

User Oriented Remote Working Tool

Developed in Java

Ewan Donaldson

2618009

Jason Adair

April 2021

Dissertation submitted in partial fulfilment for the degree of
Bachelor of Science with Honors (Applied Computing)

Computing Science and Mathematics
University Stirling

1 Abstract

Remote working is an ever-increasing role that individuals are participating in. With the coronavirus pandemic resulting in a lockdown in most countries across the globe, employees are being asked to work from home in an effort to keep organizations open for business. These employees will therefore be required to work with a remote-working tool to be their most productive, but all this is at a cost. Workers that are quarantined will suffer from isolation as they lack in-person communication with others. They can also suffer from reduced visibility due to the comparison made between in-office workers and remote workers by superiors within the organization. These flaws in remote working result in the decline of the user's mental health and professional career.

This project aims to develop a remote working tool which helps the user in making sure they live a healthy work from home life by:

- Allowing the user to see charts showing their productivity, not compared to others but previous iterations of themselves.
- Keeping visual and audio communication channels open between users of the same company, thus allowing for meetings to be had.
- Allowing superiors to review and provide feedback on submitted work.
- Being able to understand what is required of you in terms of tasks to complete.
- Having an efficient method for the setup of tasks between individuals.
- Having a strong file system for users to access and deposit to.

This application will give the users the control over their own productivity. They will be able to gauge how effective they are within a set timeframe by comparing their current work rate against previous accounts. This prevents superiors making unfair comparisons between employees.

Research was conducted prior to the development of this project to discover what made a tool sufficient for remote working. Alternative remote working software was selected and used to understand which features would be required to create this type of software. It is important that the software developed, while fixing the issues most currently existing tools have, still serves the purpose of allowing users to easily working from home.

The output of this project is a fully functional remote working tool with the added benefits of being centered around whoever utilizes the software. This tool can be run on any windows computer as it is developed fully in Java.

2 Attestation

I am aware of and fully understand the University of Stirling academic misconduct policy, as well as the nature of plagiarism.

I can confirm that the work completed in this project is my original work which was accomplished during the time spent completing my dissertation, except for the following libraries.

- Drop Box – An open-source file system that is used to allow companies to share files between its users from home.
- JFreeChart – A Java library that allows for the use of charts.
- JCommon – A Java library that is used in conjunction with JFreeChart. It contains miscellaneous classes which supports dependencies.
- JDatePicker – A Java library with Java swing elements that grant the ability for dates to be chosen by the user.

Signature: Ewan Donaldson

Date: 26/02/2021

3 Acknowledgements

I am thankful for the time and effort my dissertation coordinator Jason Adair has spent helping me with my project inside of our arranged weekly meetings and out with them to. Jason has been very positive throughout the entirety of my project and has been great for bouncing ideas off. This has helped gain clarity in what I would like from myself. The time Jason has invested into me has not always been about the project, he would always make sure I was doing well out my project which helped on a more personal note. Thank you Jason for being there all the way through this project, your guidance is highly valued and greatly appreciated.

I would thank my family and friends for helping me through the tough times of the covid pandemic. Though they were not able to help on a technical level, their support was very valuable to me.

4 Table of Contents

1 Abstract.....	2
2 Attestation	3
3 Acknowledgements	4
4 Table of Contents	5
5 List of Figures	10
6 Introduction	12
6.1 Background and Context.....	12
6.2 Scope and Objectives	13
Stage 1: Remote Working	14
Stage 2: Isolation	14
Stage 3: Visibility	14
Stage 4: Communication	15
6.3 Overview of Dissertation	16
6.3.1 State-of-the-Art.....	16
6.3.2 Requirements	16
6.3.3 Design	16
6.3.4 Implementation	16
6.3.5 Testing	16
7 State-of-the-Art	17
7.1 Similar Tools	17
7.2 Remote Working Tools.....	17
GitHub.....	17
Microsoft Teams	17
Todoist	18
7.3 User Monitoring Software	18
Teramind	18
7.4 Issue Fixes	19
GitHub.....	19
Microsoft Teams	19
Todoist	19

Teramind	19
8 Appendix	20
8 Requirements	21
8.1 Functional Requirements	21
8.1.1 Error Handling	21
8.1.2 Display	21
8.1.3 Application Features	21
8.2 Non-Functional Requirements	22
8.2.1 Scalability	22
8.2.2 Accessibility	22
9 Design	23
9.1 Database Design	23
9.1.1 Database Tables	23
9.1.2 Normalization	26
9.1.3 Database Design Models	27
9.2 Software Design	28
9.2.1 Navigation Hierarchy	28
9.2.2 Components	29
9.2.3 UI Design	29
9.2.4 Authentication Login/Register	30
9.2.5 Colour Scheme	31
10 Implementation	32
10.1 Overview	32
10.2 User Account	32
10.2.1 Creation	32
10.2.2 Security Concerns	33
10.3 UI and its Components	34
10.3.1 Title Bar Panel	34
10.3.2 JPanel: Menu Panel	35
10.3.3 JPanel: Content Panel	36
10.3.3 JButton: Ping Feature	37
10.4 Task Feature	38
10.4.1 Overview	38

10.4.2 Layout	38
10.4.3 JPanel: Displaying Specific Tasks	39
10.4.4 JPanel: Adding New Task	39
10.4.5 Implementation Issues	40
10.5 Calendar Feature	41
10.5.1 Overview	41
10.5.2 How it Functions	41
10.5.3 JPanel: Calendar Controls	42
10.5.4 Responsive Components	43
10.5.5 Implementation Issues	43
10.6 Notifications Feature	44
10.6.1 Overview	44
10.6.2 JPanel: Create and Update Notification.....	44
10.6.3 Delete Notification.....	45
10.6.4 Error Handling	45
10.7 Productivity Feature.....	47
10.7.1 Overview	47
10.7.2 Libraries Used.....	47
10.7.3 JPanel: Controls Interface.....	47
10.7.4 Messages Chart.....	49
10.7.5 Time Working Chart.....	50
10.7.6 Implementation Issues	51
10.8 Chat Feature	52
10.8.1 Overview	52
10.8.2 JPanel: Friends List	52
10.8.3 JPanel: Message Display Panel	53
10.8.4 User Input.....	53
10.8.5 Update/message Checking.....	54
10.8.6 Error Handling	54
10.8.7 Implementation Issues	54
10.9 Drop Box Feature	55
10.9.1 Overview	55
10.9.2 Connection	55

10.9.3 Reading and Writing	55
10.9.4 JPanel: Interface.....	56
10.10 Feedback Feature.....	58
10.10.1 Overview	58
10.10.2 Jpanel: Interface	58
10.10.3 Edit Feedback Pop-Up	59
10.11 Settings Feature	60
10.11.1 Overview	60
10.11.2 Jpanel: Interface	60
10.11.3 File Storing	60
10.11.4 Error Handling.....	61
10.11.5 Implementation Issues	61
10.12 Dashboard	62
10.12.1 Overview	62
10.12.2 JPanel: Information Panel	62
10.13 Login and Register	63
10.13.1 Overview	63
10.13.2 JFrame: Login Form.....	63
10.13.3 JFrame: Register Form	63
10.13.4 Responsive Components	64
10.14 Database Class.....	65
10.14.1 Overview	65
10.14.2 Loading Database.....	65
10.14.3 Login Records	65
10.15 System Exit Class	66
10.15.1 Overview	66
10.15.2 Implementation Issues	66
11 Testing.....	67
11.1 Methods Used	67
11.2 Internal Testing	67
11.3 External Testing.....	71
Question 1	71
Question 2	71

Question 3	72
Question 4	72
Question 5	73
Question 6	73
Question 7	74
Question 8	74
12 Conclusion	75
12.1 Evaluation	75
12.2 Plans.....	76
12.2.1 Changes	76
12.2.2 Future Work	76
13 References	77
Appendix 1 – Testing Case	78

5 List of Figures

The list of numbered figures that are used throughout this write-up. The list should contain the figures in the order they appear in the report.

Figure 1:	Harvard Business Motivation Survey	Page 13
Figure 2:	User Table Structure	Page 23
Figure 3:	Task Table Structure	Page 24
Figure 4:	Statistics Table Structure	Page 24
Figure 5:	Login-Records Table Structure	Page 24
Figure 6:	Notifications Table Structure	Page 25
Figure 7:	Chat Table Structure	Page 25
Figure 8:	Database Normalization	Page 26
Figure 9:	Database Design Model	Page 27
Figure 10:	Application Navigation Topology	Page 28
Figure 11:	Main Application Wireframe	Page 29
Figure 12:	Login Form Wireframe	Page 30
Figure 13:	Register Form Wireframe	Page 30
Figure 14:	Colour Scheme	Page 31
Figure 15:	Database Connection with Register Account	Page 32
Figure 16:	Web Server User Accounts	Page 33
Figure 17:	Register Error Messages	Page 33
Figure 18:	Title Bar Object Creation and Initialization	Page 34
Figure 19:	Title Bar Interface (with ping bell)	Page 34
Figure 20:	Menu Interface	Page 35
Figure 21:	Button Action Listener (Displays Calendar Feature)	Page 36
Figure 22:	Card Layout Assignments	Page 36
Figure 23:	Task Feature Display	Page 38
Figure 24:	Action Listener for “Add New Task” Button	Page 39
Figure 25:	Specific Task Display	Page 39
Figure 26:	Adding New Task Display	Page 40
Figure 27:	Creating Calendar Object	Page 41
Figure 28:	Calendar Display	Page 41
Figure 29:	Calendar Controls Interface	Page 42
Figure 30:	Date Error Checking Example	Page 42
Figure 31:	Calendar Responsiveness Example	Page 43
Figure 32:	Notifications Feature Interface	Page 44
Figure 33:	Add New Notification Form	Page 44
Figure 34:	Update Notification Form	Page 45
Figure 35:	Delete Selected Notifications	Page 45
Figure 36:	Productivity Panel Controls	Page 47
Figure 37:	Radio Button Click Listener	Page 48
Figure 38:	Messages Sent Bar Chart	Page 49
Figure 39:	Time Working Pie Chart	Page 50
Figure 40:	Chat Feature Interface	Page 52
Figure 41:	View Messages Button Action Listener	Page 53

