106 Integrated Studio II - Assessment 2

Jayden Everest, Conor Cook, Kev Karati

Yoobee Colleges

25/11/2022

Table Of Contents

Introduction

Goal

The purpose of this project is to develop a collaborative software defect tracking application that is simple, well designed and useful to its end users. To accomplish this, we have undertaken extensive research that culminated in the development and testing of a Hi-Fi Prototype for our application.

This was followed by the development of the full defect tracking program which included reviewing our previous findings and black box testing to ensure that our application is usable, useful and our original purpose has been met.

Target Audience

Since the purpose of the application is to track software defects, it'll be primarily used by developers working on team projects, to determine what they need to work on. It'll also be used by testers, when they find program defects. Lastly, it'll be used by administrative departments, to assign work to different employees and to determine what issues need to be prioritized.

Assumptions

Before doing research into what's needed in the app, we created a list of features (Below), that we assumed would be needed in the final product.

- Users will need to be able to login, and there will need to be different access levels for different account types (Admin / Dev / Tester accounts).
- Administrators should be able to create projects and rename them.
- Admins should be able to add / remove accounts and assign them to collective groups / departments and to individual tasks.
- All users should be able to add / edit defects to the system.
- Admins / Devs should be able to add / edit defect priorities.
- All users should be able to create logs for defects, but only admins should be able to change a defects status.
- Admins should be able to assign tasks to groups / departments.
- User accounts will need to be customizable (With profile pictures, banners, a username, and so on)

Software Process Models

Considering that this project will take place over two months, and will be worked on by 3 different group members, it's imperative that we choose a software process model, in order to manage the project workflow. However, since there is a large variety of models, we need to evaluate each of them in order to determine which model best fits this project.

First, we looked at the waterfall model, where the different project phases (Such as research, design, and development) are completed in order, and the current phase must be completed before moving onto the next phase. The main benefit to this approach is its rigid structure, which makes it easier to meet deadlines and track progress. However, this rigidity also means unexpected issues can cause much larger problems, and can delay the entire project, while segments that are completed quicker than expected just lead to a pause in productivity, instead of increasing the overall project speed.

Secondly, we looked at the agile model, which focuses on continuous iterative development. During each iteration, a small part of the project is developed, typically taking place over several weeks. The main benefit to this approach is its flexibility and adaptability, especially with changing project requirements. This is especially useful when the program requirements are unclear initially, such as in game development. However, since requirements are constantly being added and removed, this makes it far harder to track progress, and deadlines are much harder to set. It also creates more legacy code, since systems are constantly added and removed.

Thirdly, we looked at the XP model, a derivative of the agile model at focus on code quality and communication. It achieves this through pair programming, where developers are paired up and work at the same computer, in theory making problems much faster to solve. However, this requires much more effort, and lowers overall productivity. It also limits code typing speed, since only one developer can type at any one time.

Fourthly, we looked at the Kanban model, which uses team task boards to distribute tasks to all team members. Each task is converted into a card, containing information about the task, and some priority/categorization information. Then, individual members can either be assigned, or assign themselves, to different tasks. This makes tracking progress much easier, since all tasks and backlogs are clearly visible. It also prevents members from accidentally working on the same task, since everyone can see what others are working on. However, the main drawbacks with this model is the time it takes to separate each task into individual cards, and the task boards can become cluttered quickly.

Lastly, we looked at the RAD model, which focuses on code reuse and rapid development, mainly through the use of pre-existing code, both internal and open-source. This speeds up development drastically, since much of the project's code only needs to be adapted to fit the scenario, instead of having to be written from scratch. However, since the code isn’t being specially designed for the project, it’ll be less efficient, and may be outdated. It also only works when there is a large backlog of code that fits the project, so the more innovative the project, the harder it is to utilise the RAD model.

After evaluating several different models, we’ve decided to go with a waterfall-agile-kanban-RAD combination. Since the deadline cannot be moved, we need to ensure that we meet it. As such, we’ll lay out the overall project using the waterfall model, dedicating a given amount of time to each core element of the project (Such as documentation, or the program).

However, we’ll use an agile-kanban hybrid for each waterfall phase, focusing on making the highest quality result before the waterfall deadline forces us to move to the next

phase. We chose the agile model because of its efficiency, since if a task is completed quicker than expected, we can just complete more tasks than we initially anticipated. This results in more tasks being completed, and a higher quality result. In turn, we chose to use the Kanban style of task assignment, creating a large board of tasks, and self-assigning tasks as needed. We did this since our group size makes keeping track of everyone's progress easy, even if we are each focusing on different things. If the group was larger, we would have likely needed to get someone to assign tasks to everyone instead, to ensure that everyone was equally productive.

Finally, we chose to use the RAD model for certain aspects of the program, namely the login & password hashing system. Previously, we had developed a teaching administration system that included an in-depth login system, so to speed up development, we'll adapt that login system to fit this project, instead of starting from scratch.

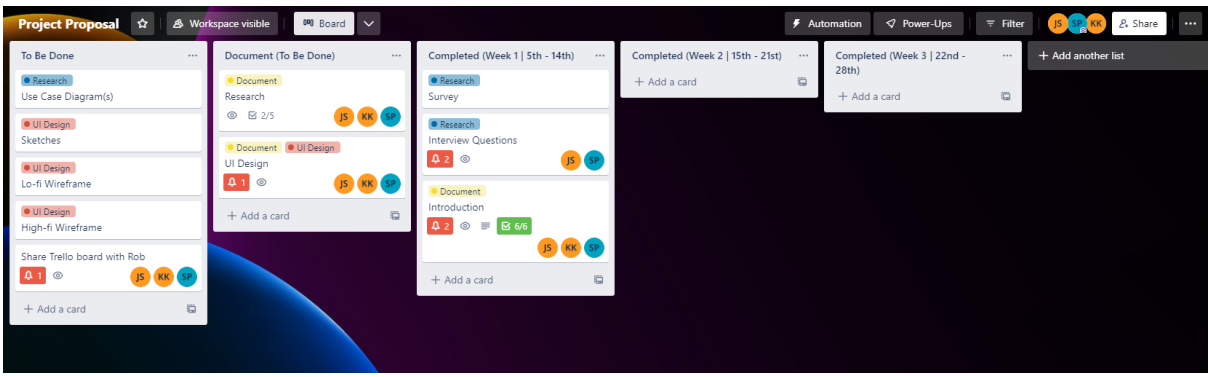
Timeline

This project will take place over 8 weeks, with the planning stages complete by week 4. Specifically for the planning stages, we'll plan out all our research during week 1, conduct and analyse the research during week 2, and create a navigable prototype during weeks 3 & 4.

To ensure that this timeline is kept, we'll distribute responsibilities evenly between the 3 project members, by deciding what we should each work on through weekly group meetings. We'll also use collaborative software to assign tasks to each of us, so that we can keep track of every group member's progress (See below screenshot).

Fig. 1

Trello (2022) Kanban board



Tools

To streamline the design and development process, we've decided to use a variety of digital tools to aid us. Firstly, we'll use Trello for project management, especially for the Kanban aspects of our project, like task distribution. We chose Trello (2022) primarily for its accessibility, thanks to its mobile and desktop compatible layout, making it easy for us to check tasks anywhere. We'll also use Figma (2022) for developing the Hi-fi prototype of the app, due to our familiarity with it, since its industry standard, and for its ease of use. Finally,

we'll use wireframe.cc (2022) for our Lo-fis, since its minimalist design will force us to keep the designs simplistic, and paper for the sketches, to keep them rough.

For the development portion of the project, we'll be using Github Desktop (2022) for version control, so that if a file is accidentally deleted, we won't lose access to it, and to make code collaboration much easier, thanks to its cross-computer syncing. We'll also use C++ as our programming language of choice, due to its speed and optimization, and the QT framework (2022) as our GUI system, since we were the most familiar with it. Lastly, we'll use the QT creator IDE (2022) for developing the GUI parts of our program, because of its drag and drop interface, and Visual Studio Code (2022) for the code-heavy parts, because of its auto-completion tech and aesthetic.

Constraints

The largest constraint on this project is the GUI development tool we've been tasked with using, being QT (2022). Due to its simplistic and mostly code based development environment and tools, we're limited in how visually complex we can make the program, in regard to elements such as drop shadows, and animations.

What this means in practice is that we need to consider how difficult visual features will be to actually implement, as a feature easy to implement in the Hi-fi may be vastly more difficult to implement in code.

Research

Real-world interviews

To identify what features are broadly expected in our application, we decided to interview several developers with industry experience, focusing on their past experience with issue tracking software. Before interviewing, we first wrote up 4 open-ended questions to prompt the interviewee, listed below.

Q1: When a tester notices a bug in your code, how do you typically react?

Q1-1: If you document it, how and what do you document?

Q1-2: If you don't, what do you do?

Q2: When working on a team project, what features do you expect from issue tracking software?

Q3: When debugging a personal project, how would you choose which issues to focus on?

Q3-1: Following that, if you have worked for a company, how were tasks/issues distributed?

Q4: What parts of an account profile do you think are important when using collaborative tools?

After this, we then interviewed 4 industry developers, noting down everything they said (See below). We also collated any features/needs/use-cases they mentioned into a table, so that we could confirm that our end-result catered to the majority of them (Considering our time constraint, we'll focus primarily on the most important features).

Participant A (Test Analyst, Ex-Programmer)

When a tester notices a bug in your code, how do you typically react?

When a tester discovers a bug in code the tester raises a bug task on the board or against the job they were testing. The task documents what the bug is, the steps they did to identify the bug. It also states what result they were expecting to see. The programmer then uses this information to recreate and rectify the issue.

If you document it, how and what do you document?

When I was programming, and was fixing a bug, it was policy to record any changes in the program log (at the top of the program code) noting date, task number and description of the change.

When working on a team project, what features do you expect from issue tracking software?

We use kanban/scrum board with swim lanes. With the lanes you are able to visually see where a task is at eg: in development, code review, testing, release mode etc.

When debugging a personal project, how would you choose which issues to focus on?

Tasks are usually put into priority order. This is determined by the product owner or team discussion.

Following that, if you have worked for a company, how were tasks/issues distributed?

I work in a team which is led by a product owner (PO). The PO usually knows which tasks are the priority and therefore order the job tasks in priority order on our scrum board. When people become available they normally pick up the next task on the board unless it is decided that a particular person is better suited to it.

What parts of an account profile do you think are important when using collaborative tools?

Who raised the task, when raised, who is currently working on it, current status (eg: not started, in development, testing etc), the estimated size of the task.

Participant B (Team Leader)

When a tester notices a bug in your code, how do you typically react?

I will talk to the developer to see if they already know about it, and check the requirements.

If you document it, how and what do you document?

Log the ticket and record the expected result, actual result, requirement, environment, screen shots

When working on a team project, what features do you expect from issue tracking software?

- User details
- Status
- Who it's Allocated to
- Environment (dev/test/UAT etc)
- Priority
- Affected areas/code
- Ability to attach documents
- Ability to add comments
- Add others so they can watch
- Type of issue

When debugging a personal project, how would you choose which issues to focus on?

- The ones which stop the project from functioning
- Ones that corrupt the data
- Once project is functioning the ones that would impact the user the most
- Then additional functionality

Following that, if you have worked for a company, how were tasks/issues distributed?

By priority.

What parts of an account profile do you think are important when using collaborative tools?

My name, date/time of action, and the application/project I'm currently working on.

Participant C (Senior Analyst Programer)

When a tester notices a bug in your code, how do you typically react?

The first thing I would normally do is perform an initial assessment of it and document my thoughts alongside the description of the defect.

If you document it, how and what do you document?

If the tester is sitting near me I might tell this to them verbally, but it would/should always be written down. This assessment is done quite quickly and would indicate: How much work would be involved in fixing it, its complexity, its impact on other parts of the system, and a simple rating system assigned to it (Small, Medium, Large, XLarge, etc).

When working on a team project, what features do you expect from issue tracking software?

Some things I would expect:

- Description of defect.
- Assessment notes = A history of who may have an opinion of how to fix it as various members of the team or other staff in other teams may have some input.
- Assignee - Who is it currently assigned to (or empty if unassigned).
- State - eg: Awaiting to be assigned; In progress; Fixed; Resolved; etc System Version - What version is/has the fix been incorporated in.
- Audit history of what was updated and who updated the issue in the tracking application.

When debugging a personal project, how would you choose which issues to focus on?

I would assess tasks/issues using the following questions:

1. What is the most interesting or fun to do?
2. What is the quickest or easiest thing to do that will make the biggest impact?
3. What needs to be fixed first as other tasks/issues are dependent on it working.

Following that, if you have worked for a company, how were tasks/issues distributed?

Generally, a senior staff member assigns tasks/issues to their staff members. The senior staff member may evaluate who to assign it to by assessing: Availability of their staff members to do the work (E.G: Who has free

- Availability of their staff members to do the work, ie: who has free time.
- Who has the most expertise in the area where the defect occured in the system.
- The priority of the defect to be fixed.

What parts of an account profile do you think are important when using collaborative tools?

- Name
- Job title (or responsibilities)
- Areas of expertise in the system
- Location and/or time zone of person (if they work in another country)
- Contact details

Participant D (CEO):

When a tester notices a bug in your code, how do you typically react?

We don't actually use testers in a traditional way, our code is tested automatically in parts via a process called unit testing, in which we test the code with hundreds of thousands of iterations and test for known edge cases. This occurs progressively during development, and each *dev is responsible for testing their own components.

**Dev(s): Developers*

If you don’t typically document bugs, what do you do?

These bugs are often not documented, as they're fixed near instantly, but for bugs that stick around longer, require cooperation from various people, or are reported by users once we're in production, we file issues on our git repos. Information recorded here, is typically who encountered the error, what they were doing, the approximate time and date when the bug occured, as well as stack trace information and any relevant log files from the instance where the error occurred.

When working on a team project, what features do you expect from issue tracking software?

We expect to be able to tag issues with severity, assign people to be responsible for overseeing the issue, carry out conversations, and point to relevant code sections

When debugging a personal project, how would you choose which issues to focus on?

Our priorities depend typically on a very straightforward order of operations, both inside and outside of the workspace.

Following that, if you have worked for a company, how were tasks/issues distributed?

Typically issues are self assigned or assigned by a team lead, in this priority order

What parts of an account profile do you think are important when using collaborative tools?

None, really. As long as a dev is identified by name, and can be assigned to issues, we don't really care.

Summarised Answers

Q1	Q1-X	Q2	Q3	Q3-1	Q4
Testers create reports	Record any changes in log, including date and description	Progress indicator	Priority Order	Admins assign priority	Project working on
Record bug description	Create report (Called a “Ticket”)	Estimated task difficulty / complexity	By how much it contributes to program stability	People self-assign tasks	User status
Steps to reproduce	Bug environment	Account details	Impact / Feature dependencies	Priority	Name
Expected result	Screenshots	List of allocated users	Most interesting / fun	Senior staff distribute tasks	Expertise / Job Title
Confirm no similar reports exist	Name of report creator	Forward report to users	Impact - Time cost ratio	Staff availability	Location / Time zone
Check project requirements	Overall system impact	Type of issue	Order of operations		Contact details
Document thoughts	Rating system	Affected areas			
	File issues in git repos	Attach documents			
	Date / Time bug found	Descriptions of issues			
	Log files	Notes left by users			

Stack trace information	Version issue found in	Audit history	Severity

General Interview Conclusions

First, we found that participants expect all users to be able to create reports, but only admins/senior staff to be able to assign report priorities. What this tells us is that users are largely unrestricted in what they can do in a software defect program, and that admins are more for organisation than security. This is likely because software defect programs are mostly used internally in a company, so everyone using the program is a registered employee, and is accountable for any mis-management of the application.

In terms of report requirements, participants generally expected an issue description, the steps/version to reproduce, date / time found, and the name of the report creator. We also learned that such reports are referred to as “Tickets” in industry, so we may want to include that terminology in our application.

For report meta-data, progress and priority indicators were widely suggested, as well as the ability to assign developers to an issue (And view everyone assigned). However, meta-data changes were limited to admins/senior staff. Notably, senior staff were given the same priority level as admins, so we may want to be able to grant any account admin rights, instead of creating separate admin accounts.

Lastly, participants did not place any importance on account customizability, instead focusing on a list of information for each account that would be useful when assigning them to a particular task, such as their expertise, job, time zone, contact details, status, and name. This means we may need to make an account viewer page, so that users can view the account information of other users, to make educated assignment decisions.

