dropped the course, a new person joins the team)
 - a. Design project structure/team roles to separate tasks between teammates while documenting changes made for each task. This way, if one member leaves, their work documentation will help others to pick up where they left off. This also works for new members joining the team as they can pick up a task from another member and view the documentation for that task to develop an understanding of what needs to be done.
2. Unproductive team member(s)
 - a. Discuss unproductive member(s) with remaining team members and come to a consensus on how to handle the issue and if the issue is affecting the team as a whole or is just a single person problem. Team members will discuss the issue with the accused in a non-confrontational manner in order to reach a mutually beneficial solution for all members (i.e. discussing whether outside class workload is too heavy for a certain member to complete their task(s) in a timely manner).
3. Team member(s) lacking expected background
 - a. Members will be expected to bring up any questions pertaining to syntax, design patterns or anything else within the team channel. Team members will discuss solutions and if none are found, the issue will be escalated to the appropriate class channels on MS Teams for the instructor and/or T.A. to assist with.
4. Illness or unanticipated life events (e.g. death of family member)
 - a. Team members will discuss the severity of the issue. If it is an emergency and the member is unable to finish their task(s) or cannot contribute to finishing the project at all, the

instructor will be informed of the situation to discuss further action. If the issue is an illness and the member cannot finish their task(s) to completion or can only handle a portion of the previous workload, team members will coordinate handling the rest of the workload while the sick team member only handles a small portion.

LEARNING & TOOLS

1. Inexperience with new tools

- a. As outlined in II.3.a., members will be expected to bring up any technical issues or questions within the team channel first, then potentially escalate the issue to the appropriate class channel(s). Experienced team members may also discuss new and helpful tools with other members in order to optimize workflow.

PROJECT SCHEDULE

Sprint	Deadline	Feature
Sprint 1 (Project Proposal)	October 16 th @ 11:59PM	N/A
Sprint 2	October 30 th @ 11:59PM	<ul style="list-style-type: none"> • Create User • Create Issue <ul style="list-style-type: none"> ○ Assign Issue to User ○ Set Issue Title ○ Set Issue Status • Get Issue • Update Issue <ul style="list-style-type: none"> ○ Update Title ○ Update Status • Delete Issue
Sprint 3	November 20 th @ 11:59PM	<ul style="list-style-type: none"> • List All Issues • Remove User • List All Users • Search Users by Attribute • Rate Issues by Vote
Sprint 4	December 9 th @ 11:59PM	<ul style="list-style-type: none"> • Add Comment • Delete Comment • Get Comment • Get All Comments

SOFTWARE DESIGN:

DESIGN AND DESIGN RATIONALE

Use Case	Method	REST Endpoint	JSON Response
Create Issue	CREATE	/issues/create	{"issue": "id"}
Get Issue	READ	/issues/get/{issue}	{"issue": "id"}
List All Issues	READ	/issues/listAll	{"issues": "ids"}
Assign Issue	CREATE	/issues/assign/{issue}	{"issue": "userId"}
Update Issue	UPDATE	/issues/update/{issue}	{"issue": "status"}
Delete Issue	DELETE	/issues/delete/{issue}	{"issue": "id"}
Set Title	CREATE	/issues/setTitle/{issue}	{"issue": "title"}
Update Title	UPDATE	/issues/updateTitle/{issue}	{"issue": "title"}
Set Status	CREATE	/issues/setStatus/{issue}	{"issue": "status"}
Update Status	UPDATE	/issues/updateStatus/{issue}	{"issue": "status"}
Add Comment to An Issue	CREATE	/issues/comments/add/{issue}	{"issue": "comment"}
Get Comment from Issue	READ	/issues/comments/get/{issue}/{comment}	{"issue": "comment"}
Get All Comments from Issue	READ	/issues/comments/getAll/{issue}	{"issues": "comments"}
Delete Comment	DELETE	/issues/comments/delete/{issue}/{comment}	{"issue": "comment"}
Create/Add Users	CREATE	/users/create	{"user": "id"}
List All Users	READ	/users/listAll	{"users": "userIds"}
Remove User	DELETE	/users/remove/{user}	{"user": "id"}
Search Issues by Single Attribute	READ	/issues/search/{attribute}	{"issue": "attribute"}
Specify OS of Issue	CREATE	/issues/os/{issue}	{"issue": "os"}
Specify Type of Issue	CREATE	/issues/type/{issue}	{"issue": "type"}
Vote on Issue Priority	UPDATE	/issues/vote/{issue}	{"issue": "points"}

Where possible we (as a team) will try to use design patterns in our project to help reduce software complexity. One of those ways to reduce software complexity may be to use the MVC model for this project since the project can be

split up into the view, model and controller. When it comes to creating the various classes that will be used to create the controller, more design patterns may be necessary to ensure that the software design is consistent throughout the project development. It is difficult to say what exact design patterns will be used for this project since that would involve drafting an extensive UML diagram drafting initially and then would later need to be revised several times as needed throughout the project development. However, since the MVC model is flexible in the sense that it can accommodate various design patterns which is the reason why it is ideal to use as the base of the project. Furthermore, the model part of the MVC model may also use design patterns as well to ensure the software remains as flexible as possible to new additions of code and thereby reducing overall development time. Since the view class will be the simplest part of the project to implement the rationale to use a design pattern for that part of the project may not be warranted/required.