2207-BSE

Implementation – Project Description

CS106.2

Alexander Craig, Oliver Anders Grönkrans, Alexander Legner, Liam Konise

Document Outline

Table of Contents

Table of Figures

Table of Tables

No table of tables.

Intentionally Blank

Project Description

Our group is tasked with proposing a Help desk ticketing system to the class.

Goals and Objectives

What are the goals and objectives of a help desk ticket tracker?

The definitive goal of a help desk is to provide resolutions for user requests as efficiently as possible. Help desks are designed to be reactive in nature and are often considered a subset of the service desk. A good help desk strategy is to offer a centralized resource to address service disruptions and provide end-user service support.

Overview of product system requirements:

To create a system which allows users to register a ticket categorized as support or incident, then gather the details including but not limited to category, urgency, user details and tags. The system will then issue the ticket with a ticket number and forward all information to a designated group or agent for resolution. At any point a ticket should be updateable and allow users to track its status using the associated ticket number. Users should be able to search for specific incident details and provide formative feedback for the resolution of a submitted ticket.

Design Proposal

Our Goals Design Proposal:

- The team's responsibilities, constraints and objectives are outlined along with the deadlines for each task.
- An in-depth research document is collated including target audience, business wants/needs and user scenarios to ensure our product fits an example client.
- Requirements are collated within an SRS document with a minimum of 10 use case diagrams.
- An array of sketches is created, and user tested before Lo-fi and Hi-fi prototypes are created.
- Elements and functions of all included prototypes are user tested.
- A clear and concise presentation is made with all relevant information.

Our Objectives Design Proposal:

10/5/23 - Goals/Objectives, Timeline/Project constraints, Team members and responsibilities.

Outlining the overall goals, constraints, and responsibilities of each member within the group ensures the team is working in unison and understand what steps need to be taken at any given time to complete the project on time and to a satisfactory level. We decided to prioritise completing this part of the project quickly to ensure we had enough time towards the end of the project to update and modify our ideas with the research we will have collected.

17/5/23 - Functional/non-functional requirements, Interviewing, Target audience, Stories/Scenarios, Requirements classification/organisation/prioritisation/negotiation, SRS documentation, requirement validation.

Gathering the data on all our requirements and target audience is essential to ensure our project is on track to delivering the best system in accordance with our initial goals and the project brief. This research will also help us make informed decisions later in the project as to what changes will be made and why.

26/5/23 - Sketches Lo/Hi-fi, Screen layout, Main/secondary windows, Functionality, Elements, User testing.

Used to gain a full understanding of how the result will look and feel, completing Lo-Fi sketches and conducting user testing through each stage and change will give us a deeper understanding of what needs to be done and where our initial design ideas fall short. Once Lo-fi's are completed we will work on Hi-fi's and conduct another set of user tests before focusing on the desired elements and functionality we want to incorporate into our design.

2/6/23 - Presentation.

The due date for this project is the 2nd of June and we are planning to have a minimum of 3 days leading up to this date to make our final changes with the information we have collected and make any cuts that we believe from our user testing need to be done. Finally, a presentation will be collated with all relevant information to be presented and marked.

Product Implementation

Our Product Goals

- Customers find it easy to navigate the help desk and submit queries/incidents.
- Queries/incidents are trackable and updateable at any time.
- Tickets are forwarded to the relevant groups for resolution.
- Clients are updated with relevant information regarding their ticket's status.
- Ticket information is saved for later use/reference.
- Clients can provide feedback to help improve existing problems.

Our Product Objectives

An adequate spaced menu will be created, providing a selection of activities including new submission, existing submission, and search.
When an incident or submission is created users will be provided with a ticket number allowing for easy access to existing submission, each ticket will have an update feature accessible by both the original user and the technician assigned to the ticket allowing both the update the information if required.
When a ticket is created it will automatically assign it to the next available technician with the least amount of work currently on hand at any given moment.
All information gathered when an incident is created will be saved for later use. This information should still be available to be changed later.

After a user has submitted an incident report and it has been actioned by a technician, they will be provided with the opportunity to provide feedback on their experience.

Timeline and Project Constraints

Group Project Proposal

Timeline

To accommodate for the short timeline, we have divided the research phase into four different sprints, each one building upon the previous phase. We have not specified dates for individual tasks, but rather for the entire sprint. This is to accommodate for issues and blocks that may arise.

Constraints

An adequate spaced menu will be created, providing a selection of activities including new submission, existing submission, and search.

When an incident or submission is created users will be provided with a ticket number allowing for easy access t

We also have a strained timeline of four weeks per stage, the initial stage being the proposal formation, and the second, final stage being the implementation. To work within this, as specified in the timeline, we have divided work into scrum sprints. This will allow us to accommodate for issues while keeping productivity up.

Group Project Implementation

Timeline

As with the proposal timeline, we plan on using sprints to organize the work for the implementation. This way we can create a framework upon which we can add features as we go along, improve quality and documentation as time allows, and gives us flexibility in testing different available routes.

Constraints

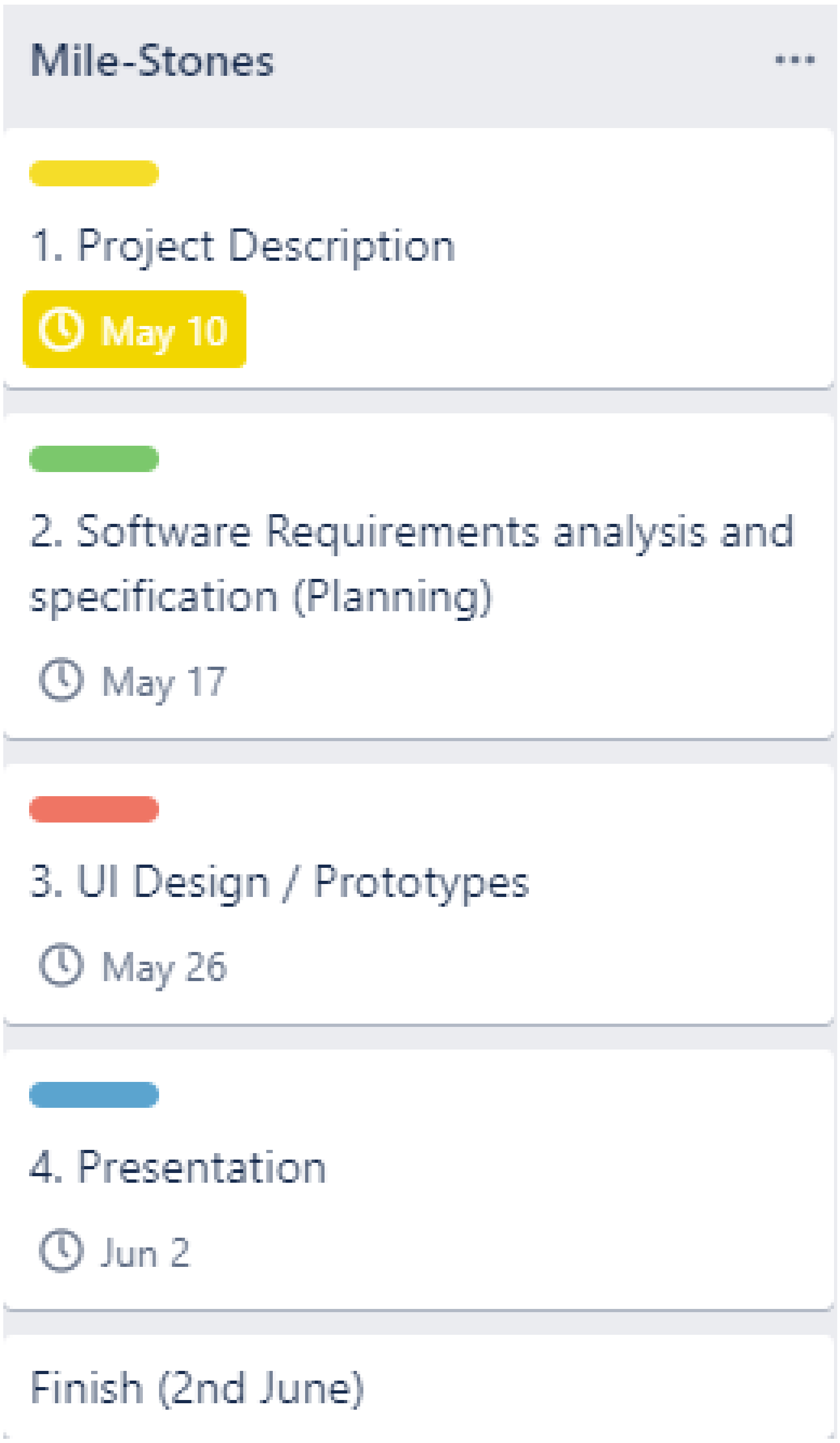
An uncertainty that must be navigated is the lack of previous experience in the planned frameworks. This means both that there is research required, and room for potential issues and blocks. If we implement a feature that later needs reworking as we learn the frameworks and find that it does not work in the set context, there must be time and resources available for it. A rough overview of how the sprints may be structured could be as follows:

- Brief research sprint to get familiar with the frameworks.
- Long implementation sprint where the groundwork is laid down.
- Long sprint to implement secondary functions.
- Brief quality improvement sprint.
- Finally, a documentation and refactoring sprint.

Note that this structure is just an example of how we may approach it, and not representative of how we currently intend to approach the implementation.

The final major constraint is the lack of financial freedom, which means that any products that could increase productivity and overall product quality may be locked behind license costs. With that said, a lot of software suites offer free student access, which means we may be able to use productivity tools that we may not have been able to otherwise.

Phases Overview



Mile-Stones

1. Project Description

May 10

2. Software Requirements analysis and specification (Planning)

May 17

3. UI Design / Prototypes

May 26

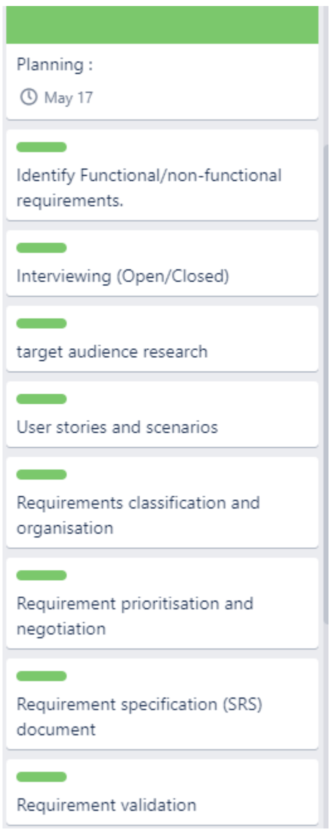
4. Presentation

Jun 2

Finish (2nd June)

Figure 1: Screen snippet of scrum board, showing when different phases are planned to finish.

Implementation


Planning :

May 17

Identify Functional/non-functional requirements.

Interviewing (Open/Closed)

target audience research

User stories and scenarios

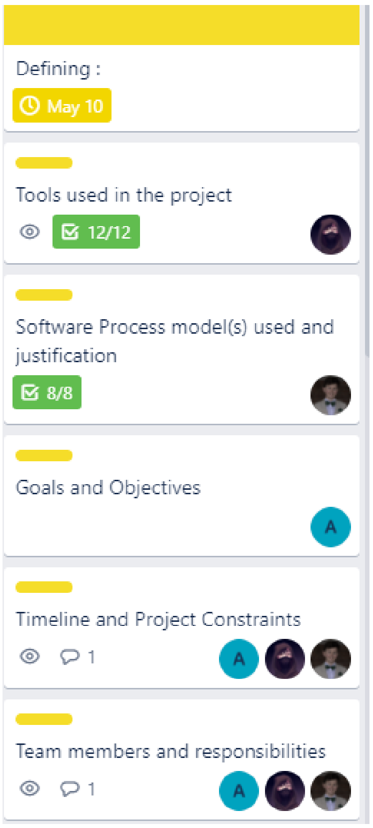
Requirements classification and organisation

Requirement prioritisation and negotiation

Requirement specification (SRS) document

Requirement validation

Figure 2: Screen snippet of the initial phase, where the requirements are defined for the project.


Defining :

May 10

Tools used in the project

12/12

Software Process model(s) used and justification

8/8

Goals and Objectives

Timeline and Project Constraints

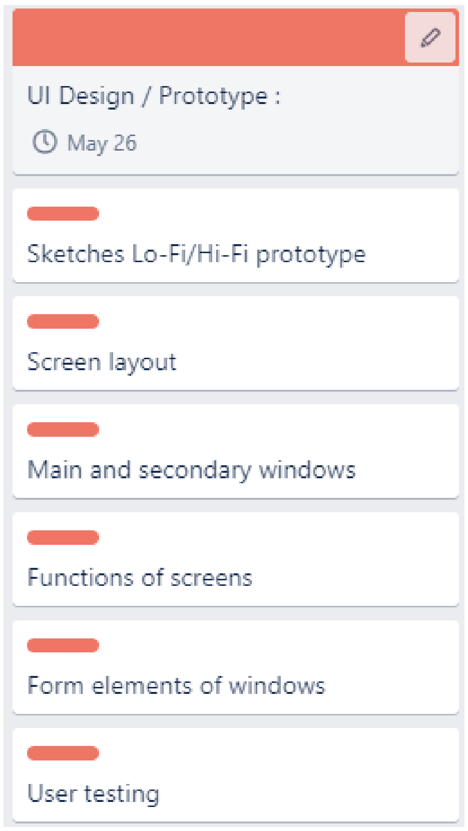
1

Team members and responsibilities

1

Figure 3: Screen snippet of the second phase outlining the SRS Document Requirement

Implementation



UI Design / Prototype :

May 26

Sketches Lo-Fi/Hi-Fi prototype

Screen layout

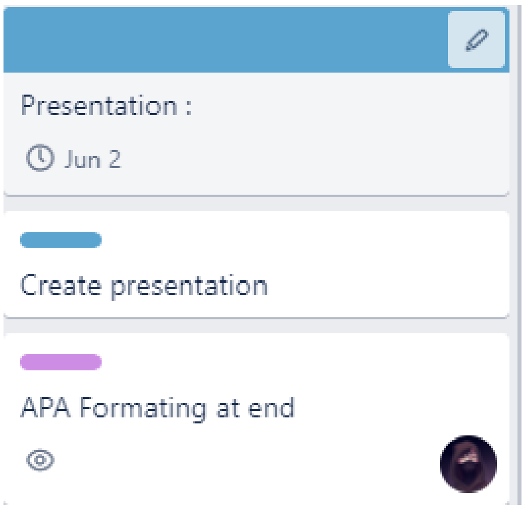
Main and secondary windows

Functions of screens

Form elements of windows

User testing

Figure 4: Screen snippet of third phase which consists of Research and UX



Presentation :

Jun 2

Create presentation

APA Formating at end

Figure 5: Screen snippet of any other relevant work outlined.

Team Members and Responsibilities

Team Members:

Our team consists of four team members:

- Oliver Grönkrans
- Alexander Craig (Alex C)
- Alexander Legner (Alex L)
- Liam Konise

Main responsibilities

Scrum Master: Alex C

The scrum master is responsible for keeping the team on task within our software process model (Agile - as explained later) and keeping the development process streamlined.

Product Owner (Representative): Oliver

The product owner is responsible for keeping the product in line with what the stakeholder is wanting. They are responsible for regular meetings with the stakeholders and ensuring what the team is developing is the same as the plan approved by the stakeholder.

Back-end Coordinator: Alex L

The back-end coordinator is responsible for the development of back-end related code include its design, documentation, and merge conflicts.

Technical Artist Coordinator: Oliver

The technical artist coordinator is responsible for coordinating with the front-end and back-end coordinator to ensure the code follows the proposed UML code design documents. They are also responsible for creating the bridge between front-end and back-end to ensure the back-end code works correctly as intended with the front-end UI/UX.