Survey

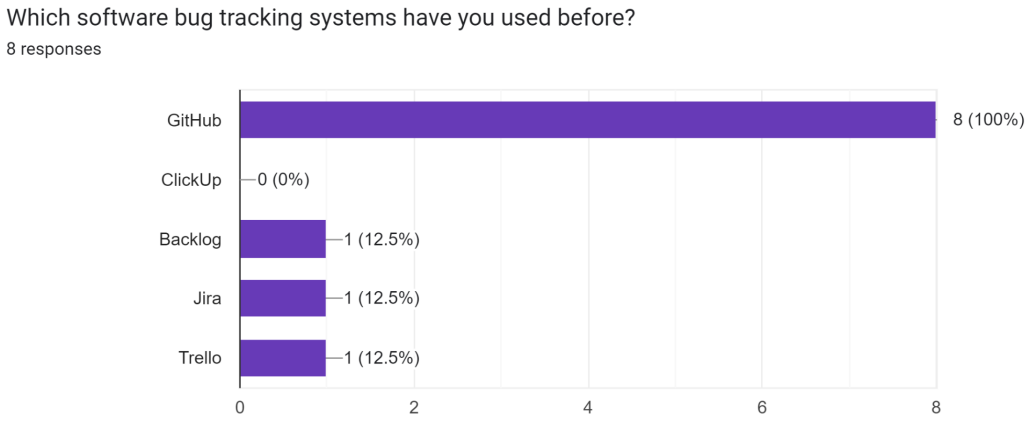
To gather data on how to implement specific features, and to identify some competitor applications we can compare our app to, we created a digital survey, which we then sent out to a large collection of participants that fit our target audience. The benefit of this approach over interviews is data quantity, since it's much faster to send out a short survey than individually interviewing participants. It also gives us more quantitative data, by constraining participant answers to primarily multi-choice questions.

Q1: Which issue/bug tracking software have you used before? (Can select multiple answers)

A. GitHub
B. ClickUp
C. Backlog
D. Jira
E. Other: _____

Fig. 2

Survey question 1 resultant graph



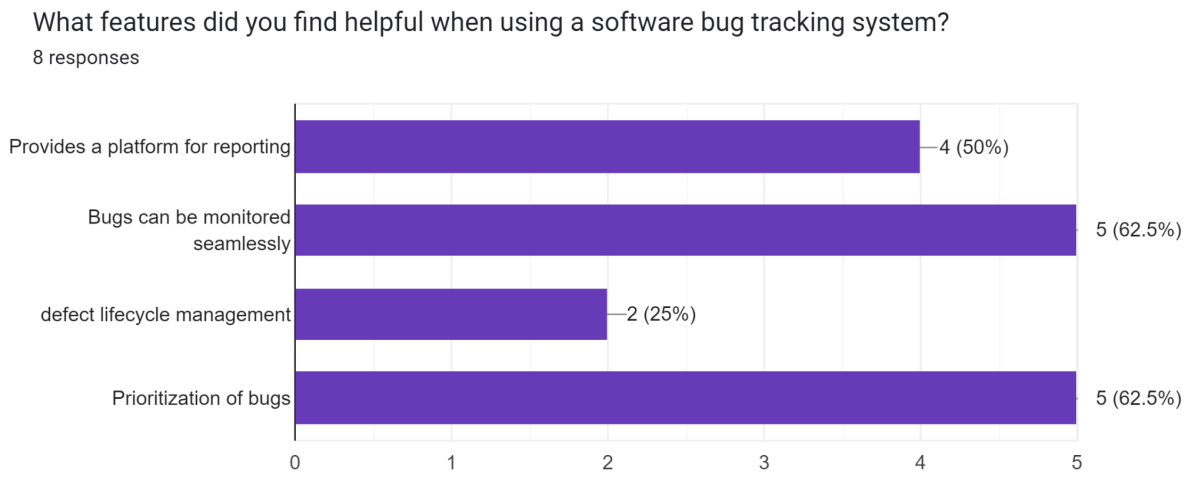
100% of survey participants have used GitHub (2022) as a bug tracking system while Backlog (2022), Jira (2022) and Trello (2022) have been used by one user each. As there is a common system used by all users, the data gathered can more accurately identify what works and what improvements are needed.

Q2: What features did you find useful when using said software? (Can select multiple answers)

A. Bug reporting
B. Bug monitoring
C. Managing/tracking bug progress
D. Bug prioritisation
E. Other: _____

Fig. 3

Survey question 2 resultant graph



All four options provided for this question have been identified as helpful features, with prioritisation of bugs and seamless monitoring of bugs both being voted for by 62.5% of participants. 50% of participants found the reporting platform helpful and 25% found defect lifecycle management helpful.

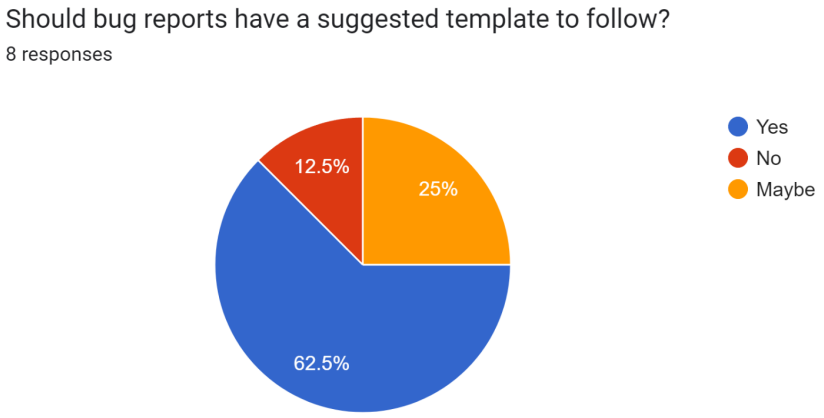
Our bug tracking system will require an easy to use prioritisation system to reflect the result above. The process of reporting bugs by developers/testers will not need to be changed, if reporting remains simple and follows a consistent format it will be a helpful function.

Q3: Should bug reports have a suggested template to follow?

A. Yes
B. No
C. Maybe

Fig. 4

Survey question 3 resultant graph



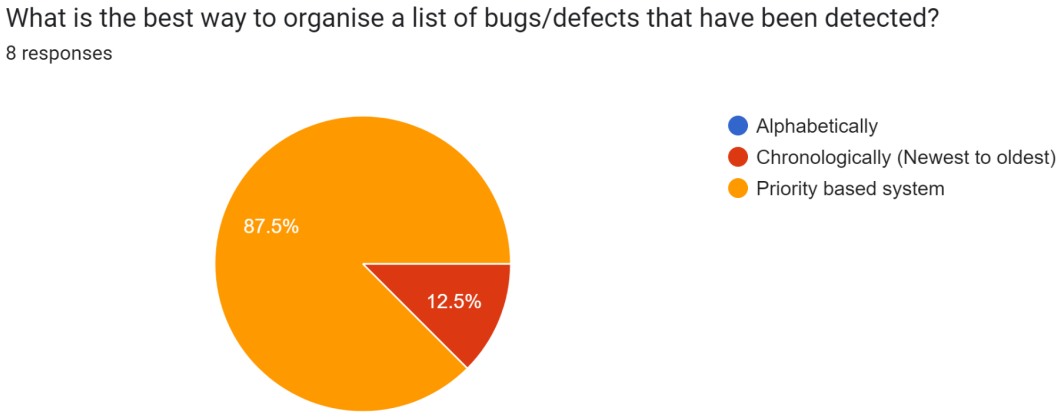
The majority of participants believe bug reports should follow the suggested templates. With two people feeling uncertain about the template. From this we can definitely be certain that having a template for the bug reports will be a good idea.

Q4: What is the best way to sort a list of bugs/defects?

A. Alphabetically
B. Chronologically (Newest to Oldest)
C. Priority
D. Other: _____

Fig. 5

Survey question 4 resultant graph



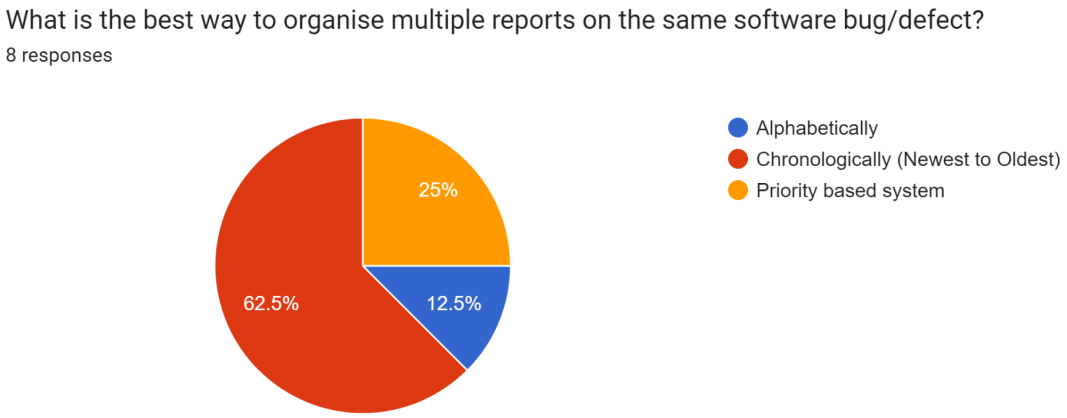
87.5% of participants believe that bug reports should be organised in a priority based system. We assumed that the majority of participants would choose this option as it makes sense for developers/testers to be focused on the most important task they have. A priority based system will allow them to quickly identify urgent bug fixes so they can be solved in the shortest amount of time.

Q5: What is the best way to organise multiple reports on the same software bug/defect?

A. Alphabetically
B. Chronologically (Newest to Oldest)
C. Priority
D. Other: _____

Fig. 6

Survey question 5 resultant graph



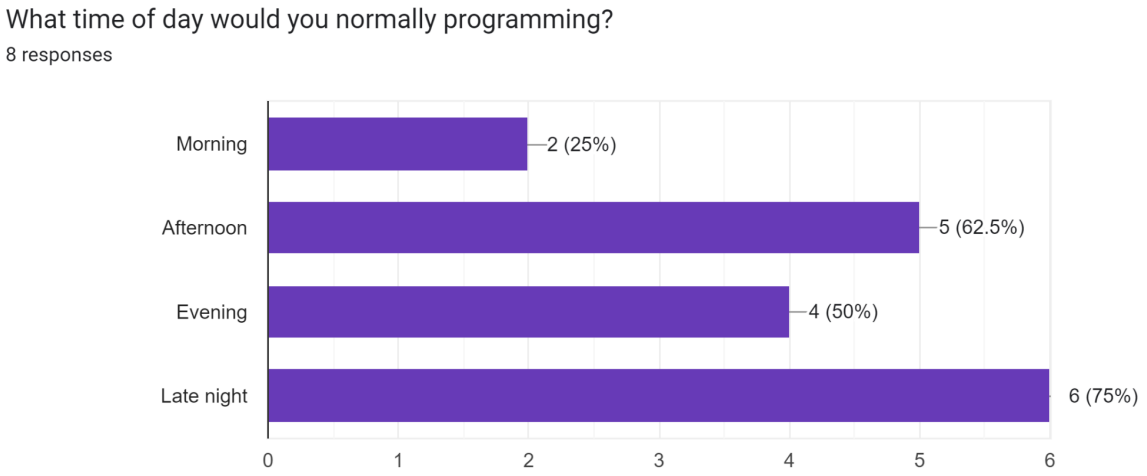
Interestingly, the majority of participants believe reports on the same bug should be ordered chronologically, while the initial bug reports are preferred to be priority based (see Q4). We can infer that developers want to be able to identify the bug with the highest priority and then view the most recent report/log on that particular bug.

Q6: What time of day would you normally program? (Can select multiple answers) (See Ethnography section for analysis)

A. Morning
B. Afternoon
C. Evening
D. Late night

Fig. 7

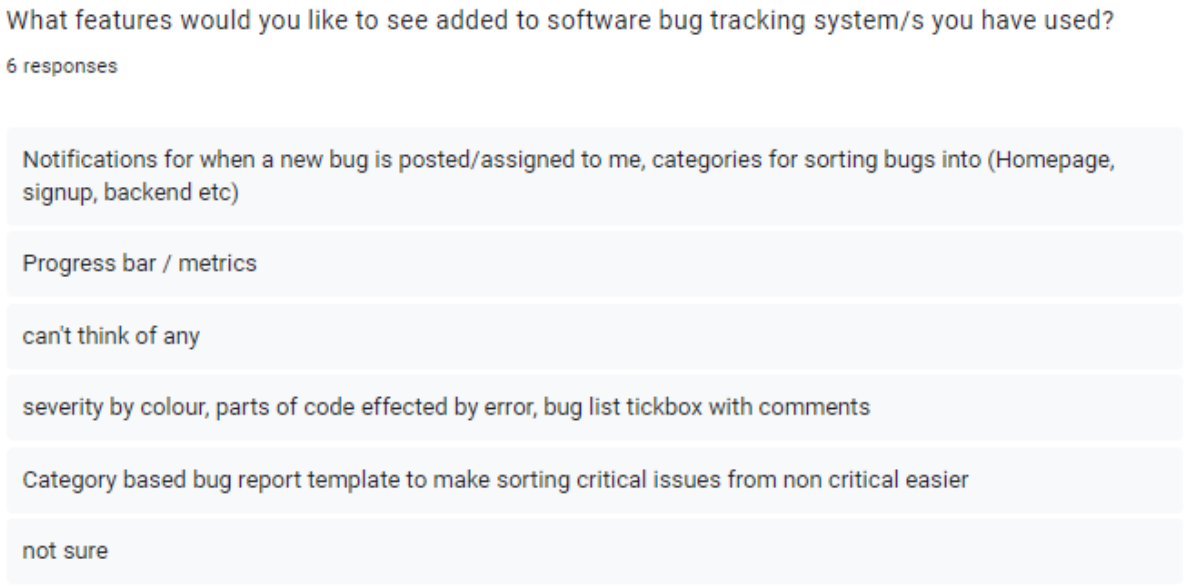
Survey question 6 resultant graph



Q7: What features would you like to see added to issue tracking software you've previously used? (Can input anything)

Fig. 8

Survey question 7 resultant graph



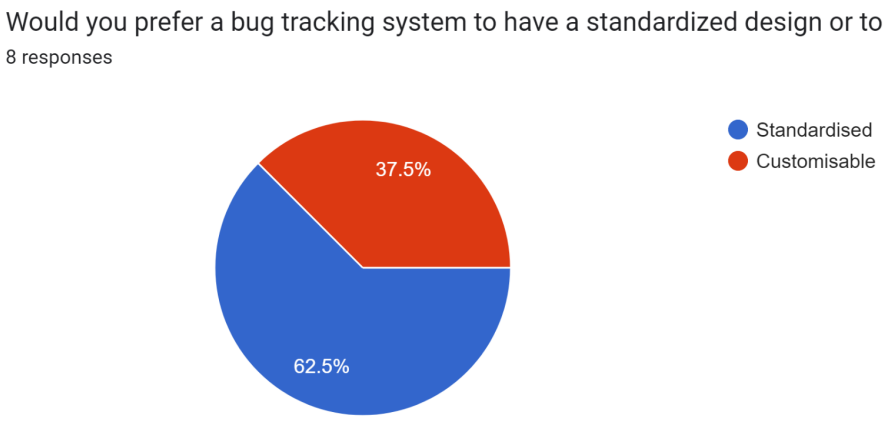
Suggestions from our participants will prove to be vital in ensuring we can deliver the best possible system. Colour coding and categorising our priority system will save developers valuable time. Also, including notifications to alert developers/testers when new bugs or issues are assigned to them should smooth work flow and increase usability and navigation of the system.

Q8: Would you prefer issue tracking software to have a standardised design or to be customizable?