Figure 42:	Database Chat Table Records	Page 53
Figure 43:	Scheduled Executor Service	Page 54
Figure 44:	Runnable Method for Messages Panel Updates	Page 54
Figure 45:	File Explorer	Page 55
Figure 46:	Dropbox Connection	Page 56
Figure 47:	Dropbox User Interface	Page 56
Figure 48:	Feedback Feature Interface	Page 58
Figure 49:	Feedback Panel Interface	Page 58
Figure 50:	Edit Feedback Pop-Up Interface	Page 59
Figure 51:	Settings Page Interface	Page 60
Figure 52:	Dashboard Interface	Page 62
Figure 53:	Login Page Interface	Page 63
Figure 54:	Register Page Interface	Page 64
Figure 55:	Question 1 Results	Page 71
Figure 56:	Question 2 Results	Page 71
Figure 57:	Question 3 Results	Page 72
Figure 58:	Question 4 Results	Page 72
Figure 59:	Question 5 Results	Page 73
Figure 60:	Question 6 Results	Page 73
Figure 61:	Question 7 Results	Page 74
Figure 62:	Question 8 Results	Page 74

6 Introduction

6.1 Background and Context

The Coronavirus pandemic has caused a large number of companies across the globe to convert a large number of staff to remote workers. Before the pandemic occurred data from the Federal Statistical Office [1] stated that between 2013 and 2018 there was an increase from 18% to 24% in workers that started remote working. This figure has now jumped due to the pandemic so now 50% of people now work from home. As this been artificially enlarged it will likely subside to some degree. Although it will not revert back to normal as the Federal Statistics Office in the same study has found that 34% of workers will remain remote workers. This shows that due to the pandemic the number of remote workers will have increased by 10%. Due to these statistics its clear to see why remote working tools such as mine are required now more than ever.

Due to the spike in remote workers, there are more workers suffering from isolation. Isolation occurs when an employee suffers from being secluded from their colleagues for an extended period. An article published through the website hosted by the global media company “Forbes” [2] stated:

“Some symptoms of isolation include increased stress levels and bad decision making. For an employer, these are concerning characteristics for someone who has crucial responsibility. Unfortunately, being isolated also means these symptoms are difficult for employers to detect”.

This is true as working in a new and ineffective environment can often affect their work ability, which can result in a decreased visibility. The following quote by Professor Teresa Amabile from Harvard Business School [3] aids this in saying:

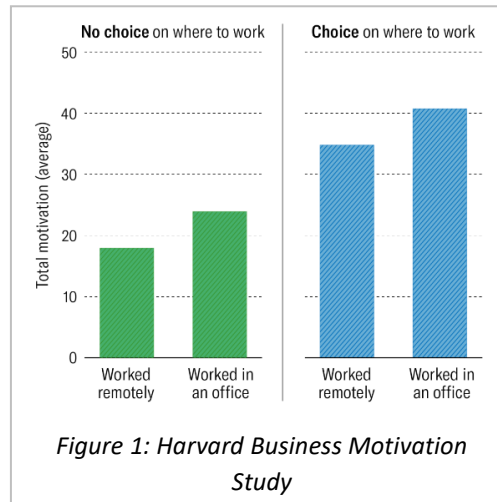
“People are more creative and productive when they experience more positive inner work life, including more positive emotions, stronger motivation toward the work itself and more positive perceptions of the organisation”.

The problem being tackled is the increase in the cases of isolation which is brought on by the number of stay-at-home workers increasing within multiple fields of work. Also, with the increasing number of jobs that can be completed by users anywhere in the world the demand for strong remote working tools is increasing. These tools are sought after more due to the coronavirus pandemic which has resulted in global lockdowns, because of this more people have decided to stay and work from home to keep the risk of personal infection to a minimum. A survey by Christopher Stanton, Zoe Cullen, Michael Luca from Harvard Business School [4] was released to 1800 employees from small and large businesses. The study indicates that at least 16 percent of people will decide to continue remote working after the pandemic has end. In the same study over a third of companies that had moved workers to remote working have stated they would keep workers at home after the pandemic has concluded. These stats prove that even though the pandemic will eventually disperse, on a global scale this will still have an impact on the number of people who will return to work resulting in higher demand for a strong remote working tool.

Decreased visibility is another large problem being tackled within this application. Aiming to ensure that employees are having their contributions to their company seen and rewarded. In being a remote worker it's common to become a hidden employee due of the limited interaction with superiors. This problem causes issues especially when remote workers are compared against employees who work from the office. This is because employees venturing into the office environment may seem more dedicated as they make the effort to travel. This face-to-face communication between an employee and employer can attribute

to good rapport, which can lead to a higher likelihood of promotions. This in person interaction is something that a remote worker will get very little of.

Lack of motivation is often a problem that occurs when workers make the switch to remote working. A study conducted by Harvard Business Review [5] in which 9700 United States workers answered questions about what motivates them to do work.



The study demonstrates that people agree there is a lack in motivation when remote working, no matter whether they chose to work there or not.

6.2 Scope and Objectives

The objective of this project is a remote working tool that allows its users to gauge their own effectiveness at work, while still being able to easily see and manage the tasks set by their employers. All the following features relate to the problems being addressed this application. Negating isolation through enabling communications and increasing an employee's visibility through the collection of statistics are the ways in which employees can be assisted in fighting these issues.

- Tasks View
- Calendar
- Productivity Chart
- Notification Centre
- Chat
- Ping
- Drop Box (File System)
- Feedback
- Audio Calling

The features I am developing in my remote working tool are being completed in stages, each of which providing a deliverable product with key added features.

Stage 1: Remote Working

This phase of development puts the key structure of the project management application in place. First the task view being implemented with the purpose of allowing all using the program to be able to view every task relating to their project. Users will be able to see relevant information about the tasks along with the people who are assigned to the task. Due to this feature the users can quickly get a grasp of the work they are to do on that given day.

The calendar feature is then developed as a visual aid for users. It will display the tasks that have been set by the project manager in a calendar format, whilst showing the timescale for the tasks. It will also include other detailed information such as who is involved in the task.

The productivity chart is the next feature being developed. This feature shows various graphs that allow the user to gauge their own effectiveness in multiple areas, all designed to aid in allowing the user to both escape isolation and increase visibility.

The final feature in this stage is the notification centre, designed to present work and tool related notifications. Work related notifications will be sent out from either employers or fellow employees. With notifications relating to features within the application being introduced as well. Having both sets of notifications will allow users to have a location for all information they require.

Stage 2: Isolation

This stage of features includes a chat function which allows users to communicate with colleagues from the same company. This allows employees to easily assist each other in the current problems being tackled. It is also useful because reduced productivity can be a result of using third party chat applications such as Facebook which are designed to distract the user.

The next feature of this stage is a ping feature which will be a button that send out an alert to other employees' notification centres stating that assistance with a task is required. This feature is valuable because it maintains communication channels between employees and increases productivity as a quick method for assistance is available.

Stage 3: Visibility

Stage three of development contains two features, a cloud-based file system and a feedback feature. These features have been paired together because it is from the file system where employers will be able to review a piece of work to then output feedback into the feedback section of the tool.

The cloud-based file storage feature will be the integration of an open-source api distributed by "Dropbox". This software offers its users cloud storage along with file synchronization to ensure there is a centralised location for files and documents. This will be accessed through the application resulting in a less time navigating to separate file systems therefore increasing your productivity.

The feedback feature is a section of the application which allows the users to see the work they participated in and how it was reviewed by their employer. Feedback will only be displayed if already provided by those who perform this action. Feedback can improve motivation to perform well in your work, thus yielding positive result with regards to visibility.

Stage 4: Communication

The communication feature is audio conference calling which allows users to communicate via audio. Having audio conference calling allows a higher level of communication which could not happen in the chat feature. Having a conference call give a more professional and efficient environment. It also provides a more natural communication medium, helping to restore a sense of normality to the workforce. All this benefits the problems outlined making it an asset to a strong remote working tool.

6.3 Overview of Dissertation

6.3.1 State-of-the-Art

The state-of-the-art section of the report contains analysis of software similar to this project. This includes both remote working tools and user monitoring tools. Drawbacks of these applications are stated to outline the common problems had by these types of tools. To follow, it is stated what this project will do to fix the described problems.

6.3.2 Requirements

The requirements section of the report provides an overview of the requirements for developing this application. This includes any functional and non-functional requirements.

6.3.3 Design

This section of the report will outline the design choices made for the applications implementation. Justification of design choices with regards to the application and any external systems, such as the database, will also be included.

6.3.4 Implementation

The implementation section of the report will cover the main features of the application with how each operates. Details of how the features work will be included to provide an insight into the running of the application, and how this resulted in the final solution. Issues that arose during the implementation stage and how they were resolved are also included within this section.

6.3.5 Testing

This section of the report demonstrates how the application is proven to work. Different tests are utilized to obtain accurate and detailed results proving the robustness of the tool.

7 State-of-the-Art

From the available remote working tools, I have selected applications that are already key players in the market to review and use for inspiration when designing my own.