Front-end Coordinator: Alex C

The front-end coordinator is responsible for the UI layouts and UX of the product. From prototypes to the product the front-end coordinator is responsible for anything relating to UI/UX.

Product Analyst: Alex L

The product analyst is responsible for research, user testing and feedback for the prototype and product. They are responsible for ensuring the product is created with satisfaction of end users and clients that will be using the product through extensive research and user testing.

Delegate: Liam K

The delegate is responsible for all the tasks delegated to him by the scrum master and other team members that need assistance with tasks. These will be less important tasks but will still be a crucial part of the team.

** Liam was added to the project as of the 5/11/2023 – after the completion of the Document Outline. Responsibilities remaining were limited, so we made a role for the new team member to enhance the team’s harmony and partnership to ensure that the project will be completed to a high standard by the deadlines.

Shared Responsibilities
** Due to having such a small team, all team members will be doing parts of all the roles. The main responsibilities are there to streamline the process and ensure that each team member can be held accountable for the products development. We will work together no matter what role we find ourselves in to create the product effectively with the small team we have.

Software Process models

Agile

Overview

History
The agile methodology was launched in 2001 by 17 technologists (Sacolick, 2022), with the intent to improve project management. This has since widely replaced the waterfall model and lean model. The agile model was favoured early on as many start-ups work in smaller teams, often meaning they seldom have time to optimize or refactor code in one go, instead focusing on functionality, and iteratively improving quality, documentation, etc (Beck et al., 2001). This meant that products could be developed in shorter development cycles compared to the traditional waterfall approach (Beck et al., 2001).

Functionality
Agile focuses on small, workable increments instead of a colossal launch where one must trust in all components coming together and fulfilling their purpose as intended. To achieve this the team has specific roles, which generally follows a similar structure to the following -

- Product owner (Sacolick, 2022) – Acts as link between stakeholders or owners, and the development team, distilling insights, and ideas on the stakeholders' behalf to shape a product vision.
- Team lead (Sacolick, 2022) - Varying responsibilities, but they usually estimate stories (Simplilearn, 2023) (estimated explanations of specific software features), planning implementation details in conjunction with the team, and works with the product owner to agree on architecture and functionality criteria.
- Scrum master (Sacolick, 2022) – Overviewing the agile process, coaching team members to allow them to better work within the structure, showing how to use tools and agile processes, and keeping track of the agile team velocity (Sacolick, 2013) (which is determined by the number of stories and story points committed to each agile sprint). Their main responsibility can be described as keeping the team committed to and working within the agile method.
- Analyst (Sacolick, 2022) – Usually documents user stories, reviews progress and research findings.

These roles can flow into one and another, and it is normal for team members to have multiple roles in small teams. The result of using agile is a flexible approach that gives headroom for errors and issues that arise during development. While agile is a framework to structure a team's workflow, there are different implementations, mainly scrum and Kanban (Rehkopf, 2023).

Scrum

Scrum relies on organizing work in sprints, which normally last one to two weeks (Rehkopf, 2023). Scrum consists of regular meetings (also called scrum ceremonies or scrum rituals), where the teams can plan sprints, discuss daily goals, escalate issues, and occasionally demo progress to keep the product owner updated on the project status (Rehkopf, 2023). Kanban

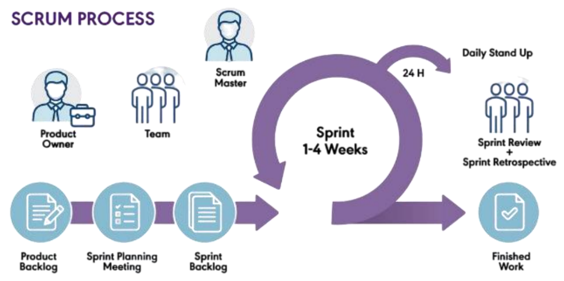


Figure 6: Visualization of scrum. (PM Partners, 2023).

Kanban

Kanban is a simple implementation and quite simple to explain, as it is a fan-in and fan-out process (Intel Corporation, 2017), consisting of funnelling stories in the workflow until they are finished (Sacolick, 2022). This limits the number of tasks assigned to team members at any given time, thus in theory maximizing output since team members will focus on finishing two task instead of focusing on multiple stories at the same time (Radigan, 2023).

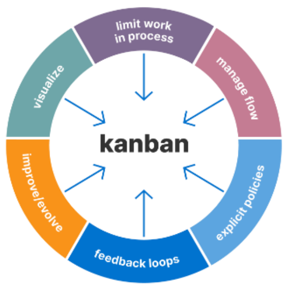


Figure 7: Visualization of the kanban model. (Aha!, 2023).

Lean

Lean Software Development (LSD) is an alteration of the lean philosophy, which has its roots in the manufacturing lean philosophy, which was created by Toyota in the 1940-1950's to minimize waste as a response to the weakened post-war economy (Womack & Lean Enterprise Institute, 2023). LSD at its core works by identifying the value clients are seeking and minimizing steps that are not strictly required to achieve this value (Lutkevich & Silverthorne, 2021). For LSD to work efficiently it requires strong documentation, to allow identifying of waste in each iteration (ProductPlan, 2021). Examples of waste can be bureaucratic processes, excessive tasks, unnecessary features, and so on (Lutkevich & Silverthorne, 2021). When using LSD, it is desirable to delay committing to irreversible decisions, for as long as possible (Lutkevich & Silverthorne, 2021). This means that developers can test different options, learn what works and what clients wants, as well as seeing how the market develops during the development cycle. This means developers can implement functionality as late as possible to avoid having to rework entire systems (Lutkevich & Silverthorne, 2021).

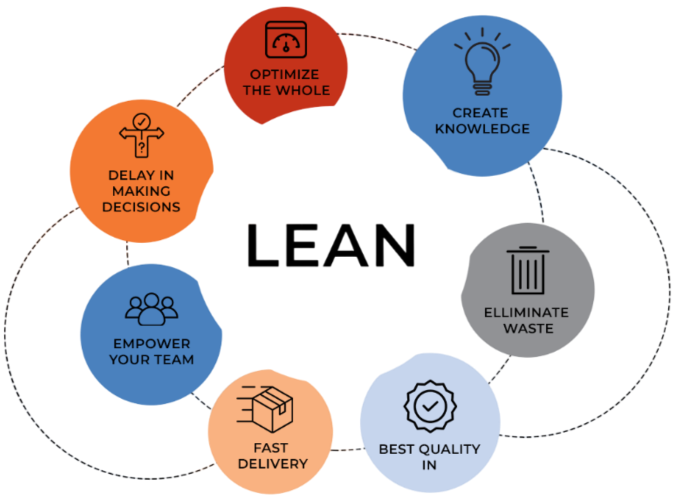


Figure 8: Visualization of the lean model. (Adam, 2021).

Waterfall

Overview

History

Established in 1970 by Winston Walker Royce, the waterfall model was one of multiple propositions in his research paper *Managing the Development of Large Software Systems* (Royce, 1970). It was made as a quick, progressive model where a given project is divided into clear steps, where each step is laid out clearly from the beginning. Though it may work for smaller projects, it quickly becomes risky and prone to failure (Royce, 1970). Royce himself suggests an iterative evolution of the waterfall model for it to work.

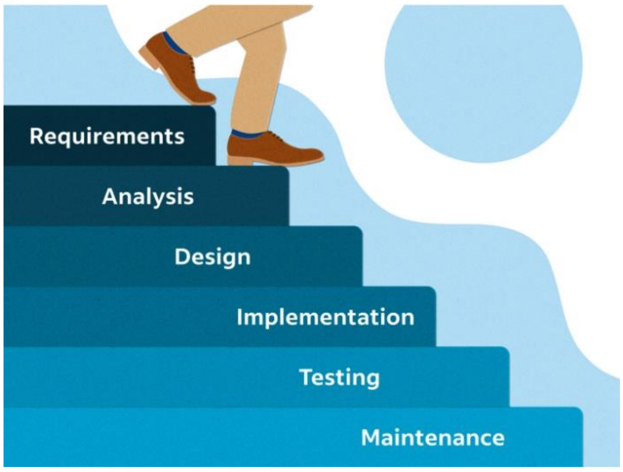


Figure 9: Visualization of the waterfall model. (Indeed Editorial Team, 2023).