A. Standardised
B. Customizable
C. Other: _____

Fig. 9

Survey question 8 resultant graph



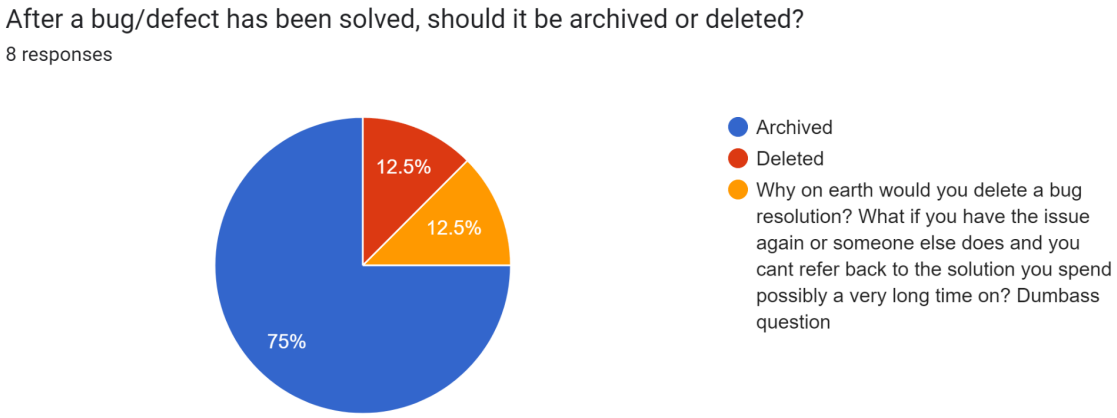
62.5% of participants believe that bug tracking systems are more suited to standardised designs rather than allowing for customisation. Due to the nature of bug tracking systems, the uniformity and consistency of a standardised design will make group projects easier to manage and provide clarity of any issues that arise.

Q9: After an issue has been resolved, should it be archived or deleted?

A. Archived
B. Deleted
C. Other: _____

Fig. 10

Survey question 9 resultant graph



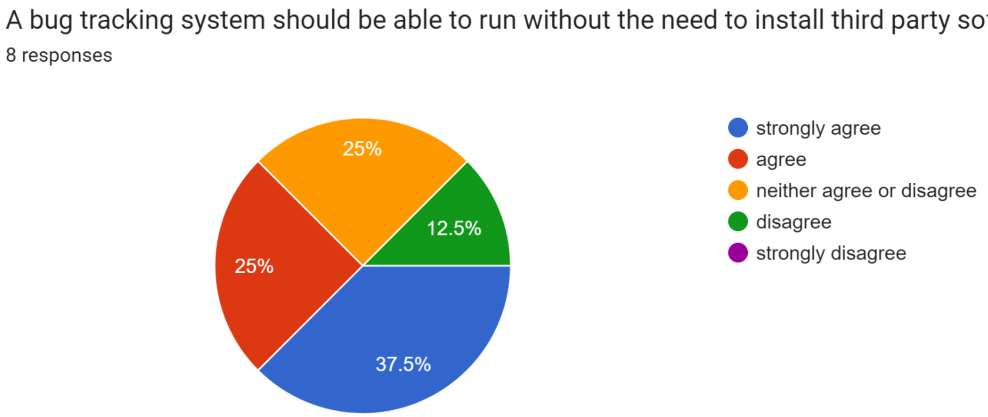
87.5% of participants think bugs that have been solved should be archived instead of deleted. Archiving these fixes will ensure that similar issues in the future can be resolved as quickly as possible. There will be a need for a filter or style of organisation to avoid unnecessary archiving of the same bug/issue to reduce memory usage.

Q10: Bug tracking software should be able to run without the need to install third party software.

A. Strongly agree
B. Agree
C. Neither agree or disagree
D. Disagree
E. Strongly disagree

Fig. 11

Survey question 10 resultant graph



Survey Conclusions

After surveying 8 people, we collated the data gathered into a series of graphs which we could draw conclusions from.

So that our application better fits into its end-user environments, we included a question in the survey about the time/s of day participants tend to program in. If they tend to program in the evening/night, having a darker colour scheme will be important, so that it matches the dimmer environment. Similarly, if they tend to program during the morning/afternoon, have a lighter theme, and if it's a mix, have multiple selectable themes.

After collating the survey data, we found that users tend to program during the afternoon and onwards, with a slight tendency towards late night programming. This means we'll have to design our application to be comfortable to look at both during the day, and during the night.

Requirements

Assumption Revisits

After analysing all the data gathered, we learned that several of our initial assumptions were incorrect.

First, different accounts will need to have different access levels, but an account access level should be separate from the account itself. So an "Admin" would just be a generic account, but with maximum access level, and a "Tester" would have the minimum access level. We decided this since, during one of the interviews, a participant mentioned that senior staff were given admin rights. This means that employees may need varying access levels depending on how long they work, hence their account will need to be adjusted accordingly.

Secondly, all users should be able to assign themselves to different tasks, however admins should still be the only users able to assign others to tasks. We're basing this off participant answers to Q3-1 in the interview, where self-assigned tasks were mentioned frequently.

Thirdly, user customizability isn't needed, with the majority of users preferring a standardized design over a customizable one, as shown in Q8 of the survey. This contradicts our initial assumption that user accounts will need to be highly customizable. However, as shown in the Q4 results of our interview, while account customizability isn't important, being able to check account statuses is. This means users will need to be able to display information about themselves, such as their job title and location. Also, one of our participants mentioned being able to easily recognize users, so making account names clearly visible, as well as potentially a profile picture, will be needed.

User Stories

After collating our data, we created three user stories, based on different participants from our interviews.

User Story A

Greg is a Test Analyst for an MMO game development studio called Team Berry. During his routine testing of the live game, he finds a duplication glitch in the shop menu that allows users to double their greater healing potions. Panicking, he creates a new ticket in the company software tracking system, detailing his findings.

User Story B

Chad is an intern programmer for Nisa. Having finished a trajectory algorithm for the Mars Climate Orbiter much quicker than expected, he needs to find something else to work on for the rest of the week. So, he logs onto her company software tracking system, to find a new task relating to the Orbiter project.

User Story C

Alice is the CEO for Facepage Corp, and has noticed that the productivity of the web development department is down 0.4%. As such, she logs onto the company software tracking system, and decides to assign several dozen new high-priority tasks to the department.

User Story D

Stephan is a team manager for Highland Street Hospital. Stephan had to leave work early for parent teacher interviews. But before he left he made a ticket for a high priority bug in the system. When he came back the next day he wanted to check if anyone was working on the ticket, he checked the log of the ticket and saw four people assigned themselves to the ticket.

User Story E

Grace is a programer for NinThreedo and was working on the latest Betroid game. There was a bug which made the game crash when the player jumps. Grace experienced this same bug when she was previously working on the Betroid. She went back through the archived tickets until she found the resolved ticket.

User Story F

Jesse is a test analyst for BlooperTube, he prioritises giving as much detail as he can about bugs he finds. He does this because previously he found quite a dangerous bug but they were in a rush and did not provide enough data on how to replicate the bug.

Functional & Non-Functional Requirements (As Snowcards)

Requirement #: C1 **Requirement Type:** Non-Functional **Event/BUC/PUC #:** 4

Description: Our program should be quicker to navigate compared to competing programs

Rationale: 100% of surveyees have used GitHub, meaning our program is competing directly with it, also User Story A - Users need the application to be quick to use, especially during emergencies

Originator: Conor Cook

Fit Criterion: Ensure the time taken by participants to open a ticket / profile, and to sign in, is comparable* or better than GitHub. (*Within 5 seconds)

Customer Satisfaction: 3 **Customer Dissatisfaction:** 2 **Priority:** 3 **Conflicts:** 0

Supporting Materials: Document Survey | User Stories

History: Created on 17th October 2022

Requirement #: C2 **Requirement Type:** Functional **Event/BUC/PUC #:** 1

Description: Allow for user freedom when adding information to a new ticket

Rationale: The majority of users like features that allow them to add detail to bug reports

Originator: Conor Cook

Fit Criterion: Once the Hi-Fi has been created, show them the ticket creation system and ask if it feels too constrained. If the majority say no, then the requirement has been met.

Customer Satisfaction: 4 **Customer Dissatisfaction:** 3 **Priority:** 3.5 **Conflicts:** 0

Supporting Materials: Document Survey

History: Created on 17th October 2022

Requirement #: C3 **Requirement Type:** Non-Functional **Event/BUC/PUC #:** 1

Description: Include a suggested template in bug reports / tickets

Rationale: 62.5% of surveyees agree the bug reports should have a suggested template

Originator: Conor Cook

Fit Criterion: Create a new ticket, and if a suggested template appears with description, steps to reproduce, thoughts, and expected result sections, then requirement has been met.

Customer Satisfaction: 3 **Customer Dissatisfaction:** 1 **Priority:** 0.5 **Conflicts:** 0

Supporting Materials: Document Survey | Summarised Answers

History: Created on 17th October 2022

Requirement #: C4 **Requirement Type:** Functional **Event/BUC/PUC #:** 1

Description: Users should be prompted to lay out bug reports / tickets

Rationale: 62.5% of surveyees agree the bug reports should have a suggested template

Originator: Conor Cook

Fit Criterion: If, when creating a new ticket, they are advised to include a description, steps to reproduce, and thoughts section, then requirement has been met.

Customer Satisfaction: 3 **Customer Dissatisfaction:** 1 **Priority:** 0.5 **Conflicts:** 0

Supporting Materials: Document Survey | Summarised Answers

History: Created on 17th October 2022

Requirement #: C5 **Requirement Type:** Non-Functional **Event/BUC/PUC #:** 1

Description: Order reports by priority

Rationale: 87.5% of surveyees agree the bug reports should be ordered on a priority based system, also User Story B - Users need to be able to find tasks easily

Originator: Conor Cook

Fit Criterion: Ensure report lists are ordered by priority, with all highest priority appearing first, and lowest priority appearing last.

Customer Satisfaction: 4 **Customer Dissatisfaction:** 2 **Priority:** 4 **Conflicts:** 0

Supporting Materials: Document Survey | User Stories

History: Created on 17th October 2022

Requirement #: C6 **Requirement Type:** Non-Functional **Event/BUC/PUC #:** 1

Description: Order logs chronologically

Rationale: 62.5% of surveyees agree that logs be ordered chronologically

Originator: Conor Cook

Fit Criterion: Check that the log list in a ticket is ordered from newest to oldest

Customer Satisfaction: 4 **Customer Dissatisfaction:** 3 **Priority:** 4 **Conflicts:** 0

Supporting Materials: Document Survey

History: Created on 18th October 2022

Requirement #: J1 **Requirement Type:** Functional **Event/BUC/PUC #:** 1

Description: Create / remove tickets

Rationale: Q1 & Q1-1 of interview - Participants mention tickets frequently

Originator: Jayden Everest

Fit Criterion: Create ticket in program, then restart program. If the ticket stays, the requirement is met. Repeat process for ticket removal.

Customer Satisfaction: 4 **Customer Dissatisfaction:** 5 **Priority:** 5 **Conflicts:** 0

Supporting Materials: Document Conclusion | Assumptions | Summarized Answers | General Interview Conclusions

History: Created on 17th October 2022

Requirement #: J2 **Requirement Type:** Functional **Event/BUC/PUC #:** 1

Description: View / add logs to an individual ticket

Rationale: Q1-1 & Q2 of interview - User Story D and Participants mention log functionalities

Originator: Jayden Everest

Fit Criterion: Add log to ticket, then view another ticket. If the log doesn't show up in the other ticket, the individual aspect of the log met. Re-open ticket, if log has stayed, requirement met.

Customer Satisfaction: 3 **Customer Dissatisfaction:** 4 **Priority:** 3 **Conflicts:** 0

Supporting Materials: Document Assumptions | Summarized Answers

History: Created on 18th October 2022

Requirement #: J3 **Requirement Type:** Functional **Event/BUC/PUC #:** 1

Description: View / update ticket metadata

Rationale: Q1-1, Q2, Q3, Q3-1 of interview and User Story F - Participants frequently mention several ticket metadatas, like priority, progress, and creation date.

Originator: Jayden Everest

Fit Criterion: Create new ticket, and confirm that the visible ticket creation date matches the correct creation date. Then, change the ticket priority and progress levels, and reload the program. If changes are saved, requirements met.

Customer Satisfaction: 4 **Customer Dissatisfaction:** 2 **Priority:** 2 **Conflicts:** 0

Supporting Materials: Document Assumptions | Summarized Answers | General Interview Conclusions

History: Created on 18th October 2022

Requirement #: J4 **Requirement Type:** Functional **Event/BUC/PUC #:** 2

Description: View / change account details

Rationale: Q4 of interview - Participants mention a selection of different profile details needed during task assignments

Originator: Jayden Everest

Fit Criterion: Be able to view any users profile, seeing their job title, location/timezone, name, groups/departments assigned to, and contact details. Also, ensure changes made to a profile are stored between sessions. If both criterias are met, then requirements met.

Customer Satisfaction: 3 **Customer Dissatisfaction:** 2 **Priority:** 1 **Conflicts:** 0

Supporting Materials: Document Summarized Answers | General Interview Conclusions

History: Created on 18th October 2022

Requirement #: J5 **Requirement Type:** Functional **Event/BUC/PUC #:** 3

Description: Have varying and updatable accounts and access levels

Rationale: Q3-1 of interview - Participants mention senior staff & admins having different account privileges.

Originator: Jayden Everest

Fit Criterion: First, check that accounts can be created, and signed into, with both the correct password and username required. Then, ensure all features available to

non-admin users are necessary for them to complete their job (E.G Testers must be able to create tickets, but don't need to be able to change their priorities).

Customer Satisfaction: 1 **Customer Dissatisfaction:** 3 **Priority:** 5 **Conflicts:** 0

Supporting Materials: Document Assumptions | Summarized Answers | General Interview Conclusions

History: Created on 18th October 2022

Requirement #: J6 **Requirement Type:** Functional **Event/BUC/PUC #:** 7

Description: View / change groups / departments & assign tasks to them

Rationale: User Story C - Administrative departments need to be able to assign tasks to large groups of employees quickly

Originator: Jayden Everest

Fit Criterion: Create a new department, assign 3 users to said department, and then assign 2 tasks to that department. If all assigned users receive the tasks, then requirements are met.

Customer Satisfaction: 4 **Customer Dissatisfaction:** 5 **Priority:** 4 **Conflicts:** 0

Supporting Materials: Document Assumptions | User Stories

History: Created on 19th October 2022

Requirement #: J7 **Requirement Type:** Functional **Event/BUC/PUC #:** 7

Description: View / change projects

Rationale: Q4 - Interviewees want to know what project they're working on

Originator: Jayden Everest

Fit Criterion: Create a new project, including a name and description, and assign several users to the project. If all assigned users can now access the project, the requirements have been met.

Customer Satisfaction: 2 **Customer Dissatisfaction:** 3 **Priority:** 3.5 **Conflicts:** 0

Supporting Materials: Document Assumptions | Summarized Answers

History: Created on 19th October 2022

Requirement #: K1 **Requirement Type:** Functional **Event/BUC/PUC #:** 5

Description: Bug tracking system will need to be usable at all times of the day/night.

Rationale: Survey Q6, survey responses suggest there is no standardised time of day for bug tracking/solving.

Originator: Kevin Karati

Fit Criterion: Create a Hi-Fi prototype that will be tested under different lighting conditions. Requirements will be met if subjects are able to complete tasks assigned to them at comparable speeds across bright, dark, and typical* lighting environments. (*Typical being equivalent to between 500-1000 lux, bright being equivalent to midday sunlight on a screen, and dark being equivalent to an unlit room at night)

Customer Satisfaction: 3 **Customer Dissatisfaction:** 3 **Priority:** 3 **Conflicts:** 0

Supporting Materials: Survey Q6

History: Created on 17th October 2022

Requirement #: K2 **Requirement Type:** Functional **Event/BUC/PUC #:** 5

Description: Design of the bug tracking system will mostly be standardised to ensure a high level of consistency within the system, especially when working as part of a group.

Rationale: Survey Q8, 62.5% of survey participants believed that a standardised system design is preferable to a customisable design.

Originator: Kevin Karati

Fit Criterion: If the core structure of the tracking system cannot be customised between users, then the requirement has been met.

Customer Satisfaction: 3 Customer Dissatisfaction: 3 **Priority:** 3 **Conflicts:** 0

Supporting Materials: Survey Q8

History: Created on 17th October 2022

Requirement #: K3 **Requirement Type:** Functional **Event/BUC/PUC #:** 1

Description: Archive bugs that have been solved for future reference

Rationale: Survey Q9, 87.5% of survey participants said bugs that have been solved should be archived, not deleted. User Story E also supports this rationale.