7.1 Similar Tools

The similar programs that were looked into are as follows;

- GitHub, which is focused on the software development remote working career. It allows users to collaborate on projects due to its open-source community.
- Microsoft Teams, which allows for people to form group channels in which files can be distributed and audio and visual conference calling can occur.
- Todoist, which is a remote working tool focused on the organization of tasks and projects for a company.
- Teramind, which is an industry leading user monitoring software which also assists with data loss prevention, insider threat detection and workspace productivity.

7.2 Remote Working Tools

GitHub

GitHub is a remote working tool which allows its users to create, edit and review code that you either use within an organization or open source. A great aspect of GitHub is that it is tailored to software developers who are content to download and upload open-source code for anybody to use, however this tool does have problems.

GitHub suffers from the ability to prioritise tasks that are to be completed, so others aren't therefore able to determine what is of a higher importance to complete. Another problem that GitHub has is the inability to set notes on tasks and to then be able to interchange these between other tasks. The purpose of this would be to allow tasks with common problems to receive notes, or to allow tasks which inherit others creating larger tasks to take these notes or vice versa. A minor issue to note about GitHub is the lack of links to social media accounts. This is a feature that could benefit many users because being able to connect outside of work is important as remote workers do suffer from isolation.

Microsoft Teams

A feature from my application which another remote working tool has is a chat feature. This is originally used in Microsoft Teams, however the chat ability isn't something new as it has been used in social media for a while. Teams uses a basic chat ability as it solely serves the purpose of communication, it also can change the communication from individuals to groups of people by adding others into the conversation. This is the extent of this ability which is no different to the chat function we find on our phones and other mobile devices, all of which are already setup with the people we know and want to communicate with.

Once tasks go live then group chats are created for the people involved. Microsoft Teams has the downfall of having no permission settings when users are being added to a group. Users will have access to everything within the channel, this being a problem because if, for example, a project manager decided to add a client to their channel so they could view a section of work then the client would have been granted access to everything available to those in the project group.

One of the large problems that Microsoft Teams suffers stems from its file saving and structure. The issue being when documents are uploaded to the team channel, they are dumped into the channels root folder with no folder hierarchy, if you were to then try and reorganize these files, all the links in the channel would be destroyed making access to them much harder.

A small problem that Microsoft Teams suffers from is not being able to move channels between teams, this means that a group working within a channel would be unable to mirror the group and their files over to the new channel. This is an issue because it produces an unnecessary duplication of the same group in MS Teams.

Todoist

Todoist is a management tool which has multiple versions; a basic version for the everyday use and a paid advanced version which is for businesses. The application has no communication features resulting in having no way to directly communicate, therefore users must rely on pre-existing methods of communicating. The issue with this method is the lowered productivity and increase in distractions. Having users leave the application to communicate results in productivity being lower due to users losing concentration. By having users go onto a social media platform to communicate about a task it leads to being distracted by what that social media site has to offer. This is how social media platforms are designed.

7.3 User Monitoring Software

Teramind

Teramind is a user monitoring software which is used to track and monitor the work being completed by employees. The following is a list from Business.com [3] which contains what the tool monitors;

- Webpages
- Instant Messaging
- Emails
- Searches
- File Transfers
- Keystrokes
- Printed Document Tracking
- Snapshots (users screen)

By having all these areas monitored the company utilising the tool can ensure that client information is being used and dealt with correctly and securely. The issue that arises from having such an in-depth monitoring of a user stems from the lack of privacy the user has. By monitoring everything listed above it can therefore lead to the software monitoring and recording information that is sensitive to the user.

7.4 Issue Fixes

GitHub

This project will fix GitHub's lack of task prioritisation by granting the users the ability to define the importance of tasks to be completed. When somebody goes to the view the tasks, they will be able to see the importance of tasks based on the assignment it has been provided.

GitHub also fails to have a method for users linking social media accounts. This project will incorporate the ability to view links to user's social media accounts if they have been made available by said user. This will be a feature that the user can choose to show or not. If the user does decide to enable this feature, then this will be displayed on the user profiles which can be seen when a user's profile is selected from the chat window.

Microsoft Teams

MS Teams was identified to have limited permission settings. This project will fill this gap by ensuring that different levels of authorisation are incorporated into each task group. The permission levels chosen are "coordinator", for those who have overall ruling over the tasks, these would be project managers; then "members" whom would be employees working on the project, and finally "guests" who are users added into the group to see tasks but have no access to documents and workings within.

This project will fix the issue regarding a suitable file system. This tool will leave the folder management to the project manager, allowing them to create their own hierarchy of folders. The benefit of this being that users can use a common filing system for the entire project as it has been created by a single person. Having the file system setup in this way is also beneficial for the users because it can be easily changed and altered without destroying the previous folder styles.

Microsoft Teams minor problem of not being able to navigate channels between teams will not be directly addressed in this project. However, it will function differently because the tasks that are issued are done so in a scrum format. This means once the task is completed it is essentially thrown away. This therefore means that my project would not require any group mirroring because tasks create the groups with users included thus preventing and duplication issues because the old group had been destroyed.

Todoist

Todoist suffers from having no direct line of communication between its users. This application being developed will fix this problem by ensuring direct lines of communication are available in the forms of text and audio chats. Fixing this problem aids users in dealing with isolation due to being in communication with others, and aids with reduced visibility due to being seen contributing to the team by attending meetings.

Teramind

Teramind over monitors their users as discussed, this project will combat this by having a set whitelist of focused apps to be measured. This project will also measure and monitor these specified apps to not discredit the users work but to help them prove their worth to their company.

8 Appendix

The appendix is a list of terminology with descriptions on specific components. These descriptions are provided to prevent the report from repeating itself, along with wanting to inform the user prior to the implementation section.

- Java Swing – This is a user interface toolkit for java. It contains multiple components that can be used to obtain fully functional user interfaces. The most used components are outlined below:
 - JPanels – These are java user interface components that act as a container, allowing for components to be added and held on top. The purpose of using a jpanel is to organize the content within using various layout methods.
 - JFrame – Like the jpanel however the frame is the component that holds everything and displays it to the user.
 - Layouts – A layout is assigned to a jpanel to aid in arranging children components. Multiple types of layout are utilized throughout the application. Examples are as follows.
 - Grid Layout – Displays data in a table format.
 - Card Layout – Has multiple panels that can be swapped between. Only one can be viewed at any given time.
 - GridBag Layout – Data is displayed in no uniform manner. Components are assigned multiple parameters to position themselves within the assigned space.
 - Box Layout – Displays components in a line along either an X or Y axis.
 - Free Layout – A layout created by the Netbeans UI editor. It allows for components to be placed anywhere using the editor tool, which then computes the required parameters to maintain the presentation.
- Netbeans – This is an integrated development environment (IDE) that allows for java applications to be developed through the user of modular software components. An example of these modules is the internal user interface editor that grants the ability to create a UI without any coding.
- Apache Web Service – Apache is a free open-source web server software that can be run on your own computer. It allows for connections to be established between servers and browsers.
- PHPMyAdmin – This is a free open-source software coded in php that makes use of MySQL. Users are granted the many database related abilities, some of which are outlined below:
 - Visually creating a database.
 - Create user accounts.
 - Create relationships between tables.
 - Indicate multiple forms of data that correspond to a single field/record.
 - Host multiple databases.
- Primary/Foreign Key – These keys indicate relationships between different tables within the same relational database. While primary keys must be unique to act as an identifier for each record, the foreign keys must link to the primary key from another table. However, the foreign keys need not be unique.

8 Requirements

8.1 Functional Requirements

8.1.1 Error Handling

The error handling in this application covers multiple areas. These areas are as follows.

- User Input – Error checking exists for user input to ensure that null values are not submitted to forms unless allowed.
- Web Server/Database Connection – Try statements are used to ensure that failed connections to the web server and database do not result in crashes.
- Logic Errors – These errors exist when the code is essentially working but does not perform in the intended manner.
- Loop Escapes – Implementing escape functions for loops are important to prevent an infinite loop scenario from occurring. This would result in the application freezing as it is unable to escape from the chunk of code.
- Console Printing – While error handling, any errors that occur are set to print to the console. This aids the program in preventing crashes, as it is preferred to have errors printed than to have them crash the application.

8.1.2 Display

Ensuring content is appropriately displayed to the user is key. The information displayed should be done in a clear manner that results in users having no problems in attempting to retrieve a specific output.

8.1.3 Application Features

1. The application must allow for users to see a list of tasks that they have been assigned.
2. Must be able to select tasks to view all information related to them.
3. Should allow for users to dynamically create new:
 - a. User Accounts
 - b. Company Databases
 - c. Notifications
 - d. Tasks
 - e. Chat Messages
 - f. Feedback
 - g. Document Files
4. Should be able to dynamically delete:
 - a. Tasks
 - b. Notifications
 - c. Document Files
5. Must be able to use graphs to gauge their own effectiveness during their time spent with application open.

8.2 Non-Functional Requirements

8.2.1 Scalability

This application is designed to be operated in windowed or full screen mode. However, the windowed mode is set to a fixed size. The reason for this is to maintain the style of the application whilst not adding a sizeable portion of additional work which would not benefit the tool. It's not beneficial due to none of the outlined issues with modern remote working tools being solved by implementing this aspect.

8.2.2 Accessibility

This application is made accessible by users who may have visual impairments. This is achieved by both having a strong default colour scheme, and the ability to change the colour scheme of the application. Giving this option to users allows them to customize the application to fit their needs and requirements.

9 Design

9.1 Database Design

The database design was altered over many iterations to allow for changes made midway through development. However, the designs for the database have the same “frame” as the implemented end-result.

9.1.1 Database Tables

Database tables will be created when the company profile is initially established. The database contains information prevalent to the organization and its employees.

The database consists of multiple tables which store the following information.