Functionality

The waterfall model builds on a clear step-by-step process, and thus heavily relies on clear and known specifications and requirements ahead of development (Hoory, 2022). This structure of defining a plan before development means there are little to no supervision needed of the plan along the way, which may work for smaller projects in small teams, but it leaves little overhead for issues that may arise. It may be considered the easiest approach from a planning perspective, which is the main draw of this model.

Rapid Application Development

Overview

History

Conceived in the 1980's, Rapid Application Development (RAD), is a well-established software process model (Kissflow, 2023). It was created as many other project management models are built on manufacturing where a team works with finite resources, whereas in software engineering it is not confined to the same restraints (Kissflow, 2023). It meant that applications could be developed in drastically short times and with high flexibility, compared to other approaches such as the waterfall model.

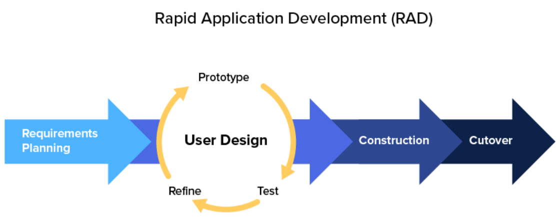


Figure 10: Visualization of the Rapid Application Development model. (Kissflow, 2023).

Functionality

RAD builds on high flexibility, where requirements can change, and is highly dependent on constant feedback (Kissflow, 2023). It works by prioritizing creating a functional model in the quickest possible way but is therefore also heavily reliant on talented team members (Kissflow, 2023). By prototyping and user-testing until a satisfactory model has been made before finalizing the product it means that a team can quickly make changes and test a variety of solutions before committing to the final product (OutSystems, 2023).

Process Model reasoning

As we are a small team with a short timeframe, the scrum workflow will allow us to make a working framework for the project, and iteratively improve functionality and documentation as time allows. This means we can focus on core functionality for a proof of concept, and iteratively test our solutions and components before committing to specific story points or ideas. The main reason we chose agile over waterfall or lean is that we are working in frameworks which we have not used previously, thus issues are likely to arise throughout the process. Having an iterative approach means we can field test our application throughout and make sound decisions and changes if we reach any blocks (Hoory, 2022).

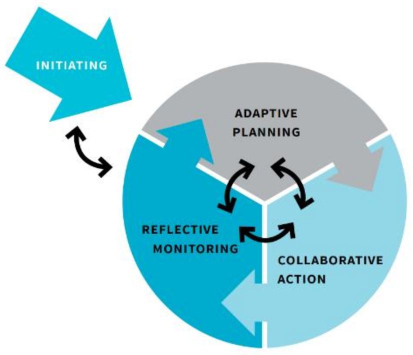


Figure 11: Visualization of process modelling. (Wageningen University & Research, 2023).

Tools Used in Project

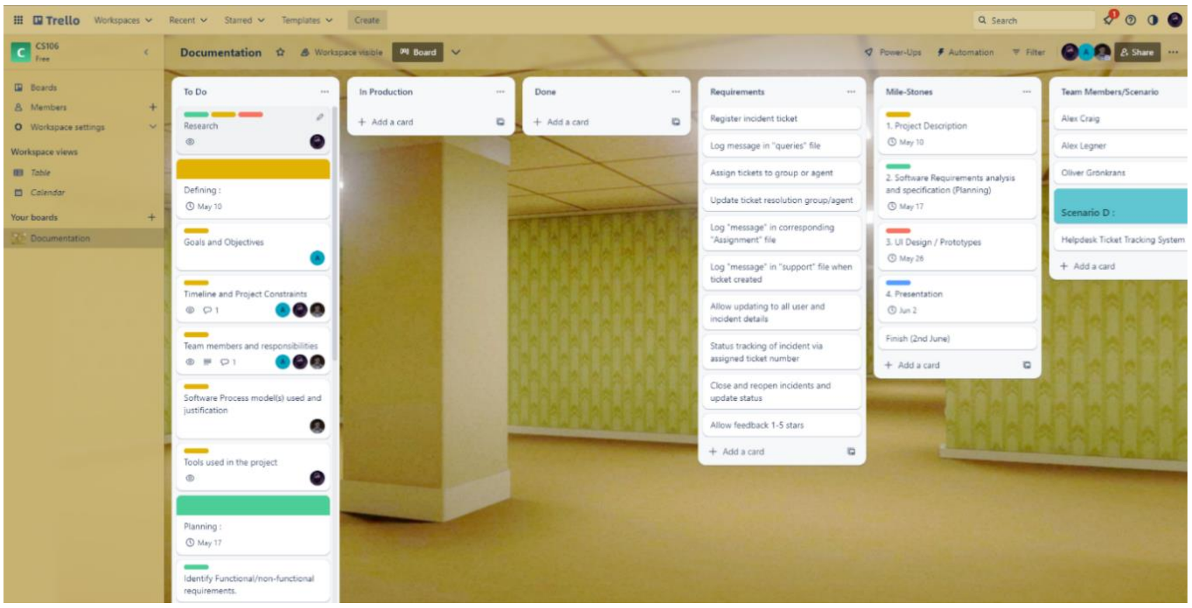
Overview

There are many tools out there that can be used and be considered useful in this project for collaboration. However, we will mainly use tools that we all know how to use. This is because we don't want to spend more time learning how to use some tools during this project and instead focus on more important parts of the project. The tools we will be using are listed below:

- Trello
- Microsoft Word
- Microsoft Teams
- Visual Studio 2019
- GitHub
- GitHub Desktop
- Figma
- Miro
- C# – Programming Language
- XAML – Programming Language
- XML – Programming Language
- .NET Framework

Trello

Trello is an agile/Kanban tool which allows us to visualise our work and progress. We selected Trello because it was a familiar tool to us and is also a free tool which the whole team can use so it will not add to the project's cost. We knew how to effectively utilise it to keep the team on track of the project's timeline. It is a visual way for the team to know when tasks are due, when tasks are being completed, and completed tasks.



To Do
- Research
- Defining : May 10
- Goals and Objectives
- Timeline and Project Constraints 1
- Team members and responsibilities 1
- Software Process model(s) used and justification
- Tools used in the project
- Planning : May 17
- Identify Functional/non-functional requirements.

In Production

Done

Requirements
- Register incident ticket
- Log message in "queries" file
- Assign tickets to group or agent
- Update ticket resolution group/agent
- Log "message" in corresponding "Assignment" file
- Log "message" in "support" file when ticket created
- Allow updating to all user and incident details
- Status tracking of incident via assigned ticket number
- Close and reopen incidents and update status
- Allow feedback 1-5 stars

Mile-Stones
- 1. Project Description May 10
- 2. Software Requirements analysis and specification (Planning) May 17
- 3. UI Design / Prototypes May 26
- 4. Presentation Jun 2
- Finish (2nd June)

Team Members/Scenario
- Alex Craig
- Alex Legner
- Oliver Grönkrans
- Scenario D :
- Helpdesk Ticket Tracking System

Figure 12: Screenshot of Trello Example

Microsoft Word

We selected Microsoft Word because it was a familiar tool to us. Microsoft Word is also a tool which the entire team has access to already so there will not be an added cost to the project because we do not need to pay for it again. We will use this software to create any documentation required for the project. We primarily picked Microsoft Word over something like Google Docs because it was a software, we all had access to while we worked on the project. It has an online auto save feature which allows us all to work on the same document at once and has extra features than Google Docs which allows to create these documents faster, in more detail and in more quality. This will improve the team’s ability to work together to document the process efficiently and effectively.