Originator: Kevin Karati

Fit Criterion: Provide a checkbox or similar for users to select once a bug has been solved. If the bug is moved from the list of bugs to be solved to an library of solved bugs, the requirements has been met

Customer Satisfaction: 5 **Customer Dissatisfaction: 5** **Priority:** 5 **Conflicts:** 0

Supporting Materials: Survey Q9

History: Created on 17th October 2022

Requirement #: K4 **Requirement Type:** Non-Functional **Event/BUC/PUC #:** 6

Description: The program needs to be able to run without extra software needing to be installed

Rationale: Survey Q10, 87.5% of participants agreed that our application should be able to run as a standalone program, without the need for third party software.

Originator: Kevin Karati

Fit Criterion: If the entire program runs on an up-to-date version of Windows 10 (As of the time of release), without any external programs, requirement has been met.

Customer Satisfaction: 2 **Customer Dissatisfaction:** 4 **Priority:** 2 **Conflicts:** 0

Supporting Materials: Survey Q10

History: Created on 17th October 2022

Requirement Events

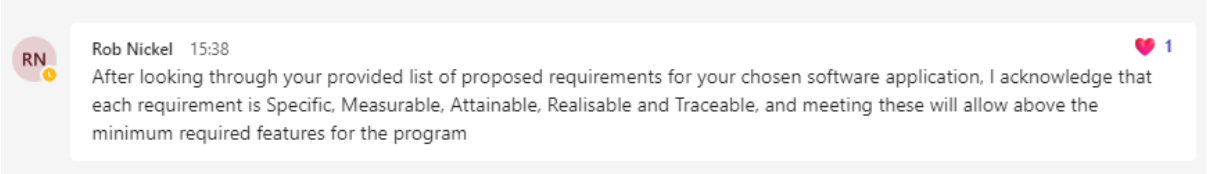
1. Tickets/Defects
2. Account
3. Security
4. Efficiency
5. Interface
6. Distribution
7. Management

Requirement Verification

After creating the project requirements, we sent our requirements to our primary client, who upon inspection found the requirements to be suitable and comprehensive for the project.

Fig. 12

Requirement verification proof



Research Conclusion

After conducting and analysing both our interviews and surveys, as well as creating several user stories and a selection of requirement snowcards, we were able to identify a broad number of needed features for our program. We were also able to identify a selection of more generalised user and business requirements.

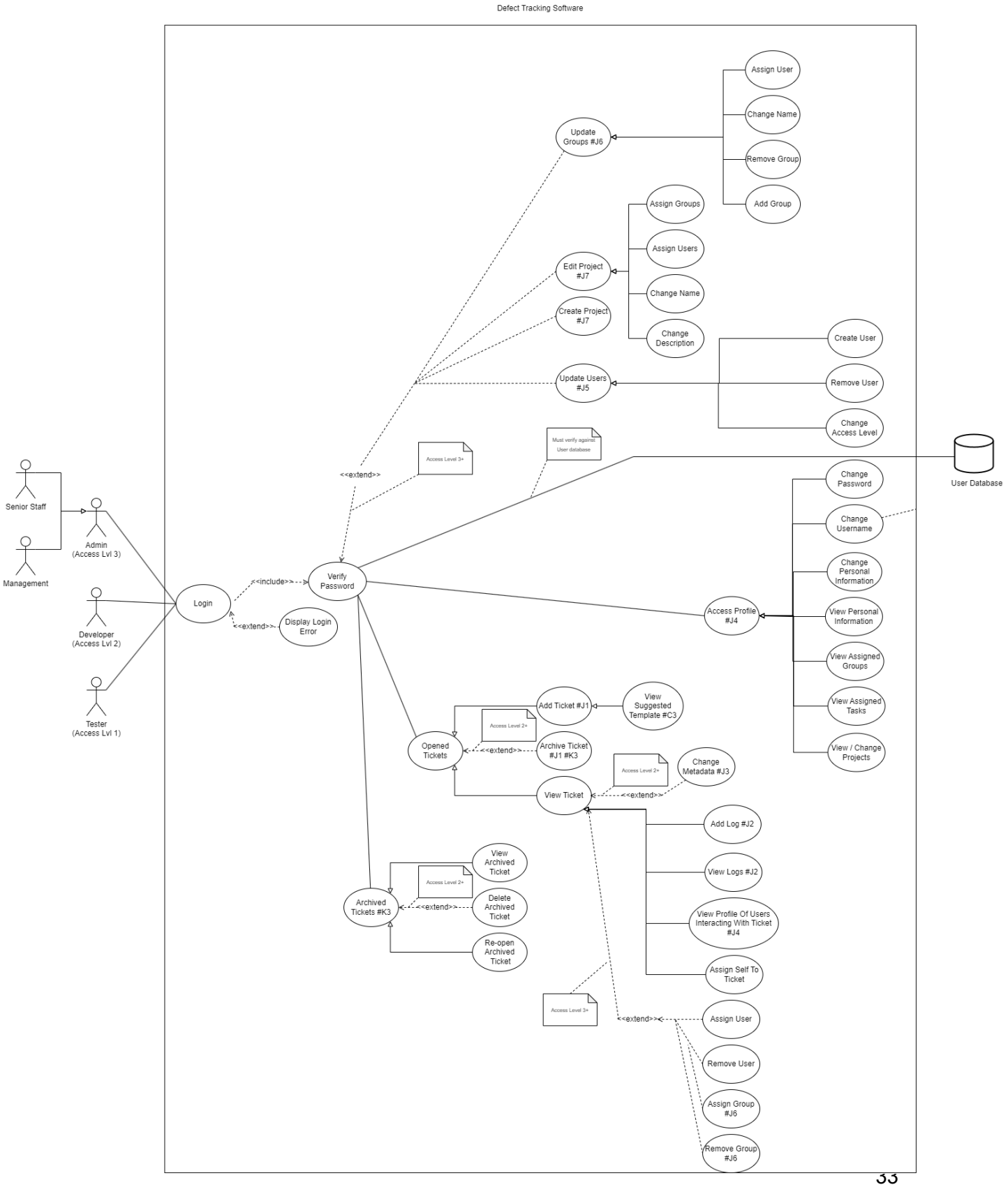
In terms of users, they need to be able to quickly login and create tickets, so emergency issues can be submitted and resolved faster (See ***User Story A***). Also, user task discovery and assignment needs to be easy, so that less time is spent finding a task to work on, and more time is spent actually completing the task (See ***User Story B***). Lastly, administrative users need to be able to assign large groups of developers to the same tasks quickly (See ***User Story C***).

In terms of the businesses using our application, they expect the application to effectively distribute tasks to all employees. They also need to be able to track completed tasks, to calculate project progress. Lastly, businesses need to be able to track individual user progress (Which tickets were opened by them, which tickets were solved, etc), to measure individual productivity.

Use Case Diagram

Fig. 13

Use Case Diagram



Design

Sketches

To get a rough idea of the layout of our program, we each created a sketch of our program, making sure everything in the Use Case Diagram was included in the sketches. After this, we discussed what we liked about each sketch, and decided on the final sketch moving forward. The sketch we decided on was Jayden's, but with the priority iconography of Conor's, and the ticket categories of Kevin's.

Fig. 14

Jayden's Sketch

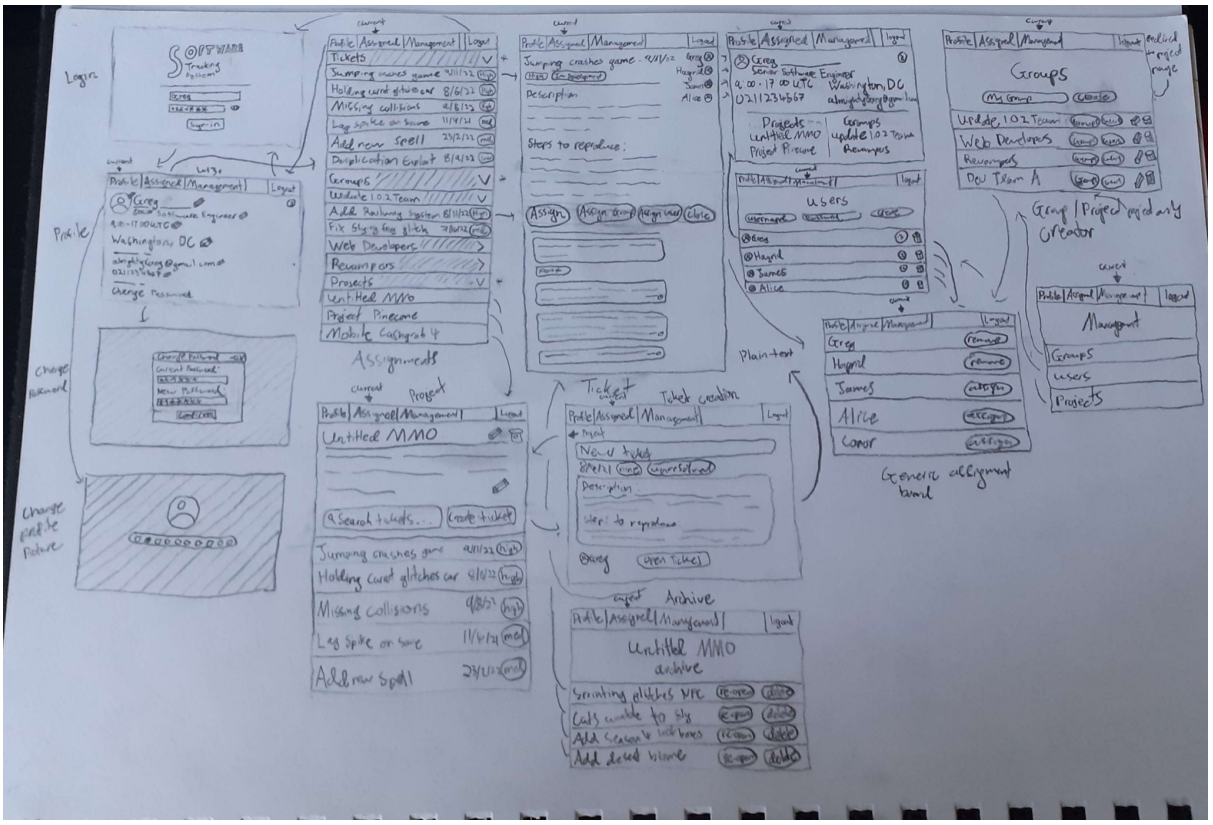
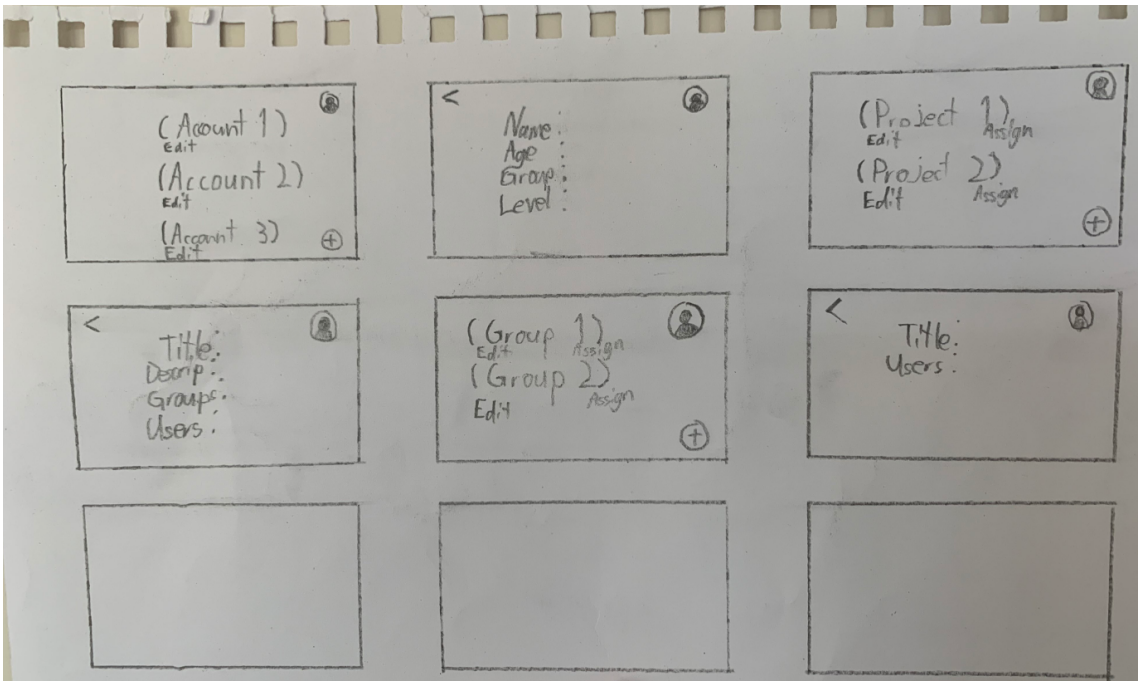


Fig. 15

Conor's Sketches



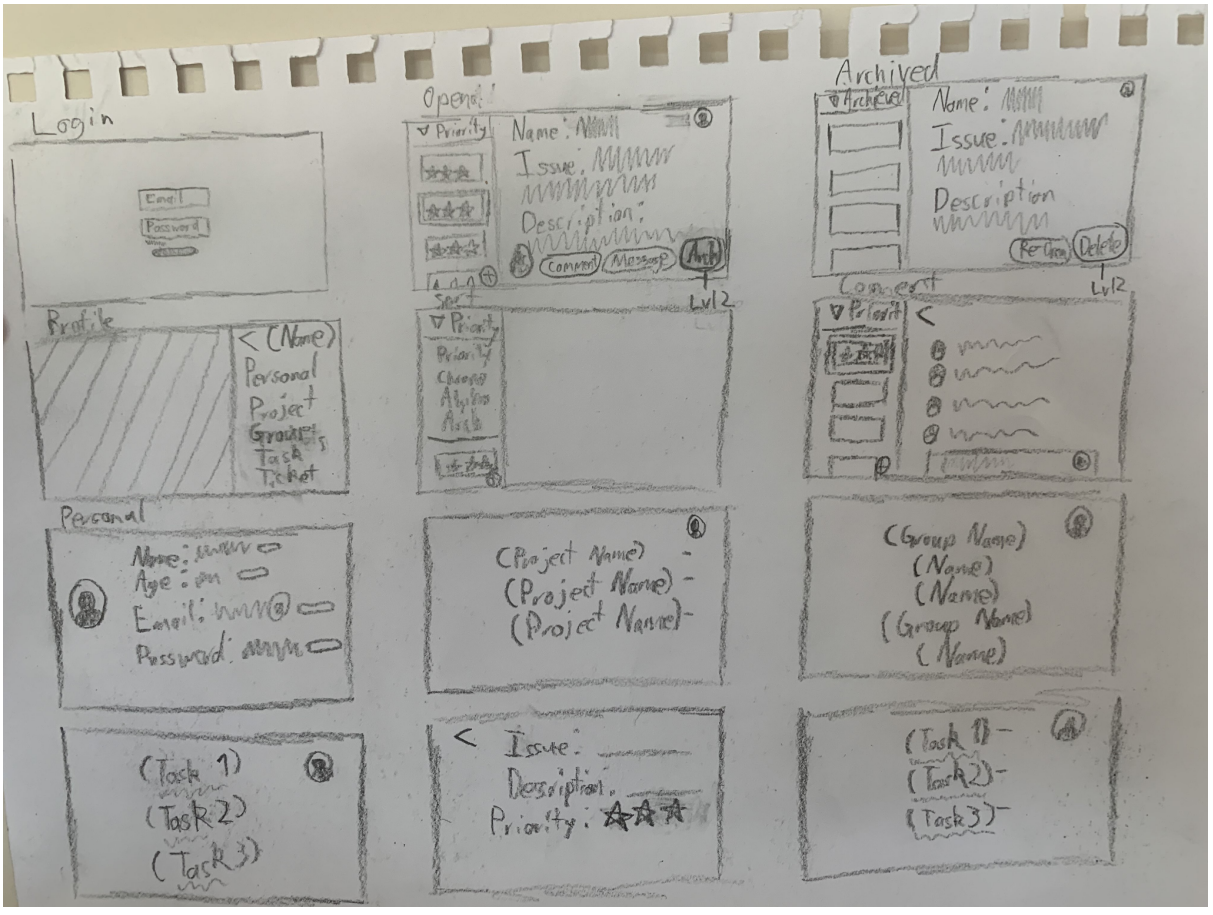
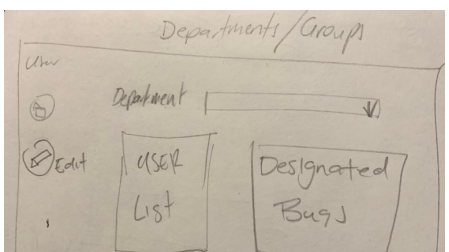
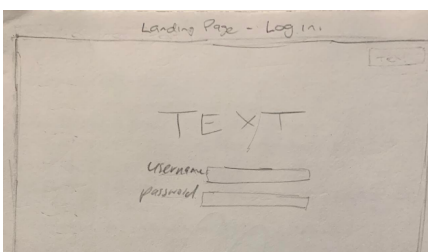
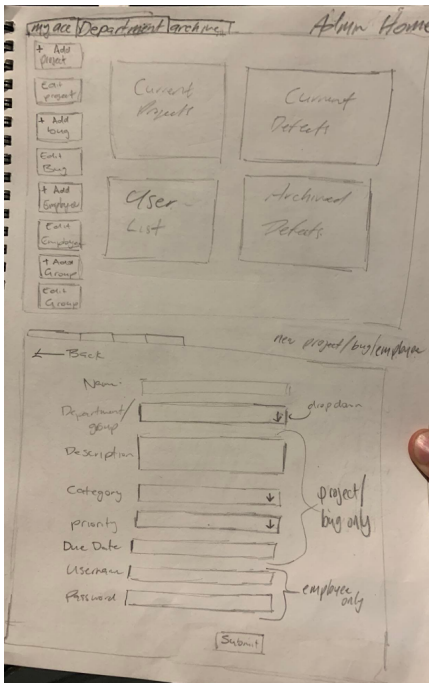