- The users who have access to the database.
- The statistics for each of the users.
- The tasks available for users to complete.
- The notifications that exist under the company profile.
- The communications made between users in the chat feature.

User Table:

The user table contains records of the users with access to the database. Each record contains a unique user ID, the user's name, their date of birth, and links to social media accounts. The user's password is not stored along with the record due to the security issues that could arise, especially with the way the project's database access has been designed.

Field Name	Structure	Description
UserID	Char(5)	Unique ID for the user
Name	Varchar(100)	The full name of the user
DateOfBirth	Date	The date when the user was born
Socials	Varchar(1000)	Any links to social media accounts which the user would like to share

Figure 2: User Table Structure

Task Table:

The task table contains the list of tasks created by users of the application. Each task record comprises of the following fields;

Field Name	Structure	Description
TaskID	Char(5)	Unique ID for the task
StartDate	Date	The start dates
EndDate	Date	The end dates (submission)
Details	Varchar(5000)	Details of the task to aid with understanding what is asked
TaskName	Char(100)	Name of the task

Feedback	Varchar(1000)	Feedback given from the employer after work has been submitted
Active	Int(1)	The boolean value of whether a task is still in use
Importance	Varchar(100)	The level of importance this task is assigned
UserID	Char(5)	Unique ID for the user

Figure 3: Task Table Structure

The active field is a boolean value which can either be represented with the value 1 or 0. This indicates whether the task is to still be displayed on the user's task list or not. Tasks can be dynamically added to the database during runtime of the application.

Statistics Table:

Some database tables are initialized as sub-tables because they hold information about a record from another table. An example of this is the statistics table which has a sub-table called "statistics_login_records". The statistics table contains a unique statistics ID along with the ID of the corresponding user and the type of record it is. The sub-table contains additional information, in this case about login records. The statistics section of the database is setup this way to allow for statistic records with different types of measurement to be logged. The statistics ID is an unique primary key in both tables, however due to the outlined setup of these tables the ID is a foreign key in the sub-table.

Field Name	Structure	Description
StatisticsID	Char(5)	Unique ID for the statistic record
UserID	Char(5)	The unique ID assigned to the user
StatisticType	Varchar(1000)	The type of statistic that has been measured

Figure 4: Statistics Table Structure

Field Name	Structure	Description
StatisticsID	Char(5)	Unique ID for the statistic record
LoginTime	Timestamp	The time the login session began
LogOutTime	Timestamp	The time the login session expired
NoOfMessagesSent	Int(255)	Count of how many messages were sent
NoOfMessagesRecieved	Int(255)	Count of how many messages were received

Figure 5: Login-Records Table Structure

Notifications Table:

This table is purposed with recording the notifications that are sent out to the users of the application. Each record within this table includes the following fields.

Field Name	Structure	Description
NotificationID	Char(5)	Unique ID for the notification record
SenderID	Char(5)	The unique ID for the person who sent it
RecipientID	Char(5)	The unique ID for the person who receives it
Content	Longtext	The content of the notification message
TimeSent	Timestamp	The time the notification was sent
Active	Int(1)	An integer that represents if the notification is in use
Name	Varchar(1000)	The name of the notification

Figure 6: Notifications Table Structure

The “RecipientID” can be left blank to allow for no direct recipient. By leaving this field null it represents that the entire set of users within this company can see this notification.

Chat Table:

This table is purposed with recording the chats that are sent by the users of the application. All messages between everyone are stored in the same table, the table makes use of ordering the contents by “TimeSent”. Having all messages in the same table is not a pretty design but works because the unique identifiers for both the sender and recipient exist. Each record within this table includes the following fields.

Field Name	Structure	Description
ChatID	Char(5)	Unique ID for the chat record
SenderID	Char(5)	The unique ID for the person who sent it
Message	Longtext	The content of the chat message
RecipientID	Char(5)	The unique ID of the person who receives it
TimeSent	Timestamp	The time that the message was sent at

Figure 7: Chat Table Structure

9.1.2 Normalization

Normalization conveys the method of converting this database from a flat file database to a relational database that contains multiple tables with relationships between them all.

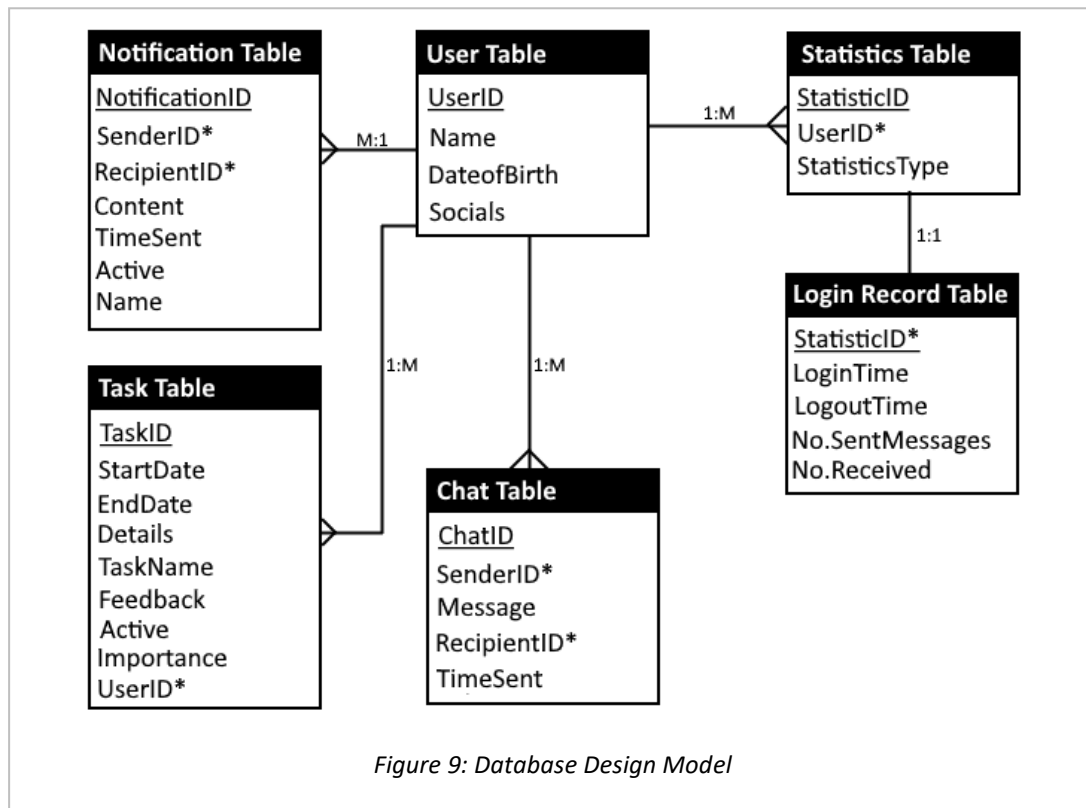
UNF	1NF	2NF	3NF
UserID	<u>UserID</u>	<u>UserID</u>	User Table
Name	Name	Name	<u>UserID</u>
DateOfBirth	DateOfBirth	DateOfBirth	Name
Socials	Socials	Socials	DateOfBirth
TaskID			Socials
StartDate	<u>TaskID</u>	<u>TaskID</u>	
EndDate	StartDate	StartDate	Task Table
Details	EndDate	EndDate	<u>TaskID</u>
TaskName	Details	Details	StartDate
Feedback	TaskName	TaskName	EndDate
Active	Feedback	Feedback	Details
Importance	Active	Active	TaskName
StatisticsID	Importance	Importance	Feedback
UserID		UserID*	Active
StatisticsType	<u>StatisticsID</u>		Importance
LogInTime	UserID	<u>StatisticsID</u>	UserID*
LogOutTime	StatisticsType	UserID*	
NoOfMessagesSent	LogInTime	StatisticsType	Statistics Table
NoOfMessagesRecieved	LogOutTime		<u>StatisticID</u>
NotificationID	NoOfMessagesSent	<u>StatisticsID*</u>	UserID*
SenderID	NoOfMessagesRecieved	LogInTime	StatisticsType
RecipientID		LogOutTime	
Content	<u>NotificationID</u>	NoOfMessagesSent	Login Records Table
TimeSent	SenderID	NoOfMessagesRecieved	<u>StatisticsID*</u>
Active	RecipientID		LogInTime
Name	Content	<u>NotificationID</u>	LogOutTime
ChatID	TimeSent	SenderID*	NoOfMessagesSent
SenderID	Active	RecipientID*	NoOfMessagesRecieved
Message	Name	Content	
RecipientID		TimeSent	Notifications Table
TimeSent	<u>ChatID</u>	Active	<u>NotificationID</u>
	SenderID	Name	SenderID*
	Message		RecipientID*
	RecipientID	<u>ChatID</u>	Content
	TimeSent	SenderID	TimeSent
		Message	Active
		RecipientID	Name
		TimeSent	
			Chat Table
			<u>ChatID</u>
			SenderID*
			Message
			RecipientID*
			TimeSent

Figure 8: Database Normalization

9.1.3 Database Design Models

The database design model is a representation of how the tables are connected together through the use of primary and foreign keys. Primary keys are underlined whilst foreign keys contain an asterisk (*). Each connection has a relationship between the tables, these are outlined by the small text next to the connecting lines.