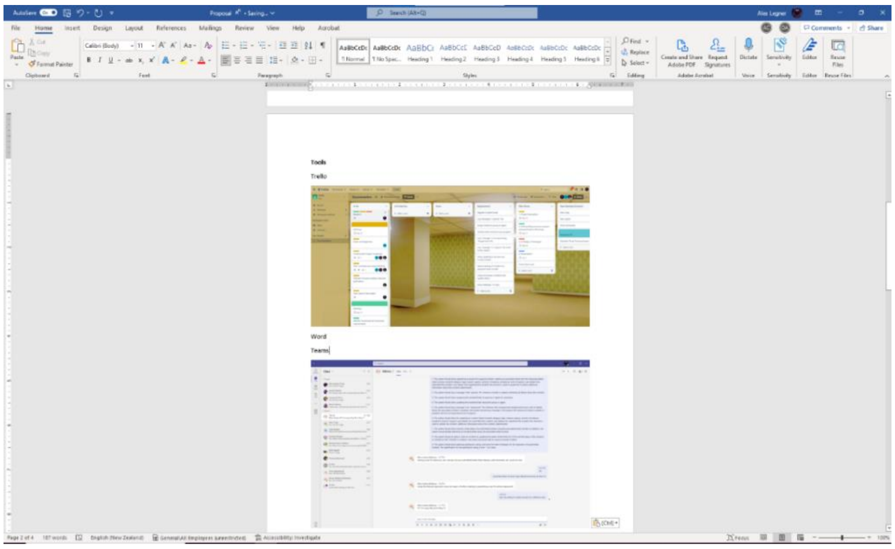
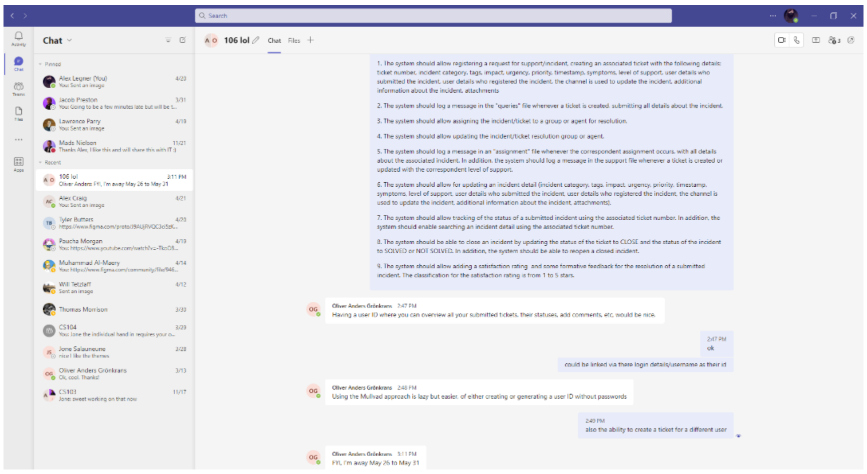


Figure 13: Screenshot of Word Example

Microsoft Teams

Microsoft Teams is a messaging system which allows for voice/video calls, messaging, group messaging and many other features not listed. We selected Microsoft Teams because it was a familiar tool to us. Microsoft Teams is also a tool which the entire team has access to already so there will not be an added cost to the project because we do not need to pay for it again. We will use this to discuss and record anything project related. It also helps us keep a record of anything in case we need to look back and see what happened. This also allows us to share any files and links to further increase our collaboration effort.



1. The system should allow registering a request for support/incident, creating an associated ticket with the following details: ticket number, incident category, tags, impact, urgency, priority, timestamp, symptoms, level of support, user details who submitted the incident, user details who registered the incident, the channel is used to update the incident, additional information about the incident, attachments

2. The system should log a message in the "queries" file whenever a ticket is created, submitting all details about the incident.

3. The system should allow assigning the incident/ticket to a group or agent for resolution.

4. The system should allow updating the incident/ticket resolution group or agent.

5. The system should log a message in an "assignment" file whenever the correspondent assignment occurs, with all details about the associated incident. In addition, the system should log a message in the support file whenever a ticket is created or updated with the correspondent level of support.

6. The system should allow for updating an incident detail (incident category, tags, impact, urgency, priority, timestamp, symptoms, level of support, user details who submitted the incident, user details who registered the incident, the channel is used to update the incident, additional information about the incident, attachments).

7. The system should allow tracking of the status of a submitted incident using the associated ticket number. In addition, the system should enable searching an incident detail using the associated ticket number.

8. The system should be able to close an incident by updating the status of the ticket to CLOSE and the status of the incident to SOLVED or NOT SOLVED. In addition, the system should be able to reopen a closed incident.

9. The system should allow adding a satisfaction rating and some formative feedback for the resolution of a submitted incident. The classification for the satisfaction rating is from 1 to 5 stars.

Oliver Anders Grönkrans 2:47 PM
Having a user ID where you can overview all your submitted tickets, their statuses, add comments, etc, would be nice.

2:47 PM
ok

could be linked via there login details/username as their id

Oliver Anders Grönkrans 2:48 PM
Using the Mullvad approach is lazy but easier, of either creating or generating a user ID without passwords

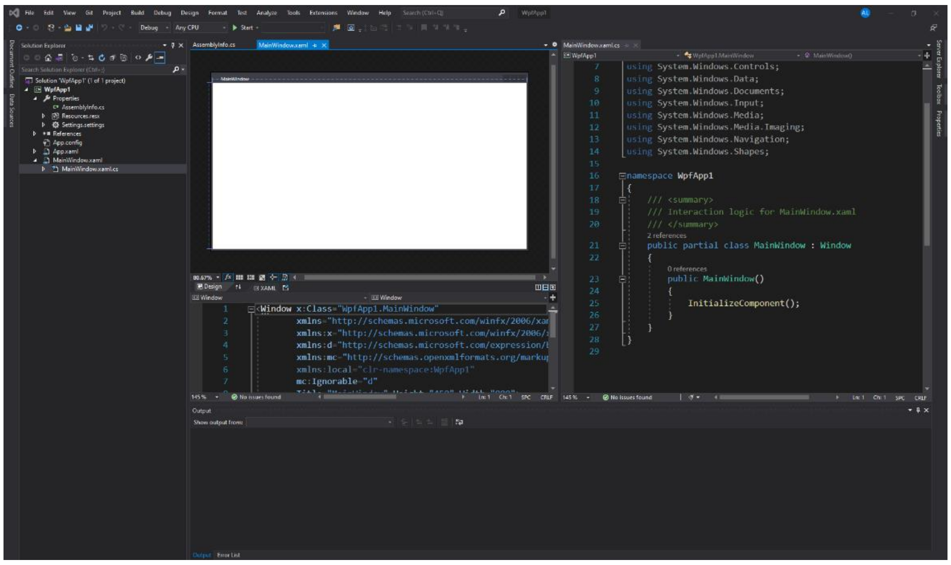
2:49 PM
also the ability to create a ticket for a different user

Oliver Anders Grönkrans 3:11 PM
FYI, I'm away May 26 to May 31

Figure 14: Screenshot of Teams Example

Visual Studio 2019

Visual Studio 2019 is IDE (Integrated Development Environment) which is natively used to develop WPF applications using their .NET Framework. We selected Visual Studio 2019 because it was a familiar tool to us. Visual Studio 2019 is also a free tool which anyone can use so it will not add to the project's cost. Visual Studio 2019 is an IDE we are all familiar with so programming and bug-fixing will be easy to do for the team when it comes to coding the application. It also has GitHub integration and is easy to use for large scale projects. It also natively supports WPF development which is what we will be using to create a GUI for our project later.



```
<Window x:Class="WpfApp1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xar
        xmlns:x="http://schemas.microsoft.com/winfx/2006/
        xmlns:d="http://schemas.microsoft.com/expression/
        xmlns:mc="http://schemas.openxmlformats.org/markup
        xmlns:local="clr-namespace:WpfApp1"
        mc:Ignorable="d"
```

```
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace WpfApp1
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}
```

Figure 15: Screenshot of Visual Studio Example

GitHub

GitHub is a file management software which allows multiple developers to write code and merge them together quickly to make them work. This also allows us to access any part of the project in case we need to recover code/files at any time. We selected GitHub because it was a familiar tool to us. GitHub is also a free tool which anyone can use so it won't add to the project's cost. All of us are familiar with GitHub and know how to use it quickly and effectively on a large-scale project. GitHub also allows to track our progress by allowing us to comment what changes we added to the project.