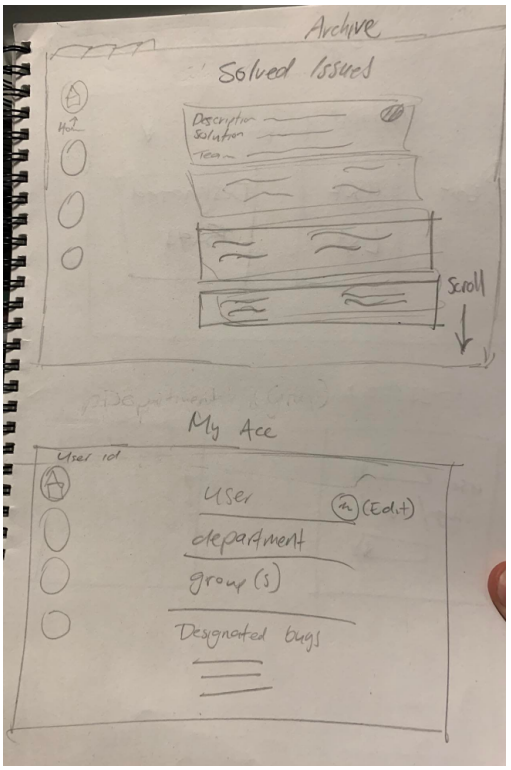
Fig. 16

Kevin's Sketches



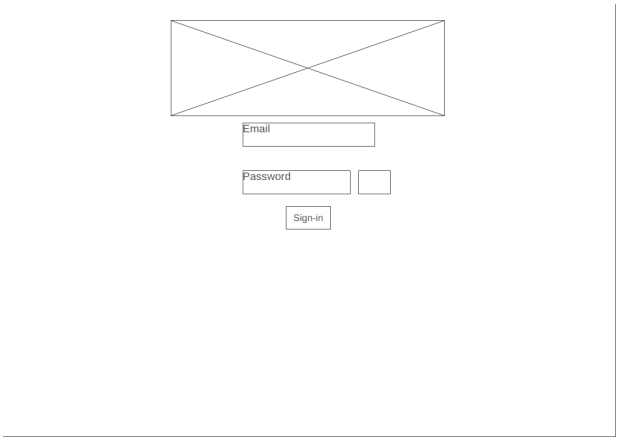
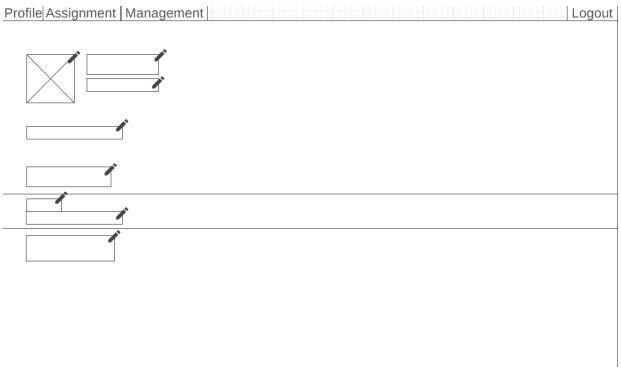
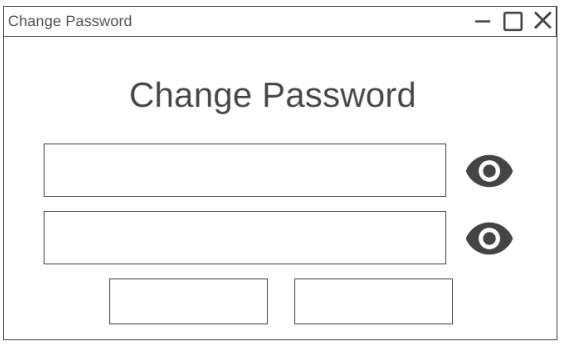



Admin Home
Current Projects
Current Defects
User List
Archived Defects
Back
Name
Department/group
dropdown
Description
Category
priority
Due Date
project/bug only
Username
Password
employee only
Submit


Archive
Solved Issues
Description
Solution
Team
Scroll
My Acc
User id
User
(Edit)
department
group(s)
Designated bugs

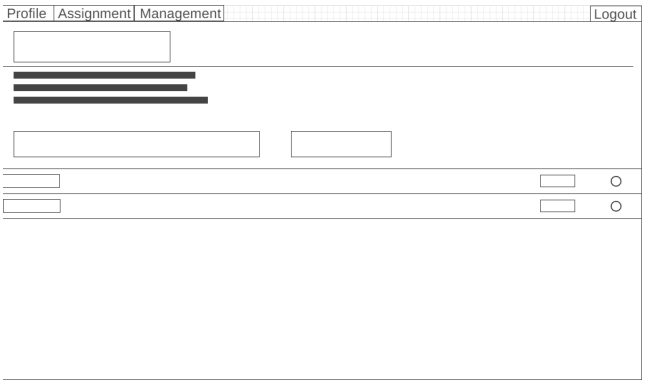
Lo-Fi

Next, we created a digital Lo-Fi, drawing inspiration from all our sketches. We did this so we could gain a better understanding of the general structure and flow of the program.

Name / Link / Purpose	Screenshot
Login https://wireframe.cc/mGncxf *To separate users based on their assigned access levels, and so that users can be identified / tracked when interfacing with the system.*	
Homepage / Profile https://wireframe.cc/WxbuDb *To allow users to edit their profile information, and to act as a portal to the program's high-level systems (Assignments & Management [Admin]).*	
Change Password Sub-Window https://wireframe.cc/R4uvU0 *To allow users to change their password, while providing security in case of an account breach.*	

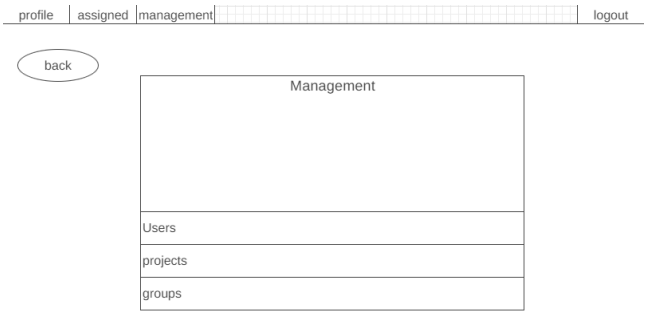
Assignments
https://wireframe.cc/SxTI7l
To display all tickets, groups, and projects the user has been assigned to.

Project
https://wireframe.cc/KaT9h0
To give a rough overview of a project's purpose, and as a portal to all the tasks part of the project.

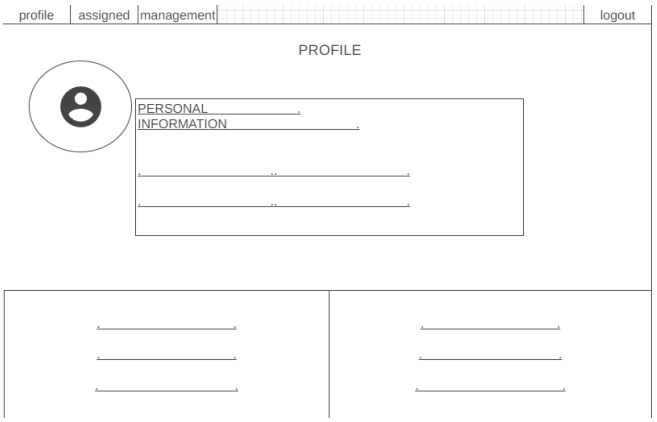


Ticket
https://wireframe.cc/UOq2ak
To display information on a ticket, as well as the users assigned to it, the ticket log, and to give an interface for manipulating ticket meta-data / adding to log.

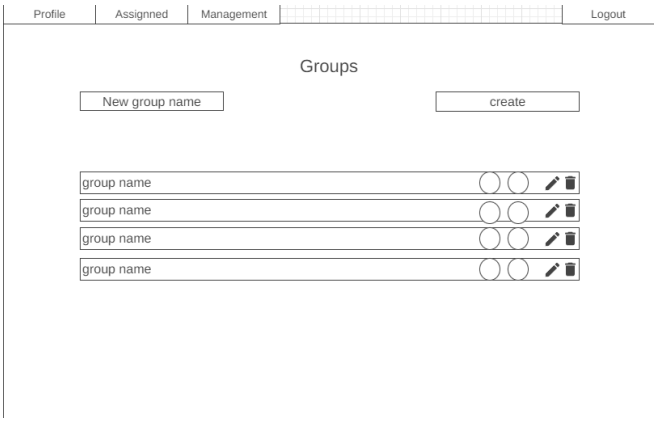
Management
https://wireframe.cc/DHc3su
To act as a portal to the management subsystems (Group, Project, and User subsystems).



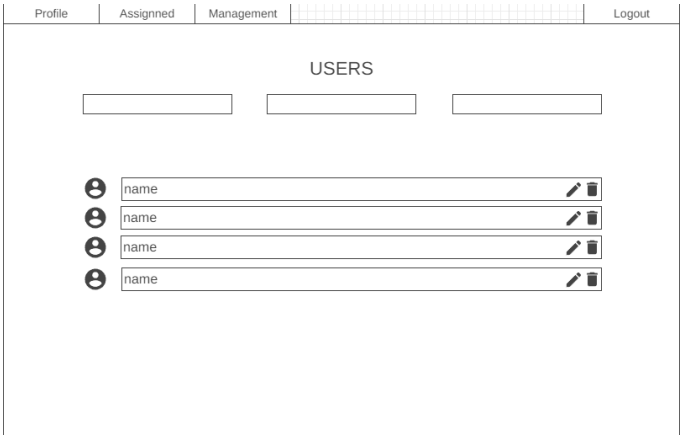
User Profile Viewing
https://wireframe.cc/wG704B
To display non-sensitive information on any user, primarily to check whether users met particular criteria, or to find their contact information.



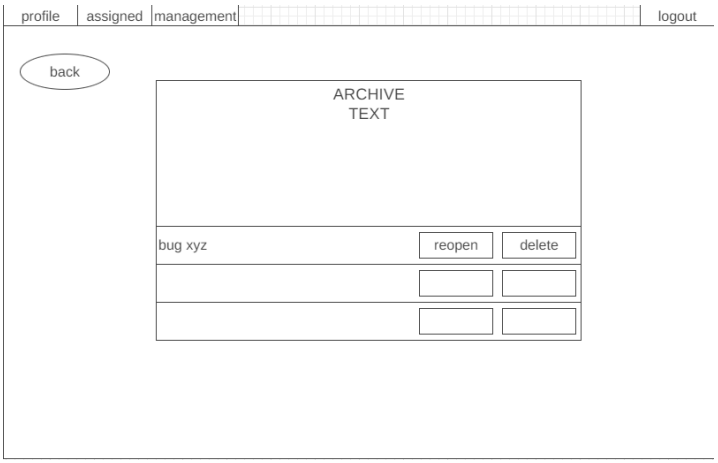
Groups
https://wireframe.cc/bAfbb3
To display, edit, add, and remove groups, and to re-assign projects / users to groups.



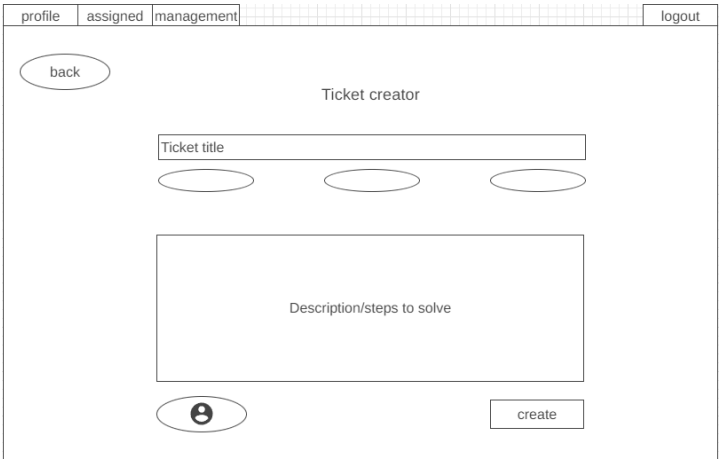
Users
https://wireframe.cc/HzjleK
To display, edit, add, and remove users.



Archive
https://wireframe.cc/iW1Rsx
To display, edit, and to act as a portal to all archived tickets in a project.



Ticket Creator
https://wireframe.cc/dw84Ns
To create a correctly formatted ticket for a project.



Lo-Fi Testing

After creating the Lo-Fi, we needed to confirm that our design was navigable to end-users. To confirm this, we got several participants to logout from the profile, view tasks assigned to them, and find details on a project assigned to them. We also timed this, however since they were using the much harder to understand Lo-Fi, so long as it didn't take them more than a minute to complete each task, we decided that would be acceptable. Lastly, we got each participant to look at each of the pages, and to critique them.