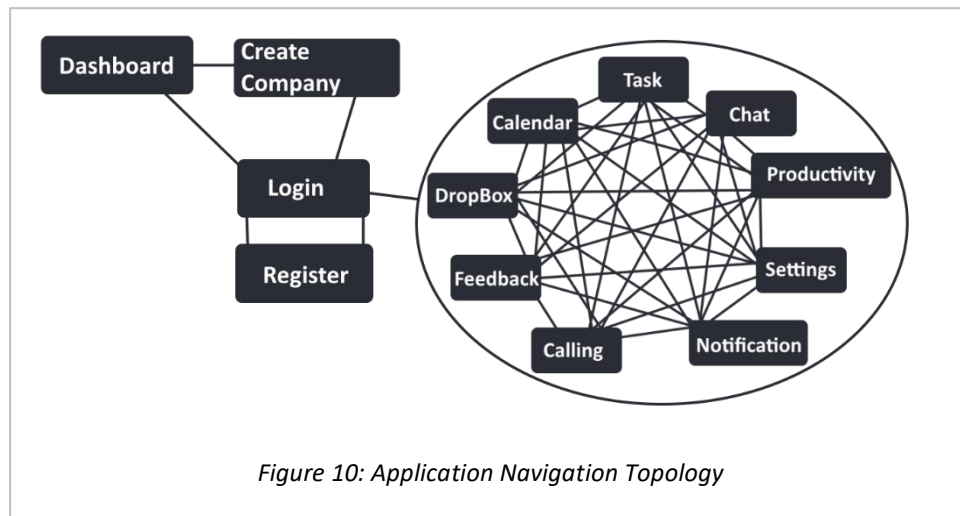
The relationship between the user table and the task table is currently one user to many tasks, meaning a single user can have many tasks. Originally the task table was to have multiple users assigned to the task, but this was changed. Instead by assigning a leader of the task the database is kept simpler, due to not have a many-to-many relationship which would require an additional table between them.



9.2 Software Design

9.2.1 Navigation Hierarchy

The navigation of the application is a combination of different types of navigation topologies. The initial program begins with a line topology as the user can only navigate horizontally between pages. Once the user has reached the main application the program changes to a mesh topology. The programs initial line topology was designed this way to provide information at the beginning of the application in the forms of a dashboard. Then the user must login or register to the application or create a new company profile. This is all achieved in a linear manner to ensure the security of the tool is maintained.



The mesh topology was the chosen navigation hierarchy in the design stage for the application. This decision was made to allow for quick navigation to any available features, allowing for an uninterrupted working.

Within each feature smaller subclasses exist; these classes are often extending JFrames which are Java Swing components that can be accessed by the user. The reason these subclasses do not appear in the applications topology is they are visual within the parent component. When a parent class, i.e. a feature, is created multiple sub classes are also created as objects allowing internal methods to be called and accessed. These subclass objects can contain user interface components which will read user input and then react accordingly. These interactions can lead to other pages being opened within the application.

9.2.2 Components

To design the application and its components, inspiration was drawn from several other applications. Any consistent styles and patterns found between different applications were noted and contributed into the design stage.

The components were designed to be universal meaning the styles of each component type (button, label, text field, etc.) would be applied to every element. This allows the program to change its colour scheme dynamically by altering the global colour scheme which was created in a globally accessible object.

For labels, buttons and some additional components action listeners will be applied. Action listeners react and perform actions based off the users' interactions with the components. In terms of design these action listeners will include a hover effect which will apply a change in colour to indicate where the user is pointing. This is a common with applications as it helps to outline what actions the user has the availability to perform.

9.2.3 UI Design

The initial design of the UI was to have a title bar across the top which allows for the program to: close, minimize to taskbar, and to full screen. A menu would span down the left side of the screen while listing all the accessible features available to the user. The remainder of the space would be the applications content. This space would update to display the content for the feature selected through user input, or any information provided from objects and sub-classes.

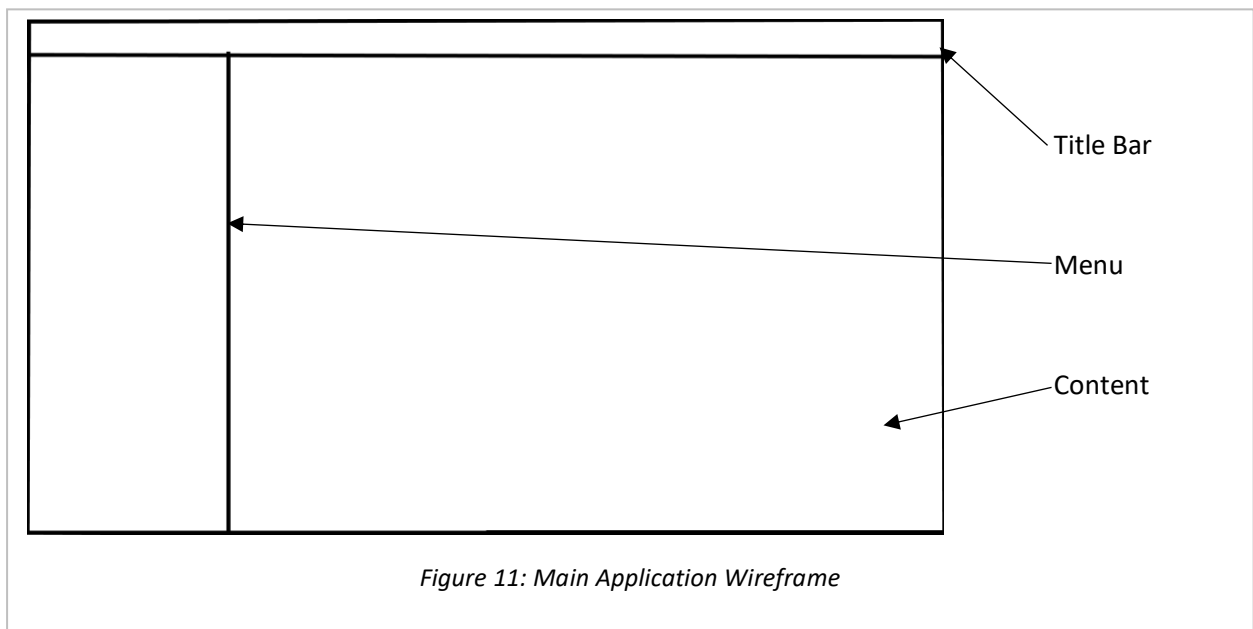
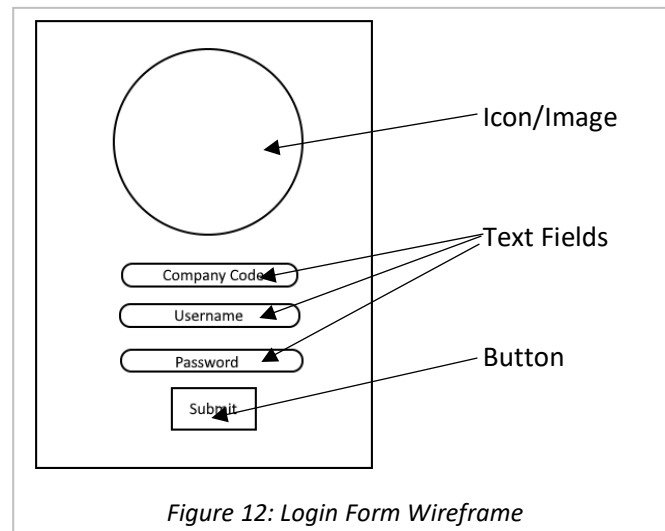
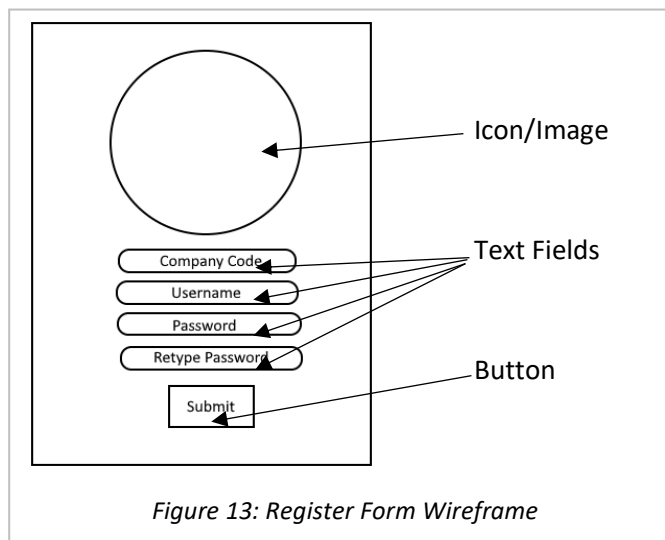


Figure 11: Main Application Wireframe

9.2.4 Authentication Login/Register



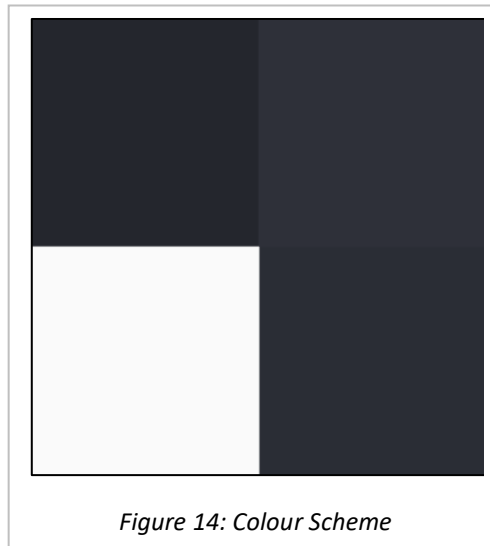
The login page is designed to be simplistic yet informative to the user. As a small popup window, the user is informed as to what piece of information goes where due to the indications in the text fields. The form shall take in the code assigned to the company, the username for the individual user, and the password chosen by the user. From here the user would then submit their entered information, if successful the next stage of the user interface will be loaded. However, if unsuccessful the user will be prompted by a notification that the login attempt was unsuccessful.