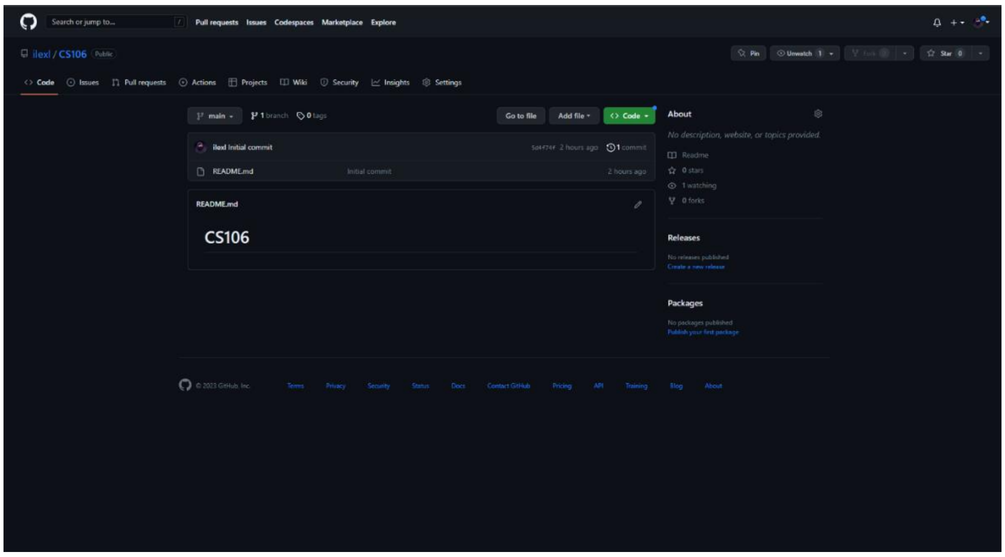
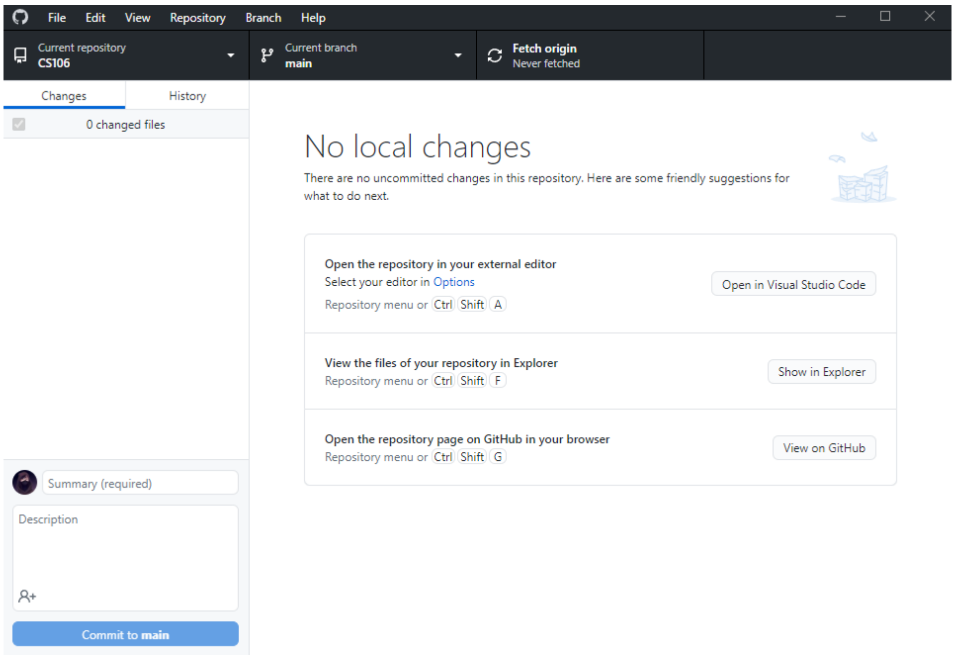


Figure 16: Screenshot of GitHub Example

GitHub Desktop

GitHub Desktop is a software to easily use GitHub. It has other features like showing what changes were made and makes uploading or reverting changes easier than using a console. We selected GitHub because it was a familiar tool to us. GitHub is also a free tool which anyone can use so it will not add to the project's cost. This will help speed up development as a team by allowing collaboration during the coding stage of the project.



No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Open the repository in your external editor
Select your editor in Options
Repository menu or Ctrl Shift A
Open in Visual Studio Code

View the files of your repository in Explorer
Repository menu or Ctrl Shift F
Show in Explorer

Open the repository page on GitHub in your browser
Repository menu or Ctrl Shift G
View on GitHub

Figure 17: Screenshot of GitHub Desktop Example

Figma

Figma is a design prototyping software which is limited in features but easily allows anyone to mock-up designs and prototypes. These prototypes are more focused on the UI elements and not functionality but works well to show the UX and design for a project quickly and effectively. We selected Figma because it was a familiar tool to us. Figma is also a free tool which anyone can use so it will not add to the project's cost. We also chose Figma to allow the team to prototype together at the same time. This would allow everyone to show other team members their ideas and quickly mock up prototypes of the design for the project.

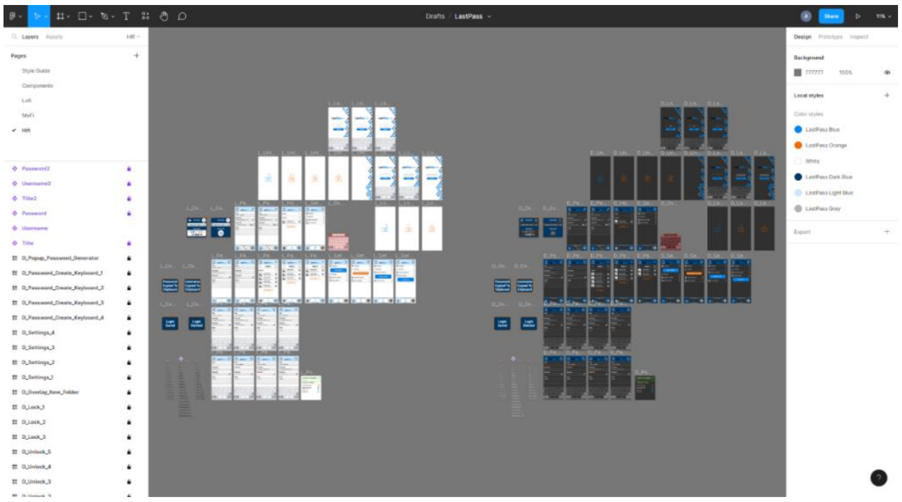


Figure 18: Screenshot of Figma Example

Miro

Miro is a simple to use white board with lots of features designed for small teams to collaborate thoughts and express ideas. We selected Miro because it was a familiar tool to us. Miro is also a free tool which anyone can use so it won't add to the project's cost. We selected Miro so that we could do smaller brainstorming and express ideas to the team in a quick and simple way. It also allows for collaboration at the same time so expressing ideas quickly will be easy for the team with this software.