Profile to Logout (Sec)	Profile to Tasks (Sec)	Profile to Project Description (Sec)	Thoughts
2	4	10	Archive doesn't need a "delete". A "details" button could be useful to view solutions. There is no way to access the archive from the starting page of the Lo-Fi. Also, the profile

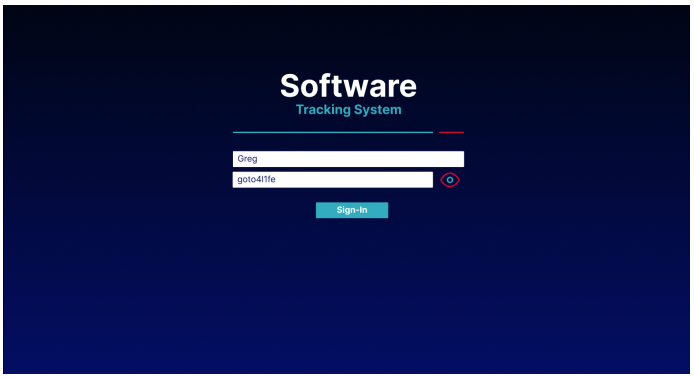
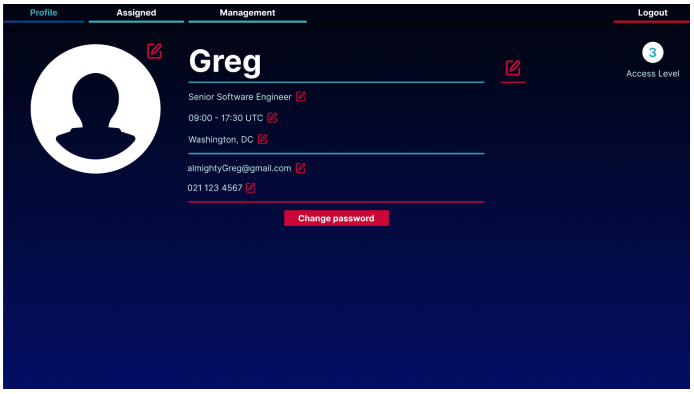
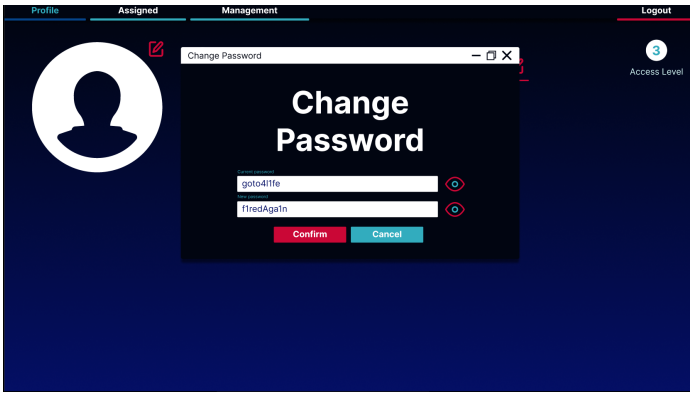
			page feels too empty, why don't you move the information to the center of the screen?
3	6	8	Some pages were difficult to understand due to low detail but the layout looks good.
2	6	9	All good, looks like the system will be very easy to navigate.
7	12	17	Group creation button feels too far away from the name input.
1	5	14	Navigation is good. Last task required a bit of wandering but was easily found
1	3	9	Layout was easy once user explored a bit and understood where everything was
3	8	11	Navigation was easy for the users. No major changes to the layout seem necessary.
Avg 2.7	Avg 6.3	Avg 11.1	

Lo-Fi Conclusions

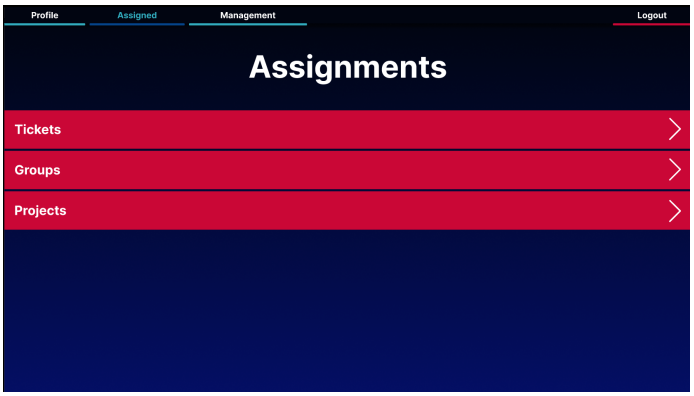
Looking at our observation results, we found all tasks to be within the acceptable time period (Less than 1 minute). We also gained insight into several things we'll need to change when moving to the Hi-Fi prototype. Namely, information on the profile page needs to be moved to the center of the screen, and the group creation buttons need to be closer together. We also forgot to add a link to the archive in the project page, so we'll need to add that in the Hi-Fi. Lastly, a participant mentioned that archived tickets shouldn't have a delete button, however we feel that not having one would lead to clutter, especially if tickets were accidentally created or had a typo. As such, we'll keep the delete button moving forward.

Hi-Fi

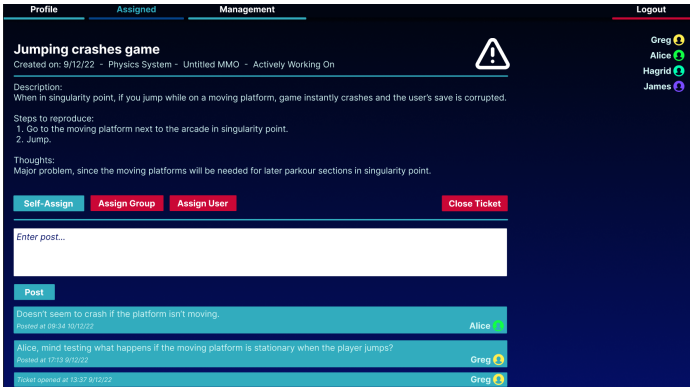
After this we created a navigable Hi-Fi prototype, to visualise what the final program would look like without having to program it. We also made sure to incorporate lessons from the Lo-Fi testing we've done previously. A link to the navigable version can be found below.
https://www.figma.com/proto/NiAijb1C26ibPQv4rqRP6B/Tracking-System?node-id=2%3A21&scaling=scale-down&page-id=0%3A1&starting-point-node-id=4%3A65

Name / Notes	Screenshot
Login	
Profile	
Change password profile sub-window	

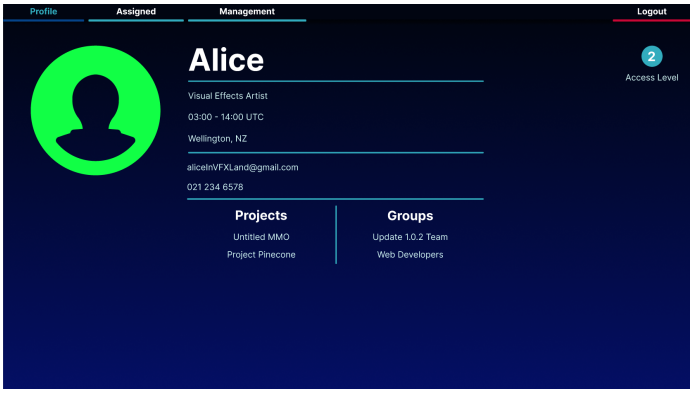
Assignments



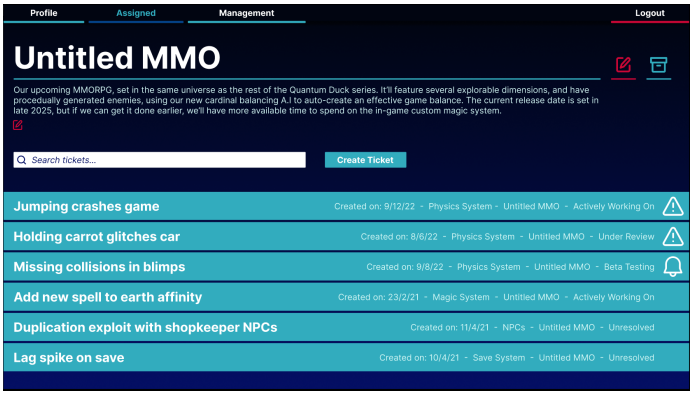
Ticket



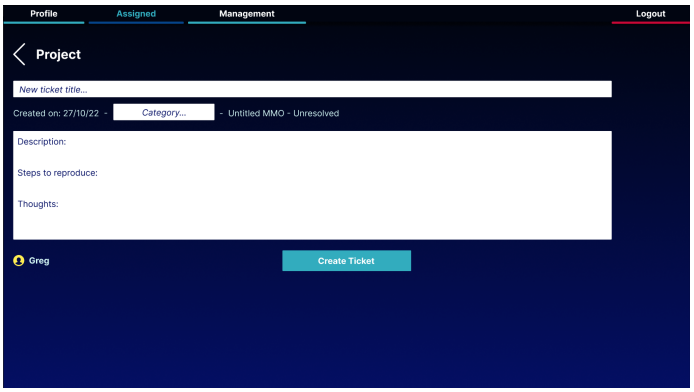
User Profile Viewing



Project



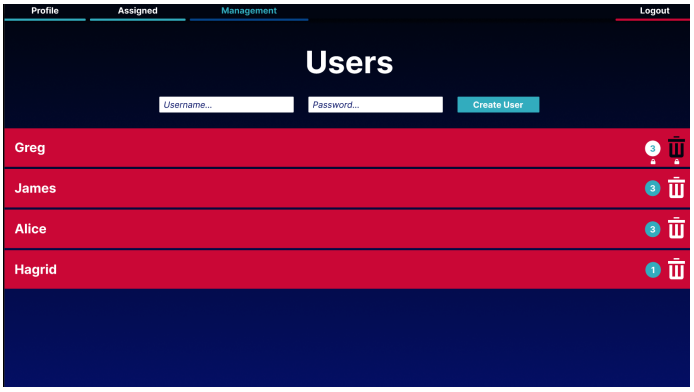
Ticket Creation



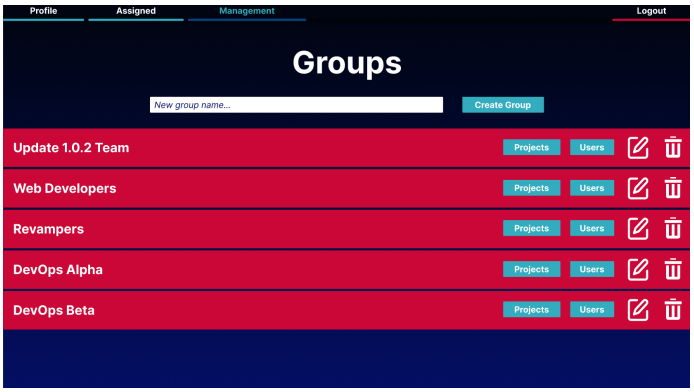
Archive

Management

Users



Groups / Projects



Management Assigning

Hi-Fi Testing

After creating the Hi-Fi, we needed to confirm that all our initial requirements for the program have been met. To do this, we got several participants to complete several tasks, and got them to complete those same tasks in GitHub (2022). Then we compared the two, to confirm that requirement #C1 has been met. We also tested how different lighting environments affected navigation speed, by getting users to navigate the program in different lighting conditions, to confirm that requirement #K1 has been met.

Hi-Fi Results

Sign-in (Sec)	Change profile picture (Sec)	Create project (Sec)	View another user's profile (Sec)
1	5	15	31
1	4	7	12
1	4	6	9
1	7	12	18
1	5	15	14
1	7	14	17
Avg 1	Avg 5.3	Avg 11.5	Avg 16.8

Github Results

Sign-in (Sec)	Change profile picture (Sec)	Create project (Sec)	View another user's profile (Sec)
3	8	23	6
6	10	14	8
4	13	18	9
Avg 4.3	Avg 10.3	Av 18.3	Avg 7.7

Lighting Results

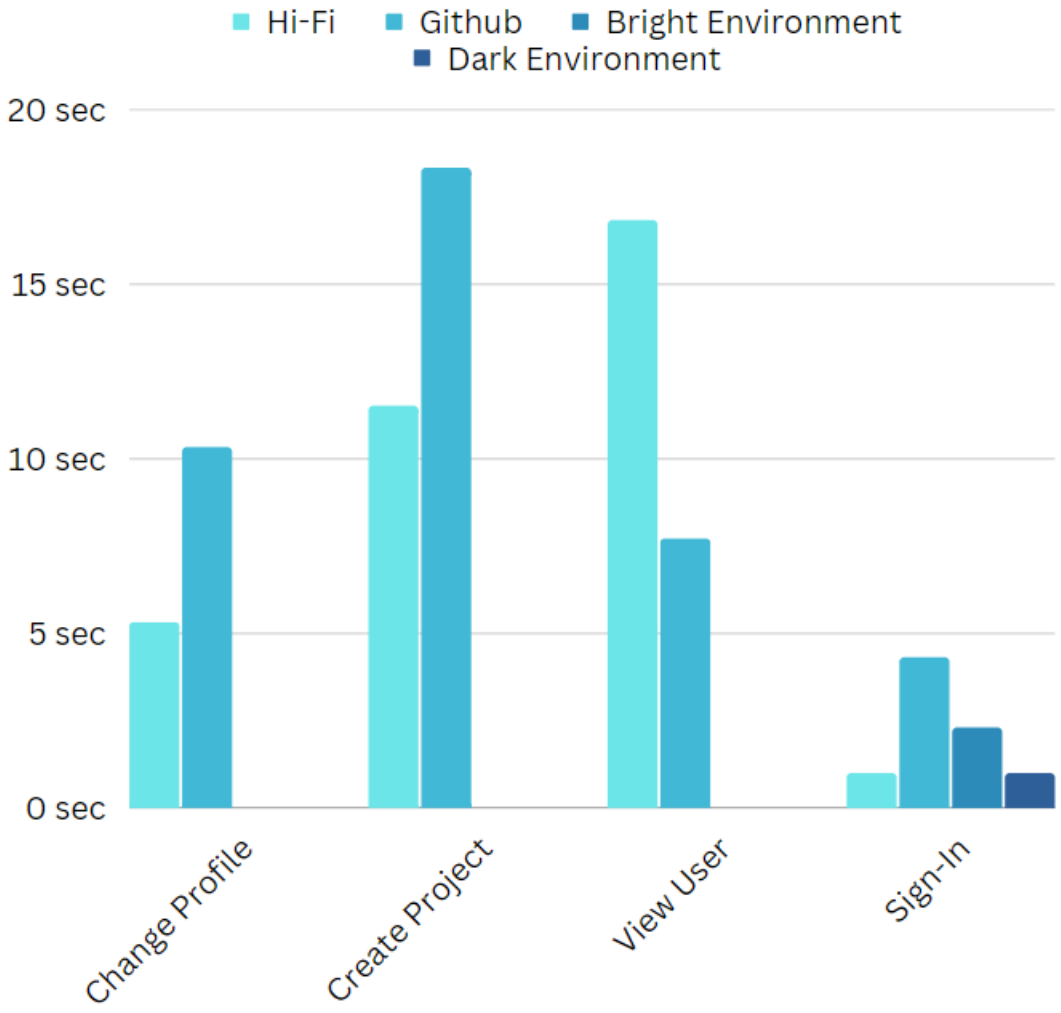
Sign-in in bright lighting (Sec)	Sign-in in dark lighting (Sec)	Sign-in in neutral lighting (Sec)
2	1	1
3	1	1
2	1	1
Avg 2.3	Avg 1	Avg 1

Hi-Fi Result Analysis

Looking at the resultant graph (Below), our Hi-Fi design tends to be substantially quicker to navigate than competing programs, in this case Github (2022), completing requirement #C1. Also, looking at the environmental results in the Sign-In test, we found that differing lighting environments had little impact on the navigation, completing requirement #K1. However, while participants mentioned that they liked the layout colours, they didn't like how dangerous buttons (Such as the delete ticket button) weren't distinguishable from safe buttons (Such as navigation buttons). So, we decided to colour dangerous / elevated privilege buttons red, and safe buttons cyan.