The register page is similar to the design of the login page. It too is a small popup window that is informative to the user by indicating what is to be entered and where. As this page will create an account rather than connect to one, an additional text field has been added to allow for verification when entering a password. This additional level of verification prevents the user from entering a password that contains a syntax error such as a typo. If successful, the user is informed of the successful attempt and is directed to the next stage in the user interface. If unsuccessful with their register attempt the user, like in the login page, will be informed of the failed attempt with a request to try again.

9.2.5 Colour Scheme

The colour scheme of this application was chosen to be a dark theme, the colours all being a shade of dark blue besides the text colour which is a plain white. The white colour being chosen for text because it allows for strong readability against the darker tones of the form.



The user benefits from having a dark style for the application as the darkness will allow for longer usage times. Using a lighter application usually results in an increased level of tiredness. This is key as this remote working tool is being oriented around the user with an increase in distractions being one of the problems being solved. By reducing the tiredness level of the user, we are therefore reducing the number of distractions on screen.

10 Implementation

10.1 Overview

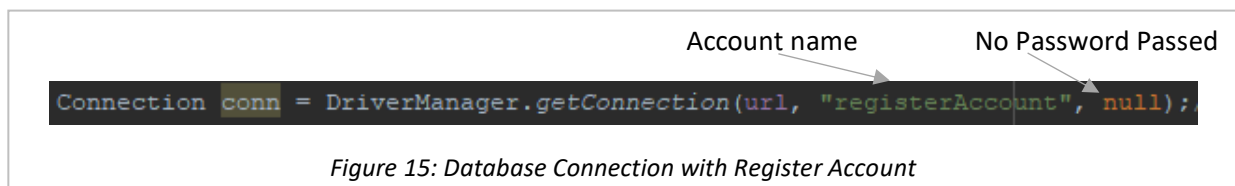
This section will cover the implementation of each feature individually, though some code is recycled and reused throughout. Each section will contain details on the feature's creation, any issues that arose, and any additional information.

To implement the designs, NetBeans was used as it contains a built-in user interface editing tool. This tool grants the ability to easily design user interfaces with components such as JPanels and JFrames. Though this feature was not always utilized it was valuable when implementing more complex interfaces because manually coding styles for individual components was overly complex and time consuming. Also due to the method NetBeans uses to initialize components, when coding components that are dynamically added during runtime it was beneficial to do so manually.

10.2 User Account

10.2.1 Creation

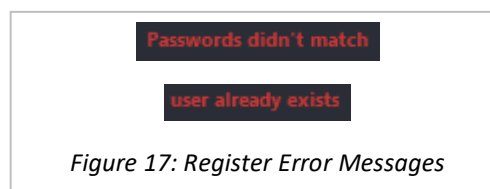
User accounts are initially created within the application. Once a user has attempted registering to a company with provided details, an attempt is made to add the user account to both the database and to the "phpMyAdmin" web tool hosted on the web server. Firstly, the application tries to connect to the web tool by using the provided "Register Account". This is an account created for the sole purpose of registering new accounts. No password exists for this account however restrictions are applied to prevent any security issues. The restrictions reduce the number of actions that can be performed, this limits the account to only be able to ass to tables in the relevant company database. Afterwards the account registers the user to the new database and to the web tool.



User name	Host name	Password
Admin	localhost	Yes
AlaskaS	localhost	Yes
EwanD	localhost	Yes
mysql.session	localhost	Yes
mysql.sys	localhost	Yes
registerAccount	localhost	No
root	localhost	Yes

Figure 16: Web Server User Accounts

The web tool “phpMyAdmin” stores the password and username together on its internal list of user accounts as seen above (the register account has no password as indicated). The database contains a user table which holds only a few pieces of information, this does not include the password for security reasons. Once the user is successfully added they are then directed to the login form.



During the registering of the new account if any errors occur then user is notified through the use of an JLabel. This label is in red font which, compared to the background, stands out to the user. This is to aid visibility.

10.2.2 Security Concerns

The security of user passwords is of paramount importance due to the level of accessibility granted on the company database when a user has access. To ensure security the passwords have not been stored on the user table within the company database. This is due to the method in which the database is loaded into the application. The tables are fully loaded with all records and then reduced later on to ensure local copies only contain the relevant information. If the passwords were included in the table, then every time this remote working tool is used, for a moment, all the passwords are read and held locally. This is an issue as having access to all these user accounts could result in unauthorized attempts to gain access to a company’s database, and therefore company information.

10.3 UI and its Components

The user interface consists of separate components within each feature. All these features are contained and displayed within the programs content panel however, the parent interface consists of the title bar and menu. These will be discussed in this section.

10.3.1 Title Bar Panel

The title bar panel was custom made to maintain consistency in style with the rest of the application. There exists a default windows title bar however this allows for no styling options.

The title bar is accessed by creating the title bar as an object which is then added to the frame. Upon the object's creation the constructor within the class calls methods to create components, apply action listeners, and finally apply styling. Initially the title bar was created in the most parent class as this

```
titlebar tb = new titlebar(UI.this); //create the titlebar class with the JFrame as the parameter  
tb.applyTitleBarSettings(titlebar); //call to apply styles and settings to the titlebar
```

Figure 18: Title Bar Object Creation and Initialization

encapsulated the entire application. However as new iterations of the application were created and new JFrames were being created, the decision to put the title bar into a separate loadable class was made.

The title bar consists of three buttons inserted into a layout manager which aligns the child components, in this case the buttons, to the right of the panel. Each button performs the same actions that would be found on a traditional title bar. From left to right these actions are as follows.

1. Minimize the application to the taskbar.
2. Toggle the application between full screen and windowed mode.
3. Exit the application.

Like on the typical title bar found on most computer applications there are icons presented on the buttons instead of text. As these icons are uniform, most users understand what actions they perform. To ensure this understanding exists on this application icons were added though they were custom created, again to match the styling of the application.



Figure 19: Title Bar Interface (with ping bell)

The title bar in a single instance on the main application interface has a fourth button to send a ping notification to all users within the same organization. This icon is only added if a ping object is passed into the title bar class when called. When added the layout stays the same with the three constant buttons maintaining their position.

10.3.2 JPanel: Menu Panel

The menu displays all the features that are available to the user along with an option to view the settings page. The menu also contains a smaller sub-menu which contains the list of tasks, a dropdown button is used to present this list. This small list contains the tasks that exist on the database within the task table.

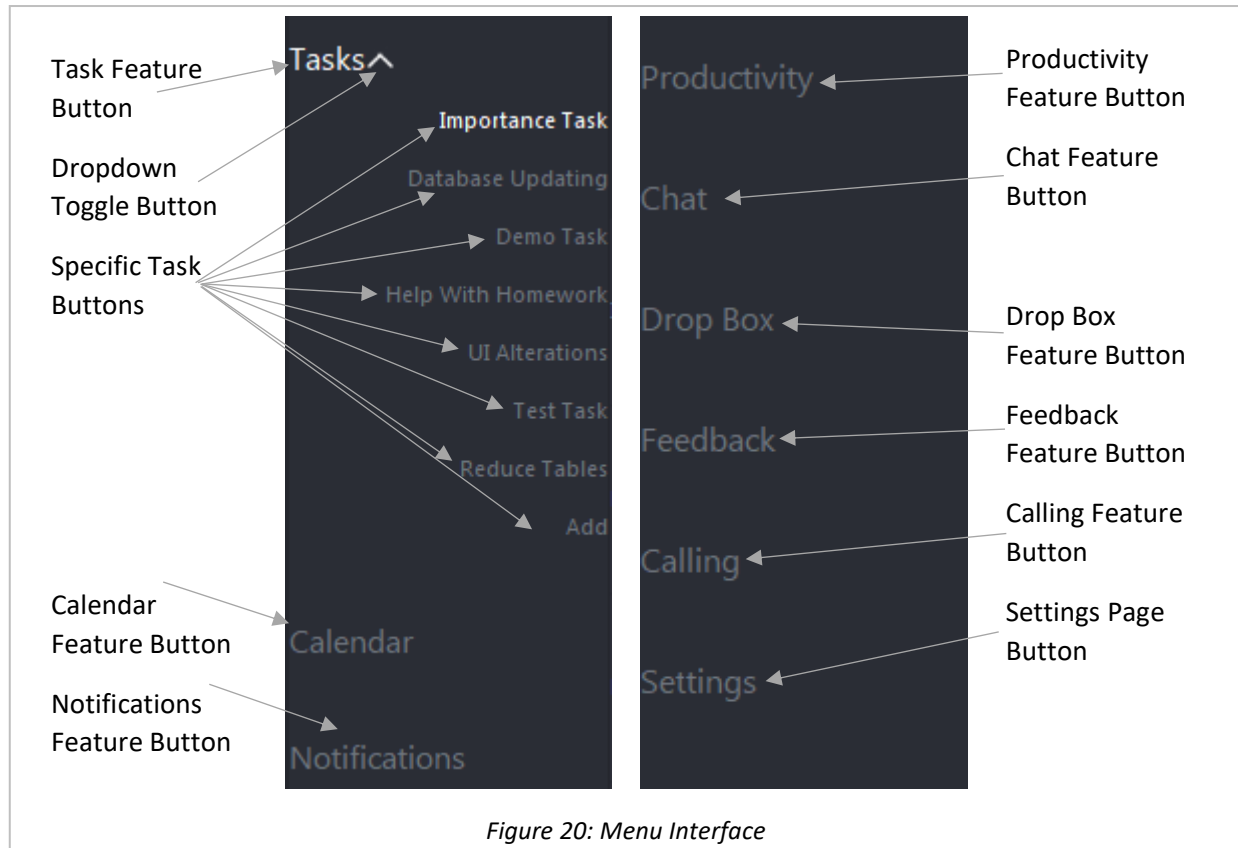


Figure 20: Menu Interface

The menu is generated dynamically during runtime of the application because of the small task list that can be displayed. Since the tasks must be loaded from the database and then added to a list, manually creating the menu to display the content was the utilized method.