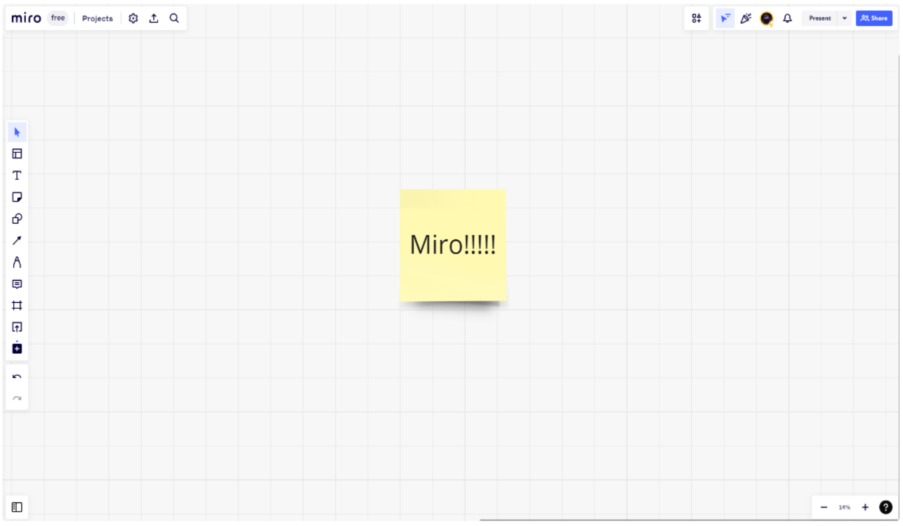
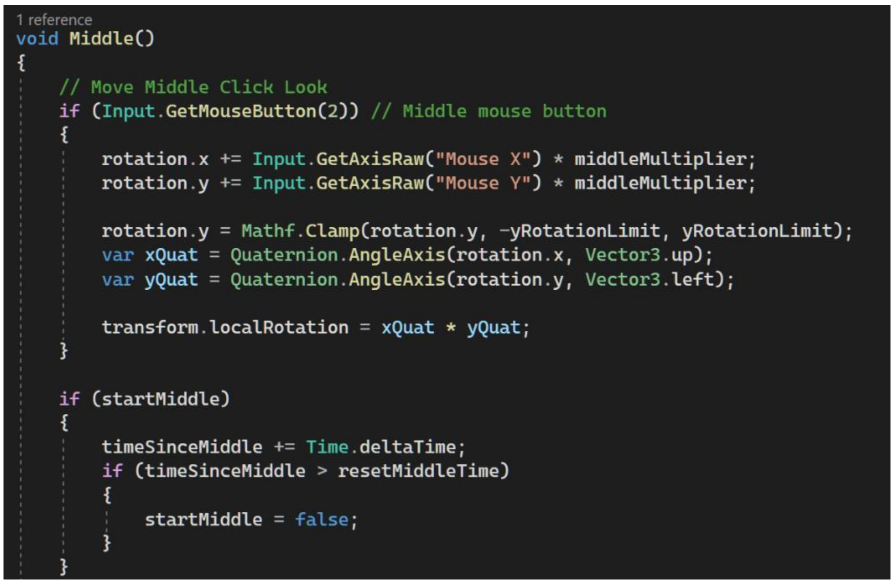


Figure 19: Screenshot of Miro Example

C# - Programming Language

C# ("C Sharp") is a high-level object orientated programming language used for many applications. The language has a framework made for it called .NET made by Microsoft which extends what C# can do. Although C# isn't well known by our entire team, we are familiar with C++ which is a hard level but similar syntax. We don't need to relearn concepts, just the specific syntax which isn't too hard. We will allocate time to learn this language so that it doesn't affect the project timeline. We also have a team member who first learnt to code in C# so there is always help within the team if needed. We are using C# because it allows us to create a GUI with XAML – WPF within Visual Studio 2019.

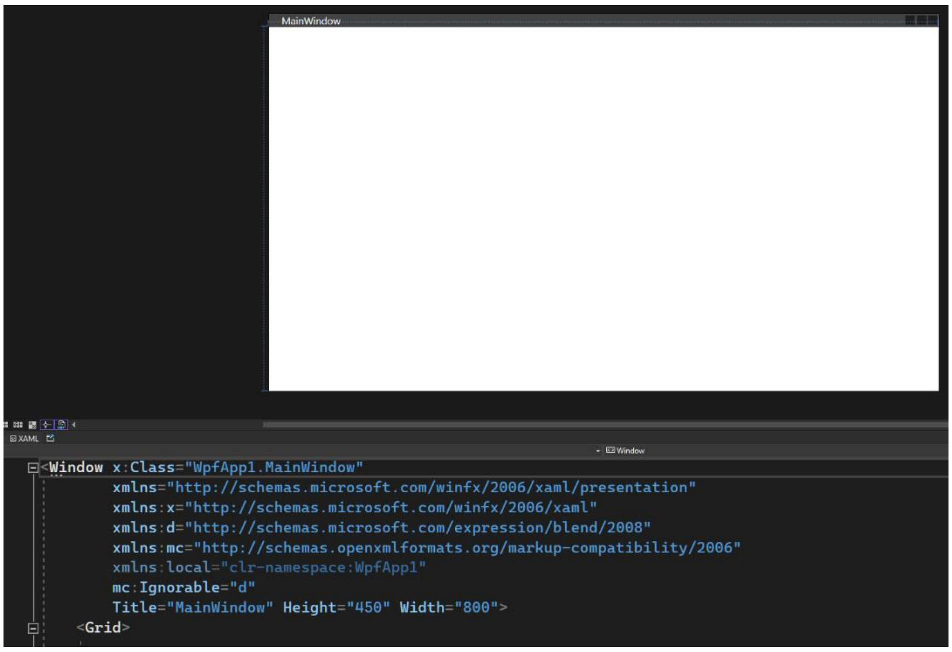


```
1 reference
void Middle()
{
    // Move Middle Click Look
    if (Input.GetMouseButton(2)) // Middle mouse button
    {
        rotation.x += Input.GetAxisRaw("Mouse X") * middleMultiplier;
        rotation.y += Input.GetAxisRaw("Mouse Y") * middleMultiplier;

        rotation.y = Mathf.Clamp(rotation.y, -yRotationLimit, yRotationLimit);
        var xQuat = Quaternion.AngleAxis(rotation.x, Vector3.up);
        var yQuat = Quaternion.AngleAxis(rotation.y, Vector3.left);

        transform.localRotation = xQuat * yQuat;
    }

    if (startMiddle)
    {
        timeSinceMiddle += Time.deltaTime;
        if (timeSinceMiddle > resetMiddleTime)
        {
            startMiddle = false;
        }
    }
```

Figure 20: Screenshot of C# Code Example for a Unity Script

WPF – XAML

WPF or Windows Presentation Foundation is UI Framework from the .NET Framework which can be coded in XAML and C#. The C# code is primarily used as backend whereas the XAML is used to code the front end. We can use both C# and XAML to code the front end. We code WPF primarily due to the client's requirements. They did give us the option to use other software, however we chose WPF in the end to keep the software within the client's expectations and requirements and because we are already using Visual Studio 2019 which allows for native and easy WPF and C# development.

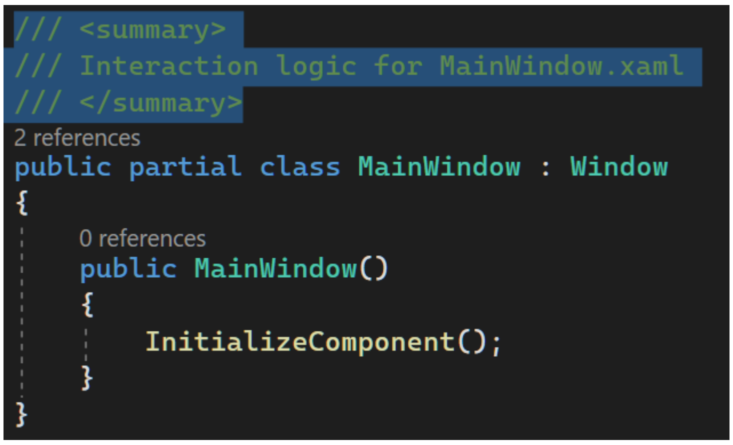


```
<Window x:Class="WpfApp1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:WpfApp1"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
    <Grid>
```

Figure 21: Screenshot of WPF Example using XAML

XML – Comments

XML not to be confused with XAML is a commenting based language which allows for more complex comments to effectively communicate what code does. This is supported within Visual Studio 2019 which the whole team is using, so we will use XML comments over classes and functions where practical to communicate what a class or function does. These comments only apply to C# as XAML doesn't allow for XML comments. We chose XML comments because it will help communicate to the team what code does more effectively. XML comments will not be used for variables because the C# comments will suffice when commenting on variables.