Fig. 17

Hi-Fi research resultant graph

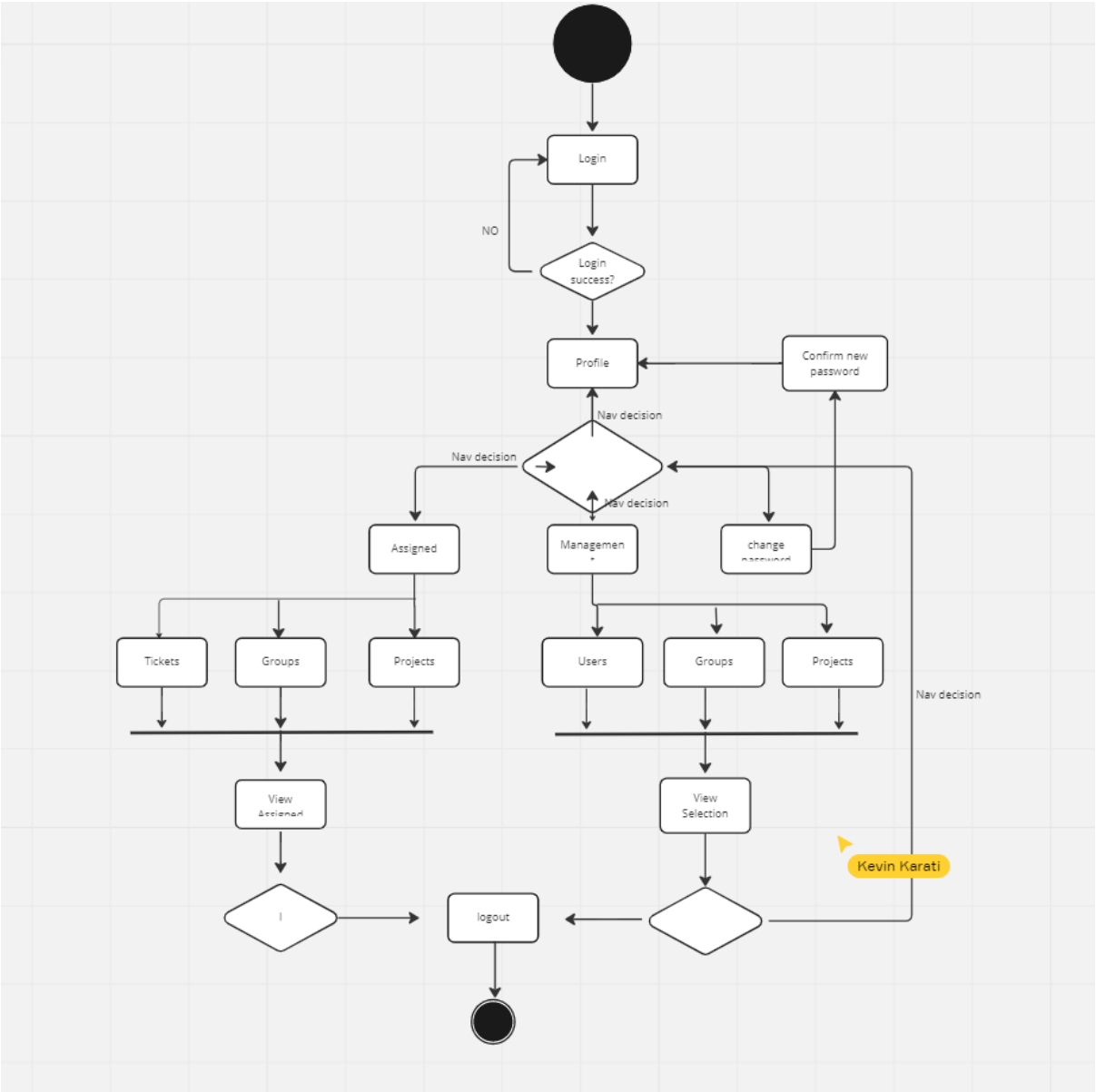

Hi-Fi
Github
Bright Environment
Dark Environment
20 sec
15 sec
10 sec
5 sec
0 sec
Change Profile
Create Project
View User
Sign-In

Project Development

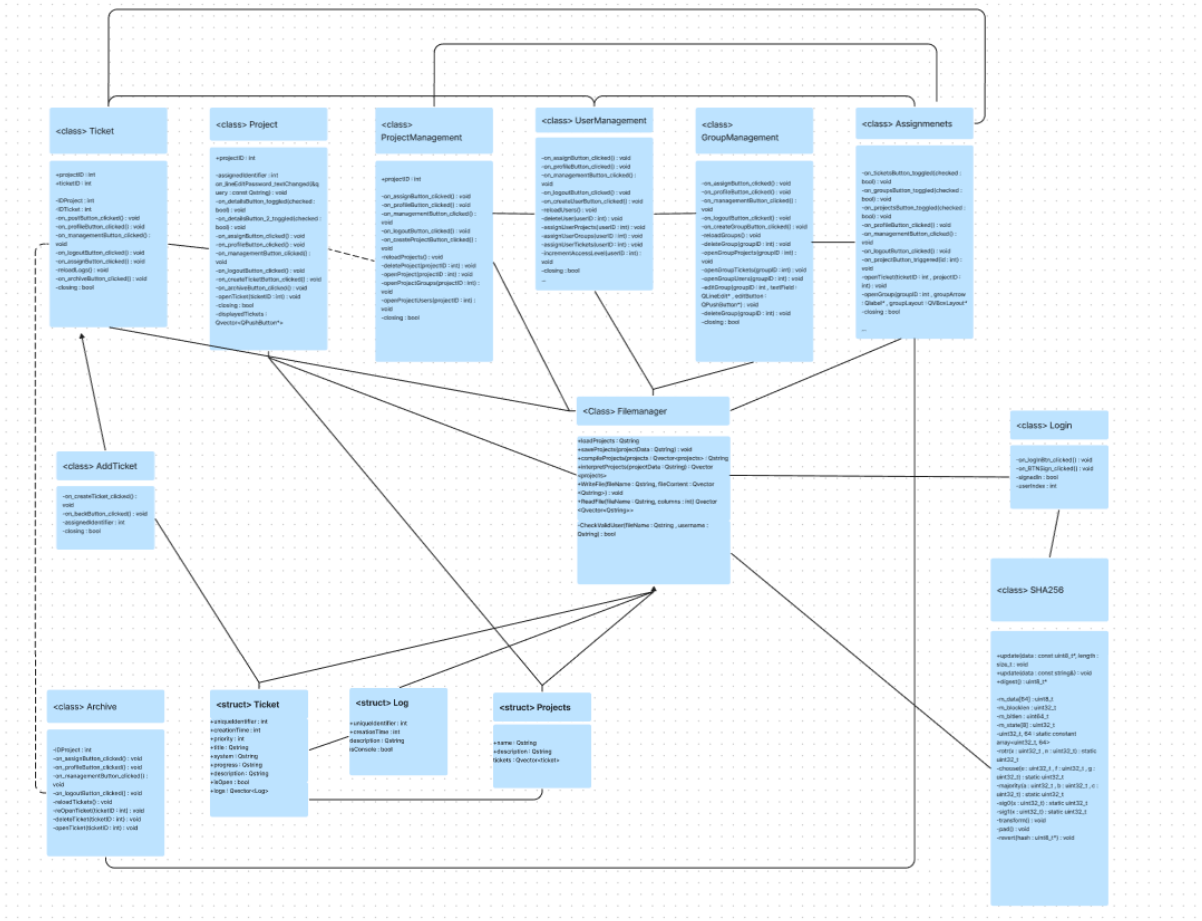
Final Project Description

This is a bug defect logging and tracking system that requires a user login to access. Passwords are protected by a SHA256 hashing algorithm. Access rights are separated into admin rights and developer rights. Admins have the ability to assign, create, or delete projects, groups and users. Admins and developers can view and edit their personal profiles, things that they have been assigned and add tickets to projects.

UML Activity Diagram



UML Class Diagram



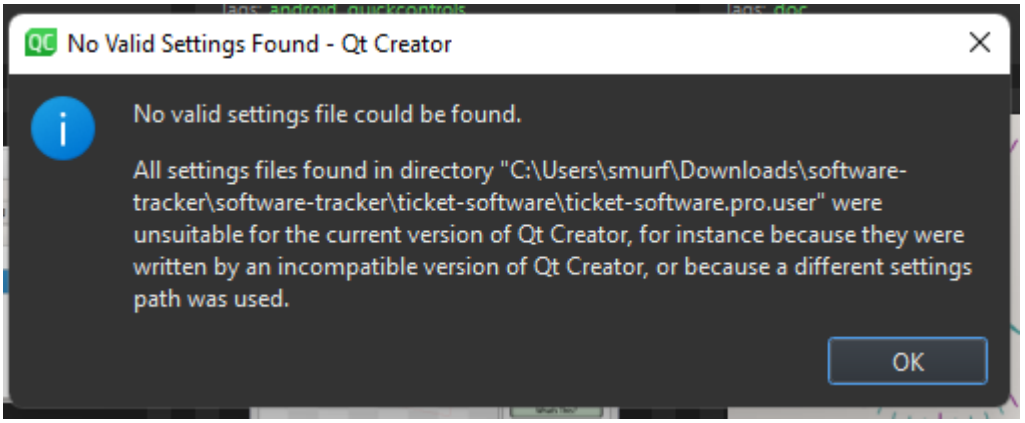
https://www.figma.com/file/KYG8oWuCRpmVAWhyRI7xWm/Untitled?node-id=0%3A1&t=pcpTqph0Plfr4NeW-1

User Installation

Download/copy the ‘software-tracker.zip’ onto your computer. Then extract the folder to a place where you will remember its location.

Open QT Creator and select ‘Open Project’ then find your extracted folder. Open the folder, open ‘software-tracker’, open ‘ticket-software’, then scroll down to the bottom for the folder and find ‘ticket-software.pro’ and double click it, ‘ticket-software.pro’ should be eight positions up from the bottom.

A window will pop up saying, ‘No valid settings file could be found.’



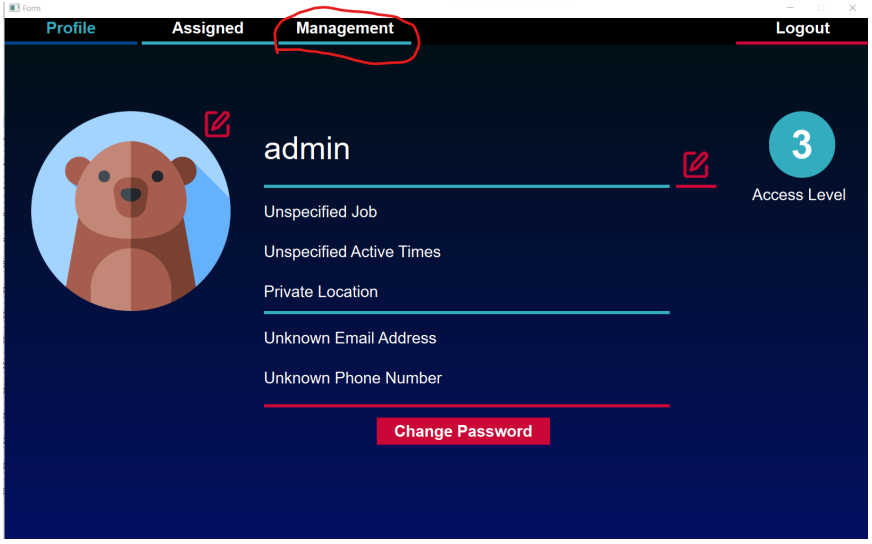
Click ‘OK’, then click ‘Configure Project’. Then there will be two green triangles in the bottom left corner of the screen. One triangle will have a small bug on it, the other triangle will not. Click on the triangle without the bug on it.

The default account’s username and password is ‘admin’ and ‘password’ respectively.

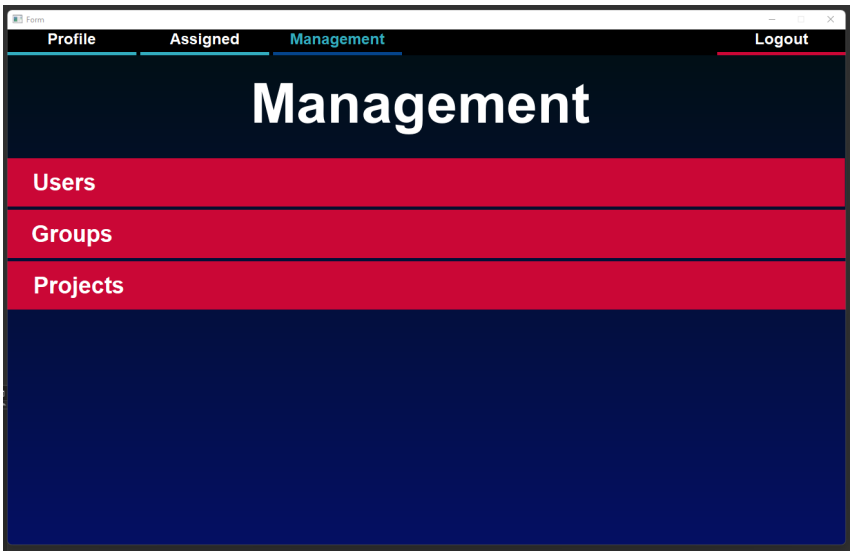
User Manual

Creating a User:

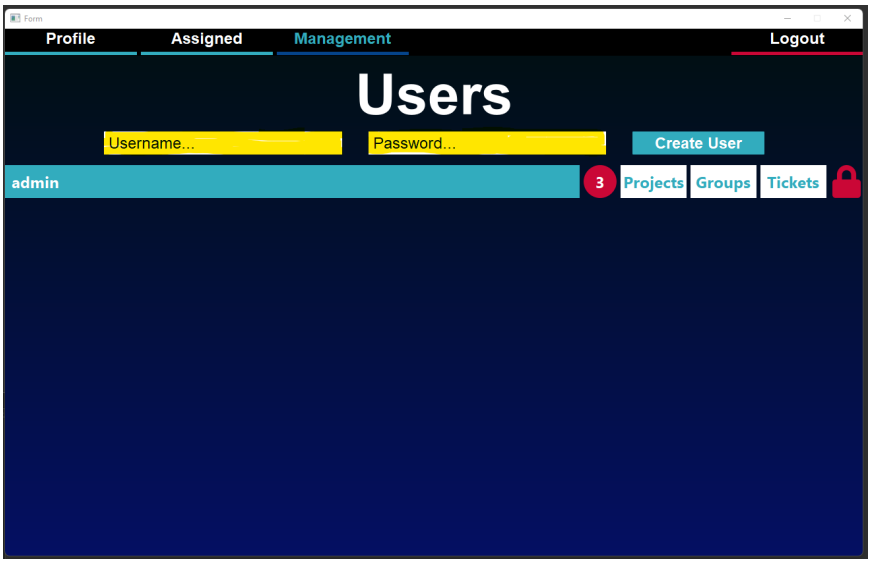
To begin, you need to be logged in as an Admin (Level 3 access)
Step 1: Click on ‘Management’ in the Navigation bar



Step 2: Click on the red tab that says ‘Users’



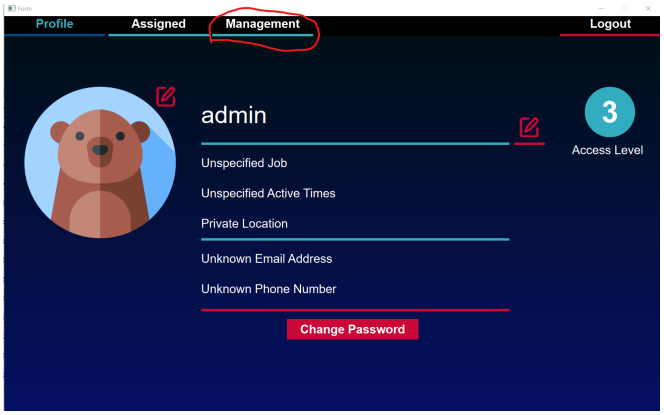
Step 3: You will then come to a page that looks like the following. You will then need to fill out the boxes label ‘Username’ and ‘Password’ (Highlighted yellow in the image below). After both boxes are filled click the blue ‘Create User’ Button on the right.

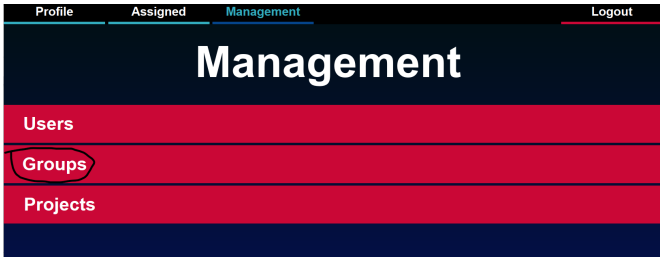


Congratulations you successfully created a user.

Admin creating a group:

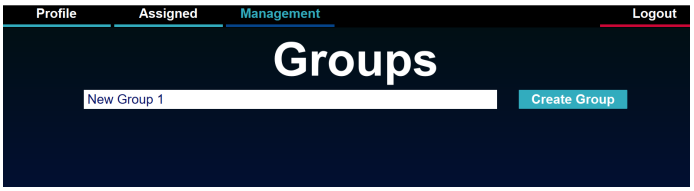
Step 1: Click on the management tab in the navigation bar, then click on the button labelled ‘Groups’





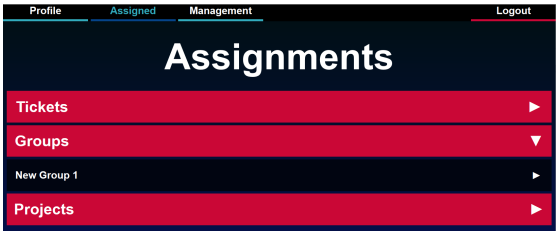
Step 2:

Enter the new group name and click ‘Create Group’, the new group should appear below



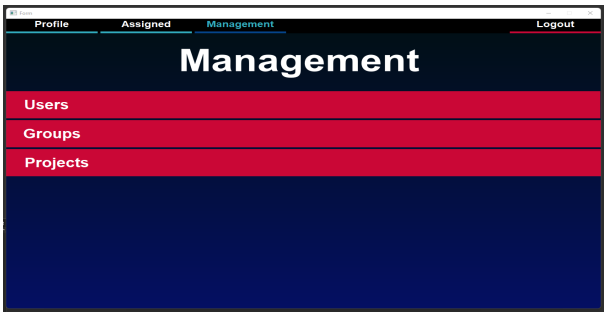
Step 3: To add users to a group, click the ‘Users button’ to the right of the newly created group. Then choose users to add by clicking on the ‘add’ button next to the users name.