The implementation of this menu is as follows: a local copy of the databases task table is created, then each record within said database is read. The name of each task was then added to an array list because these lists allow for an undefined length, thus allowing for an unlimited number of tasks to be added which is ideal due to the random quantity of tasks a company can retain. Once the list of tasks names has been created buttons are then built with these given names. Action listeners are then added to the buttons to allow for user interaction. The listeners carry out methods which update the content panels display to present information of the task with the correlating name.

For each feature that exists, a button is created that allows users to update the content panel with the chosen feature. These buttons are created in a loop containing a switch statement which runs each case based off the counter that exists in the loop. As the loop counter is incremented the switch statement runs through each case to generate the button for each feature.

With each feature button an action listener is added to allow for user interaction. These action listeners conduct the following steps that can be seen in the figure below.

```
calendar.addActionListener(new ActionListener() { //button action listener
    public void actionPerformed(ActionEvent e) {
        UpdateDatabase updateTables = new UpdateDatabase(databaseName, loggedInUser, password); //c
        tables = updateTables.tables; //update the tables parsed from the database

        changeCard("calendar card"); //change card on the content panel card layout
        calendarPanel.removeAll(); //clear the contents of the task JPanel
        updateMenuPanel(); //call method to update the menu panel

        calendar externalCalendarPanel = new calendar(tables, databaseName, loggedInUser, password)
        calendarPanel.add(externalCalendarPanel.calendarPanel); //add the JPanel from the external c
    });
```

Figure 21: Button Action Listener (Displays Calendar Feature)

1. Create an "Update Tables" object to obtain an up-to-date version of the database tables.
2. Update the local tables that will be used within the application with the ones retrieved from the database.
3. Swap the content panel by changing the card displayed on the card layout.
4. Clear all old components from the panel that corresponds to the selected feature. (In this example this is the calendar feature)
5. Call method to update the menu panel.
6. Create new versions of the feature object. (Calendar object in this instance)
7. Add the object or its internal jpanel to the content panel in the main UI.

The menu panel is only loaded once at the beginning of the applications lifetime. This means any new tasks that are added to the database by any user will not trigger an update on the menu panel, therefore the new tasks will not be displayed. Only with a full system restart can the menu be updated.

10.3.3 JPanel: Content Panel

The content panel is the JPanel that displays content to the user. It adds the JPanel created by the feature object that corresponds to the user's choice.

```
public void assignCards() {
    content.add(taskPanel, "task card");
    content.add(calendarPanel, "calendar card");
    content.add(notificationPanel, "notifications card");
    content.add(productivityPanel, "productivity card");
    content.add(chatPanel, "chat card");
    content.add(dropboxPanel, "dropbox card");
    content.add(feedbackPanel, "feedback card");
    content.add(callingPanel, "calling card");
    content.add(settingsPanel, "settings card");
    content.add(emptyPanel, "empty card");
}
```

Figure 22: Card Layout Assignments

The panel is able to switch between features and other content by making use of a card layout. This is a type of layout which instead of organizing components such as buttons and labels into a specific order like

other layouts, it organizes JPanels that occupy the same space. The card layout switches between user chosen panels which display different information. The chosen panel has its visibility set to true, with the other contained panels having their visibility set to false. To interchange between these panels each panel is assigned a “card name” (shown in figure 22) which is used to reference the selected panel. For example, having the task panel chosen would trigger the card layout to load the “task card”. These card assignments are made dynamically when each feature object is initialized.

10.3.3 JButton: Ping Feature

The ping feature allows users to simply press a button to request help from all users working from within the same company. This alert appears through the notification feature along with other notifications. Other employees will see the name of the user requesting assistance in the content field, with the notification name being a predefined title of “Assistance Notification”.

This feature grants a quick method to request assistance leading to both an increase in productivity and a higher amount of communication between employees. Maintaining communication channels provides those involved a greater sense of visibility amongst the company as being more involved leads to additional contributions, and also a lower chance of experiencing isolation due to constant involvement in other tasks.

10.4 Task Feature

10.4.1 Overview

The task feature exists to inform the user of all the tasks assigned to them. From this view they can see the following information.

- Task Name – This is an easy indicator for the task. The naming system for a company's tasks is highly flexible. This is because the naming system is fully customizable due to the “string” data type used. This allows for text, number, and symbols to be utilized.
- Start Date – The start date indicates the date that the task was assigned.
- End Date – The end date indicates the day in which the task is to be completed and submitted by.
- Details – This field contains any information relevant to the completion of the task.
- Importance – The importance is an indicator the urgency of each individual task. This is measured through a scale ranging from high to low.

10.4.2 Layout



Names	Start Date	End Date	Details	Importance
Importance Task	2023-08-26	2023-08-30	n/a	Low
Database Updating	2021-07-23	2021-07-27	testetete	High
Help With Homework	2021-07-23	2021-07-27	testetete	Low
UI Alterations	2021-08-15	2021-08-31	No dets req	Medium
Test Task	2020-11-06	2021-02-11	This task is the first in the testdb	Medium
Reduce Tables	2020-11-06	2020-11-27	Test Task 2 details	Low
+ Add New Task				

Figure 23: Task Feature Display

The layout for this feature is a simple table. This allows for easy readability due to tables being a common structure for displaying data. The headers for each field are clearly indicated at the top. When the user hovers their cursor over a record the text is highlighted, this indicates to the user what task they are selecting. At the bottom of the display exists a button that directs the user to a separate form to add a new task to the database.

```

addNewTask.addActionListener(new ActionListener() { //create listener for the add new task button
    public void actionPerformed(ActionEvent e) {
        taskPanel.removeAll(); //clear the taskPanel
        addNewTask add = new addNewTask(tables, databaseName, loggedInUser, password); //create a
        taskPanel.add(add); //add the addNewTask JPanel to the now empty JPanel
        taskPanel.revalidate();
        taskPanel.repaint(); //update the JPanel
    }
});

```

Figure 24: Action Listener for "Add New Task" Button

The action listener added to this button; clears the contents of the task panel that is currently viewed by the user, creates an "add New Task" object, and then adds it to the task panel. The task panel is revalidated and repainted to update the contents on the screen.

10.4.3 JPanel: Displaying Specific Tasks

A separate class for displaying tasks exists called "Specific Tasks". This class contains multiple parameters required to load information relevant to the indexed task. The parameters include the indexed record from the locally created tables, and the variables required to establish a connection to the web server's database. While this class creates a jpanel to hold and display all the relevant information, there is also a checkbox that makes use of the database connection.

Figure 25: Specific Tasks Display

If a task is to be displayed in the list of tasks, it must be set to active. This indicates if the task is in use or not. Due to this when the individual task is loaded a checkbox is created indicating the tasks activity. Therefore, by default this checkbox is set to selected. However, the user has the ability to change this. If the checkbox is selected or unselected it triggers an update method to be called, this results in the record being changed on the database.

10.4.4 JPanel: Adding New Task

Adding a new task is conducted in a separate class called "Add New Task". This class extends a jpanel so it can be displayed on the applications main content panel. This form contains text fields for the user to input a name, start date, end date and details, and then a drop down for selecting importance. Once all

data is entered the user can submit the record to the database. If the user fails to enter data, the record implements its error handling by preventing the records submission.

The image shows a dark-themed window titled "Add New Task". Inside the window, there are several input fields and a button. Labels with arrows point to each of these elements from the right side of the image:

- Task Name:** points to a text input field.
- Start Date:** points to a date input field with a calendar icon.
- End Date:** points to a date input field with a calendar icon.
- Details:** points to a text input field.
- Importance:** points to a dropdown menu currently showing "High".
- Submit New Task** points to a button at the bottom left of the form.

Figure 26: Adding New Task Form

10.4.5 Implementation Issues

In implementing this feature, it was difficult to obtain information about a specific record. This being the first feature that was implemented. Though the database was loaded into the application, attempting to get java swing components to be created dynamically was a challenge. However, from figuring out the solution to my problem, I had set myself up to replicate a similar system across all the other features.

10.5 Calendar Feature

10.5.1 Overview

The calendar is a more visual representation of the tasks feature. It displays the tasks on the date of which they are due to be completed. This allows users to have a stronger grasp of what tasks to do on any given day.

10.5.2 How it Functions

Without the use of libraries, the calendar was created by making use of the “Calendar” object. This allows for more simple conversions between dates and times. To make conversions easier the object has access to fields such as “YEAR”, “MONTH”, and “DAY_OF_MONTH”. With these fields the indexed date can be easily changed making dynamically adding tasks to the custom calendar less of a challenge.

```
cal = Calendar.getInstance();//create the calendar instance  
createCalendarPanel(cal.get(cal.MONTH), cal.get(cal.YEAR));
```

Figure 27: Creating Calendar Object

In the figure above it can be seen that the calendar object is initialized by obtaining an instance of a “Calendar”. With this object, named “cal”, the method to create the calendar is called with the calendar objects month and year being passed as parameters.

To load the individual calendar jpanels representing days, a loop is iterated through. This loop iterates for the number of days that the currently index month contains. For each of these days a label is added to indicate the date of the month to the user. Within each loop iteration another loop is ran, this loop goes through the locally created task table and each time accesses that individual records end date. If the date on the tasks record matches the indexed calendar date, then the tasks name is inserted into the jpanel.