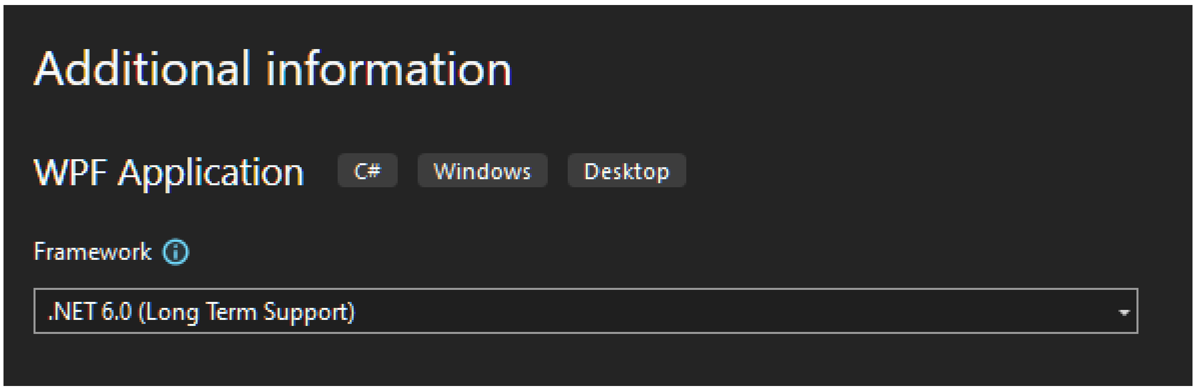


```
/// <summary>
/// Interaction logic for MainWindow.xaml
/// </summary>
2 references
public partial class MainWindow : Window
{
    0 references
    public MainWindow()
    {
        InitializeComponent();
    }
}
```

Figure 22: Screenshot of XML Comments Example

.NET Framework

The .NET framework made by Microsoft is a framework used for C# development and extends what the C# language can do. The .NET Framework can be used to develop console apps, GUI apps, ASP apps, windows services, web applications, database applications and more. We will be using the .NET Framework specifically for the WPF framework within the .NET framework to develop our application because we need to create a application with a GUI and this helps enable us to develop a GUI based application effectively within Visual Studio 2019. The .NET Framework also includes the CLR (common language runtime) which is the foundation of the .NET Framework. The CLR is responsible for code execution, memory management, thread management, security and more important code management related features which are maintained which regular security updates and bug fixes with the .NET Framework.



Additional information

WPF Application C# Windows Desktop

Framework

.NET 6.0 (Long Term Support)

Figure 23: Screenshot of .NET Framework in a WPF Application Example

References

Adam, J. (2021, October 27). Lean software development – parent or child of the Agile movement? *K&C*. https://kruschecompany.com/lean-software-development/

Adegeo. (2023, February 16). *What is Windows Presentation Foundation - WPF .NET*. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-7.0

Aha. (2023). What Is Kanban? Origin, Definitions, and Tips for Implementation. *Aha! Software*. https://www.aha.io/roadmapping/guide/agile/what-is-kanban

BillWagner. (n.d.). *C# docs - get started, tutorials, reference.* Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/csharp/

Everett, R. (n.d.). *Customer Support Software & Ticketing System | Freshdesk*. https://www.freshworks.com/freshdesk/helpdesk-ticketing/

fan-in and fan-out. (n.d.). (C) Copyright 2017. https://www.intel.com/content/www/us/en/programmable/quartushelp/17.0/reference/glossary/def_fan.htm

Figma: the collaborative interface design tool. (n.d.). [Software]. In *Figma*. https://www.figma.com/

Gewarren. (2023, March 30). *Overview of .NET Framework - .NET Framework*. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/framework/get-started/overview

GitHub Desktop. (n.d.). [Software]. In *GitHub Desktop*. https://desktop.github.com/

GitHub: Let's build from here. (n.d.). [Software]. In *GitHub*. https://github.com/

Hoory, L. (2022, August 10). Agile Vs. Waterfall: Which Project Management Methodology Is Best For You? *Forbes Advisor*. https://www.forbes.com/advisor/business/agile-vs-waterfall-methodology/

Indeed Editorial Team. (2023). A Complete Guide to the Waterfall Methodology. *Indeed Career Guide*. https://www.indeed.com/career-advice/career-development/waterfall-methodology

Jumpfactor. (2019, October 22). *IT Helpdesk Management Guide | Buchanan Technologies*. Buchanan Technologies. https://www.buchanan.com/it-help-desk-management/

Kissflow, Inc. (2023). Rapid Application Development (RAD) | Definition, Steps & Full Guide. *Kissflow, Inc*. https://kissflow.com/application-development/rad/rapid-application-development/

Lean Enterprise Institute. (2023, March 31). *A Brief History of Lean - Lean Enterprise Institute*. https://www.lean.org/explore-lean/a-brief-history-of-lean/

Lean Software Development. (2021, July 30). https://www.productplan.com/glossary/lean-software-development/

Loshin, P. (2022). help desk. *Customer Experience*. https://www.techtarget.com/searchcustomerexperience/definition/help-desk

Lutkevich, B., & Silverthorne, V. (2021). Lean software development. *Software Quality*. https://www.techtarget.com/searchsoftwarequality/definition/lean-programming

Manage Your Team's Projects From Anywhere | Trello. (n.d.). [Software]. https://trello.com/

ManageEngine. (n.d.). *The comprehensive guide to help desk software - ManageEngine*. ManageEngine ServiceDesk Plus. https://www.manageengine.com/products/service-desk/help-desk-software/what-is-help-desk-software.html

Managing the development of large software systems. (1970). In *praxisframework.org*. Winston W. Rovce. Retrieved May 16, 2023, from https://www.praxisframework.org/files/royce1970.pdf

Microsoft 365 - Subscription for Office Apps | Microsoft 365. (n.d.). [Software]. https://www.microsoft.com/en-nz/microsoft-365

OutSystems. (n.d.). *Rapid Application Development: Everything You Need to Know*. https://www.outsystems.com/glossary/what-is-rapid-application-development/

Pmadmin. (2023, February 15). *What is Scrum? | The Agile Journey with PM-Partners*. PM Partners. https://www.pm-partners.com.au/the-agile-journey-a-scrum-overview/

Process Model. (2021, December 1). MSP Guide. https://mspguide.org/process-model/

Rehkopf, B. M. (n.d.). *Kanban vs Scrum | Atlassian*. Atlassian. https://www.atlassian.com/agile/kanban/kanban-vs-scrum

Sacolick, I. (n.d.). *5 Ways to Improve Agile Team Velocity*. https://blogs.starcio.com/2013/07/5-ways-to-improve-agile-team-velocity.html

Sacolick, I. (2022a, April 6). *What is agile methodology? Modern software development explained*. InfoWorld. https://www.infoworld.com/article/3237508/what-is-agile-methodology-modern-software-development-explained.html

Sacolick, I. (2022b, April 8). *A brief history of the agile methodology*. InfoWorld. https://www.infoworld.com/article/3655646/a-brief-history-of-the-agile-methodology.html

Simplilearn. (2023). What Are Story Points in Agile And How to Estimate Them. *Simplilearn.com*. https://www.simplilearn.com/story-points-in-agile-article

The Visual Collaboration Platform for Every Team | Miro. (n.d.). [Software]. In *https://miro.com/*. https://miro.com/

Visual Studio: IDE and Code Editor for Software Developers and Teams. (2023). [Software]. In *Visual Studio*. https://visualstudio.microsoft.com/

XML Introduction. (n.d.). https://www.w3schools.com/xml/xml_whatis.asp