Step 4: Once added, the user can see what group they have been assigned by clicking on the ‘Assigned’ tab on the navigation bar



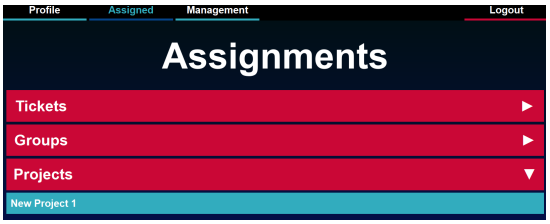
Admin Creating a Project

Step 1: Click on the ‘Management’ tab on the navigation bar, then click on projects at the bottom of the page.



Step 2: Enter the name of the new project in the text field and click ‘Create Project’, the project should appear below. To delete the project, select the trash icon on the far right of the page.

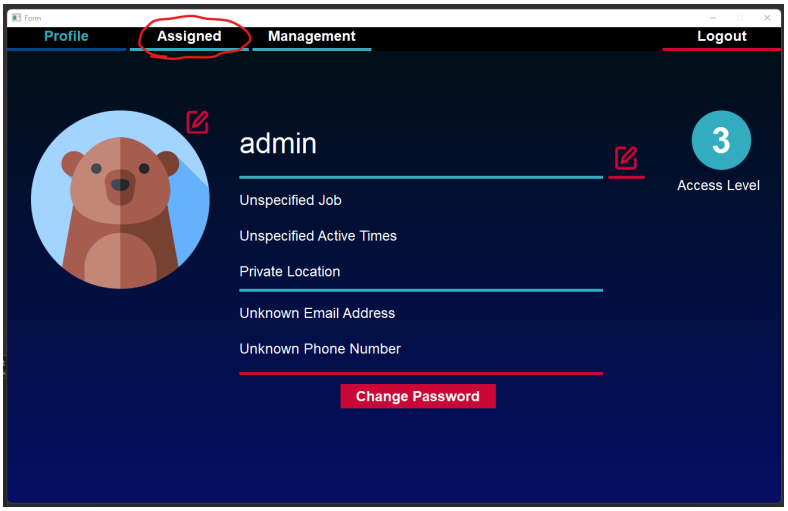
Step 3: To add users or groups to the project, select one of the options next to the new project, then select the user or group to add by clicking the ‘add’ button next to the users name. Once added, the user or group can see what project they have been assigned by clicking on the ‘Assigned’ tab on the navigation bar



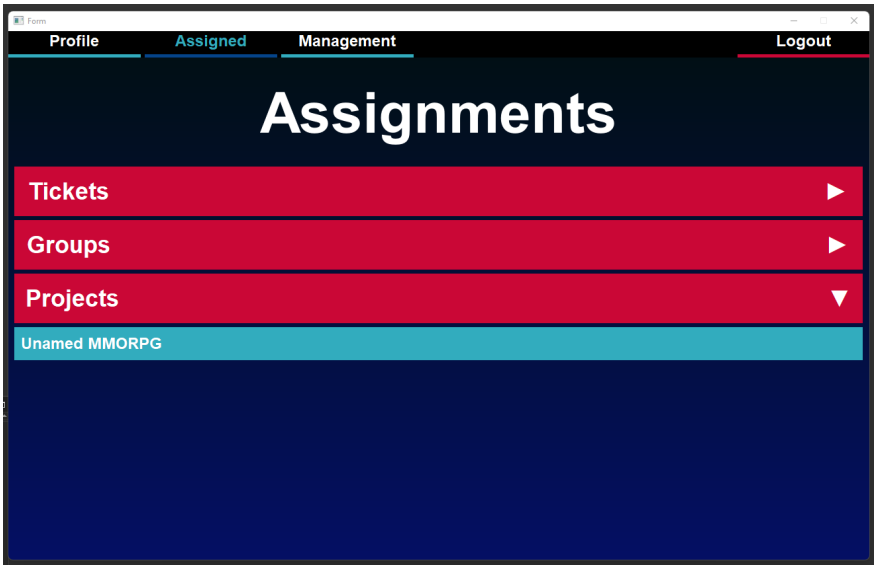
Creating a Ticket:

To do this you need to be assigned to a project (see the steps above for making a project).

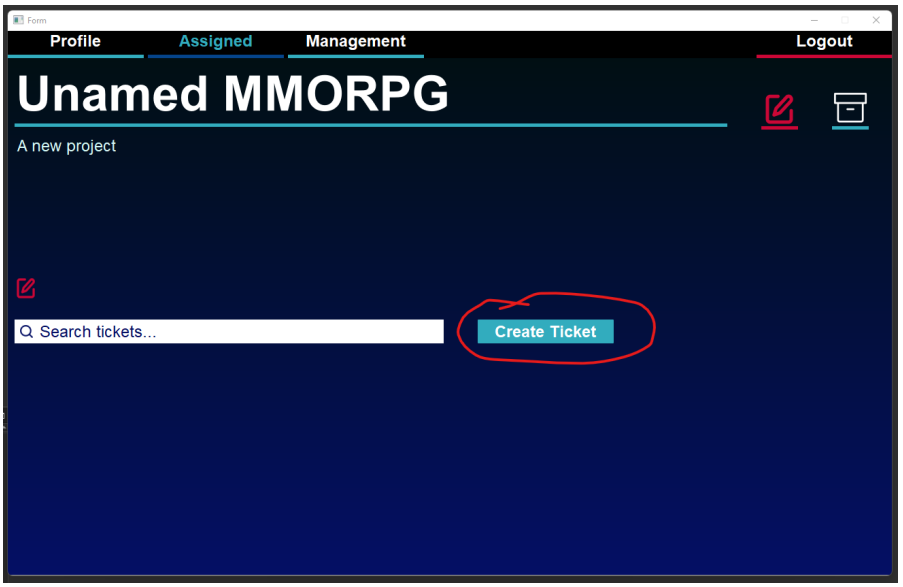
Step 1: Click on ‘Assigned’ in the Navigation bar



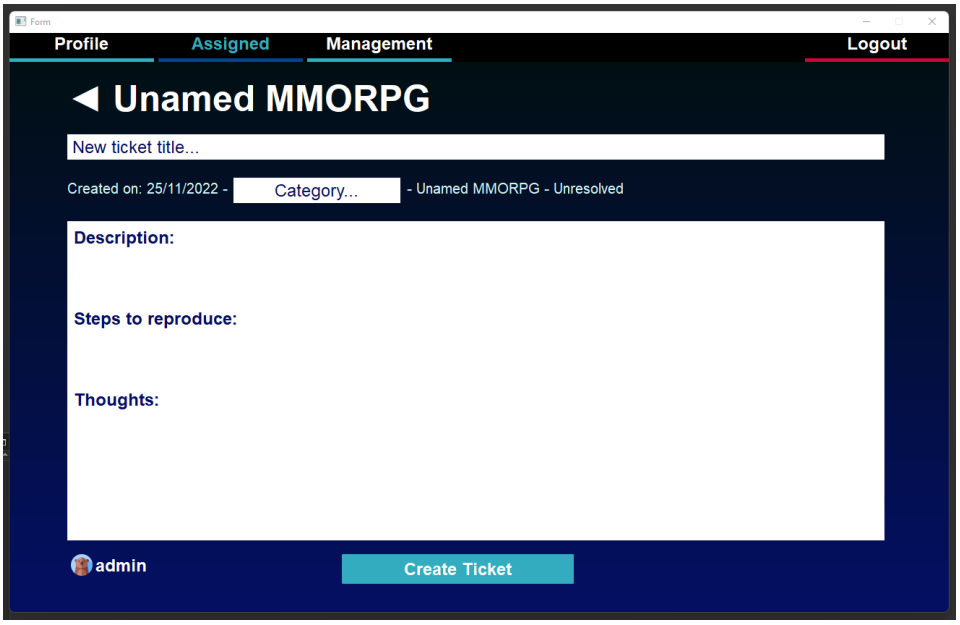
Step 2: Click on the red tab that says ‘Projects’. Multiple projects might appear, but click on the project you want to create a ticket for.



Step 3: Click on the blue button called ‘Create Ticket’



Step 4: Fill out all the fields and then click ‘Create Ticket’.



User Blackbox Testing:

For user testing we gave the users a working version of the program on a laptop. We gave the user time to explore the program first and when they were ready we asked them to complete the following tasks:

- Change your profile details
- Create a User
- Create a Group

- Create a Project
- Create a Ticket
- Assign a User to a Group
- Assign a User to a Project
- Assign a User/Yourself to a Ticket
- Resolve a Ticket

These questions would test their navigational skills and understanding of the Project features

User 1:

At the beginning they were slow as they were intrigued by the ability to change details on the profile menu. They had fast navigation skills and were logging in and out of accounts fast to see different views from all the accounts. Their comments afterwards were,
“I like the amount of customizability with the users, it’s just the right amount for a business setting.”

There wasn’t anything they’d wish to change about the program.

User 2:

Their navigation was good, they found most features relatively quickly and after they explored it all they went back to try some of them. Nothing particularly noteworthy happened. Their comments afterwards were,

“This is a very well built program. I’d like there to be a user description but I understand why there isn’t.”

There wasn’t anything they’d wish to change about the program.

User 3:

They immediately found the management tab and started making a lot of users, tickets, groups and project

s. They paid no attention to user customizability aside from assigning a name to the users. Their comments afterwards were,

“This is very accurate to real programs we use, far more simple but accurate.”

There wasn’t anything they’d wish to change about the program.

User 4:

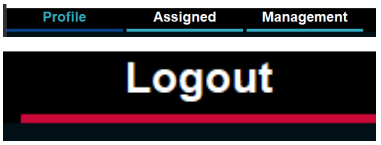
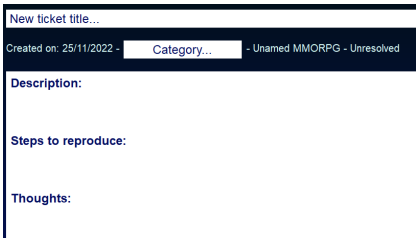
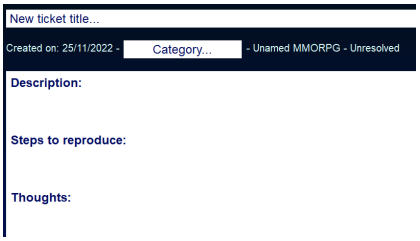
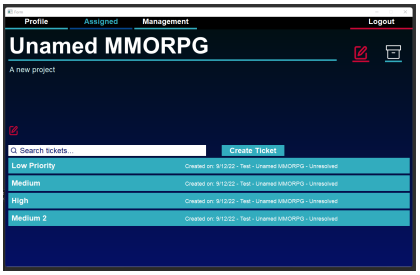
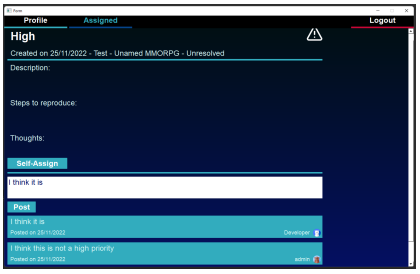
Their navigation of the program was quite poor, they constantly asked me where something was in the program. Their comments afterwards were,

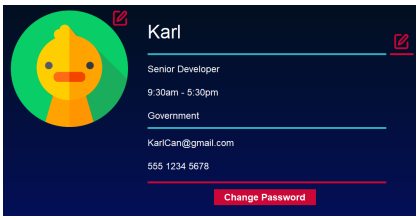
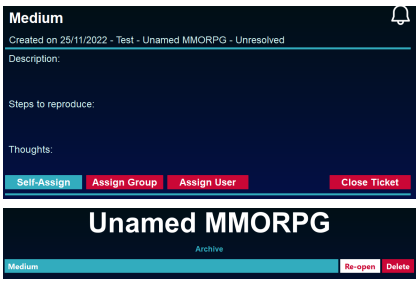
“The navigation is just a little bit too complicated. Otherwise it’s really good. Very well thought out”

Navigation is the only thing they’d like to change.

Key Functionality

These are ten features from the snow cards above that have been implemented into the program.

Easy Navigation		Navigation bar is always visible. It allows users to go to any main page in the program and log out quickly
Freedom when adding information to tickets		We allow the user to give very detailed descriptions on the tickets
Suggested Ticket Template		Again we have 5 fields users need to fill out for their tickets.
Tickets ordered by priority		We eventually decided that instead of ordering it by priority, we’ll order the tickets by date.
View/add logs to tickets		Anyone who is assigned to the project the ticket is in can view and add logs to the tickets for everyone to view.

Users can change account details		Users can change any detail on their account, besides their access level
Change account access level		Only admins can change the account level of other users
Change groups		Admins can change users in a group
Change project		Admins can change users and groups in a project
Archive Tickets for future reference		When a ticket is closed it can be viewed again in the archive part of the project. From there it can either be re-opened or deleted

Conclusion

Through the completion of our research, iterations of testing and revisiting of our functionality assumptions, we have identified flaws in current defect tracking systems and the ideal design path for our system to take.

Interviews with participants within the industry revealed what level of access all users need to complete their tasks as well as how the administrator role is focused more around organisation than security. The interview process has also clarified the requirements of user reports/logs, what metadata is required for each ticket and what level of customisation is needed for user profiles. (See “Real World Interview Conclusion”)

Survey data has detailed the best practice for categorising bugs with a priority system, progress indicators and report formatting. The survey has also outlined the importance of archiving any solved issues and has informed the colour palettes used in our system to ensure usability at all times of the day. (See “Survey”)

We used the information gathered above to review our assumptions and complete our list of functional and nonfunctional requirements. This allowed us to begin developing our Lo-Fi and Hi-Fi prototypes for testing. Lo-Fi testing indicated that the layout and navigability of our system was vastly improved in comparison to the tracking systems currently available. The

participants also provided feedback which was used to tweak our design in preparation for the Hi-Fi prototype development. (See “Lo-Fi Prototype Conclusions”)

Hi-Fi testing further proved the navigability of our system and that we had succeeded in using a colour scheme that did not show a reduction in usability in different lighting conditions. Users commented that there was a need to use colour to differentiate between ‘dangerous’ buttons (such as “logout” and “delete”) from safe buttons and this was taken on board. (See “Hi-Fi Result Analysis”)

Through black box testing, we have found that our finished application has been well received and we have stayed true to our project goal of creating a simple and useful system for end users. 75% of users tested enjoyed fast navigation and thought it was accurate to systems in the current market place while being far simpler to use.

These factors have given us the confidence that the bug tracking system we have developed has all the necessary functionality, is easy to navigate, is usable at any time of day and has met all other requirements necessary for it to be successful.

Glossary

- UI: User Interface, how the end-users interact with the program / systems.
- Lo-Fi: Low-fidelity wireframe, a non detailed digital image meant to show the layout of the user interface.
- Hi-fi: High-fidelity prototype, a fully detailed and interactable design, with the goal of feeling identical to the final program (Minus the functionality).
- Dev: Shorthand for Software Developers
- Admins: Administrators, responsible for project management and security.
- XP: Extreme Programming, a modern development methodology focusing on productivity and mental health.
- RAD: Rapid Application Development, a development methodology focusing around code recyclability.
- QT: A multi-platform software for creating graphical interfaces in C++
- C++: A high-level programming language, focusing on end-product speed and efficiency.

References

Trello, an online collaborative Kanban workflow | Website | Retrieved 28/10/22 from https://trello.com/

Figma, an online visual prototyping software | Website | Retrieved 28/10/22 from https://www.figma.com/

Wireframe.cc, an online wireframing tool | Website | Retrieved 28/10/22 from https://wireframe.cc/

Github Desktop, a collaborative version control and backup software | Application | Retrieved 28/10/22 from https://desktop.github.com/

Qt, a graphical extension of C++ | Application | Retrieved 28/10/22 from https://www.qt.io/

Visual Studio Code, a lightweight programming IDE | Application | Retrieved 28/10/22 from https://code.visualstudio.com/

Backlog, a collaborative software debugging tool | Application | Retrieved 28/10/22 from https://nulab.com/backlog/

Jira, a lightweight collaborative software debugging tool | Application | Retrieved 28/10/22 from https://www.atlassian.com/software/jira

GitHub Repository, a collaborative version control and backup software | Application | Retrieved 25/11/22 from https://github.com/whitehatcat-JE/software-tracker