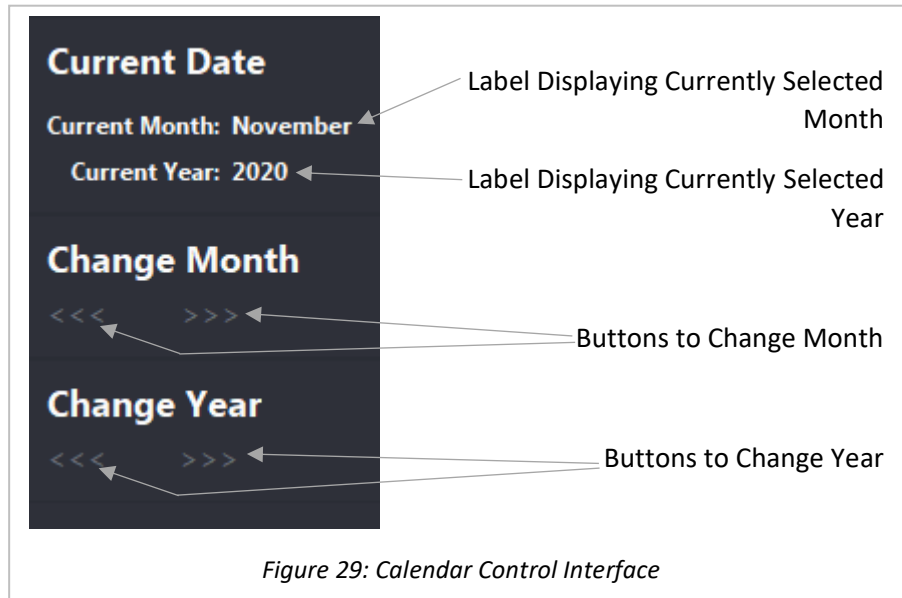
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27 Reduce Tables Add	28
29	30					

Figure 28: Calendar Display

To control the calendar a separate controls panel is created, this will be discussed afterwards. Due to the controls being created in another class, a click listener is implemented in the calendar. This listens for any user interaction from the controls panel and then outputs the correlating action.

10.5.3 JPanel: Calendar Controls

A controls panel is created in the constructor by accessing an external class called “Calendar Controls”. This class extends a JPanel and controls the user interface for navigating through the calendar. The panel displays the current month and year to the user, along with two sets of controls. The first set is to change the indexed month, and the second to change the year.



When the user interacts with these controls the click listener in the main calendar class detects which button was pressed and then calls a method to update the calendar while passing two variables. These variables state whether it's the month or year being updated and if this is a move forward or backwards. When this update method is called, statements are utilized to ensure the calendar doesn't get assigned a wrong date. A check is done to determine if the month is already set to either January or December. The following scenarios describe why this is run;

If the user chooses to navigate forward one month while the date is already set to December 2020 then the next month would be January of 2021. The issue here is this is not always the case. Therefore, it is important to determine if the currently indexed month is at the start or end of the year, depending on the direction the user has chosen.

In the figure below this error checking is completed. If the month is set to zero and the date is being pushed forward then set the month to zero and the year to plus one (just like in the scenario), if not then simply add one to the month value of the calendar object.

```
if(month == 11){//if the month is already decemeber
    createCalendarPanel(0, year + 1);//set the month to january and the year to the next year
}else{
    createCalendarPanel(month + 1, year);//set the month to the next month and keep the year
```

Figure 30: Date Error Checking Example

When this scenario arises, the code is setup to update the month and year vales of the calendar object. This will result in the correct date being presented to the user.

10.5.4 Responsive Components

The calendar and the controls both have components that visually respond to user input. As stated, the calendar creates a separate jpanel for each day. During its creation, an action listener is applied to the panel and its internal components. The listener waits for the user's cursor to enter the panels boundaries to which it responds with highlighting the panel, this is undone once the cursor leaves the boundaries. The labels within these panels that indicate the tasks names also have similar effects. The text colour is highlighted once the user hovers their cursor over the label, this implies to the user that they are able to click on it. When this occurs the user is navigated to the specific task class which displays the information of the indexed task. This feature is outlined previously in the “Displaying Tasks” part of the “Task Feature” section of the report.

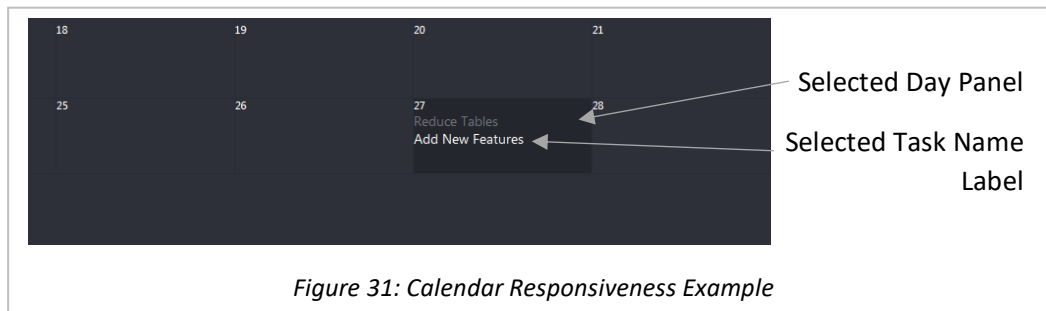


Figure 31: Calendar Responsiveness Example

The controls also have a similar responsiveness to them. The buttons used to navigate throughout the calendar have a hover effect applied to them, this allows for the user to visually see when their cursor is above the button.

10.5.5 Implementation Issues

This section was difficult to implement, especially without libraries. They weren't utilized in this section due to the difficulty that came with attempting to apply open-source libraries to the code. However, in developing a custom calendar, the visual designs of it were much easier to implement thus resulting in a calendar that visually matches the rest of the application.

10.6 Notifications Feature

10.6.1 Overview

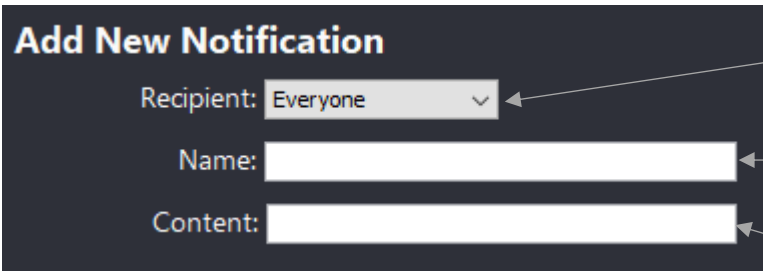
The notifications user interface contains a scroll pane for displaying the list of notifications, and a jpanel for the controls. With each notification record from the database having a separate jpanel created, thus allowing for its corresponding information to be displayed. Each jpanel created for individual records are created using a class extending a jpanel. This class contains labels for displaying the relevant information whilst also having a radio button designated for allowing the user to select and delete notifications. Also, on the jpanels are radio buttons. When selected this indicate to the delete method what records are being removed.



The user is provided the ability to create, update and delete notifications which can be sent to either a specific employee or to everyone.

10.6.2 JPanel: Create and Update Notification

To create or update a notification the user can press a button in the controls section. Upon selecting this option, the user is navigated to a class that provides the means to separately create a notification or to edit one. Both options exist on the same page to provide a centralized place for creation.



The screenshot shows a form titled 'Add New Notification'. It contains three input fields: a 'Recipient' dropdown menu with 'Everyone' selected, a 'Name' text field, and a 'Content' text field. Arrows point from labels on the right to each of these fields.

Figure 32: Add New Notification Form

To create a new notification the class makes use of the three parsed variables that allow for database connectivity. From the available user interface components, the user can enter the name, content, and recipients for this new record. Upon completion the submit button is available to insert the new record on the database.

The screenshot shows a form titled "Update Notification" on a dark background. It contains four labeled components: a "Select Notification" label above a "Select Option:" dropdown menu; a "Name:" label above a text input field; a "Content:" label above a text input field; and an "Active:" label above a checked checkbox. Arrows point from text labels on the right to each of these four components: "Select Notification Combo Box", "Name Text Field", "Content Text Field", and "Active Check Box".

Figure 33: Update Notification Form

To update a notification, a combo box is used to list the notifications that already exist. From this component the user can select the notification they wish to change. The user then inserts any changes to be made to either the name, content, or activity (activity stating whether or not it is used anymore). Again, the submit button is used to proceed with editing the record on the server's database.

10.6.3 Delete Notification

If the user decides to delete a notification by selecting the delete button, it is required that at least one notifications radio button is selected. This indicates to the application what records are to be removed.

```
for(int i = 0; i < notificationList.size(); i++){
    NotificationPanel panel = notificationList.get(i);
    if(panel.removeRadioButton.isSelected()){
        removeNotification(panel.getName()); //call method to remove
        displayPanel.remove(panel);
        displayPanel.revalidate();
        displayPanel.repaint();
    }
}
```

Figure 34: Delete Selected Notifications

When the delete button is pressed the jpanel is removed from the list and the database is accessed. The request to remove a record is achieved by indicating the unique ID that exists within the record. This results in both the front-end and back-end having removed the notification. The figure above shows that the list of notifications is looped through, with their radio buttons being checked for selection. When a radio button is selected a method to remove it from the database is called along with having it removed from the user interface. The interface is then repainted to show the user that the notification has been removed.

10.6.4 Error Handling

During the update/create methods, when a submit button is used error checking ensues on the user entered information. Whilst adding new notifications if any of the required fields are empty then the request doesn't go through. This is to prevent records with null values being sent and stored within the

database. Whilst updating a notification multiple checks are completed. A check for a selected notification is performed on the combo box to indicate which record from the database is being updated. If no selection is made, then no update query is run. Error checking also exists on the user input text fields. These simply check to ensure no fields are left empty as that would return empty values to the database.

10.7 Productivity Feature

10.7.1 Overview

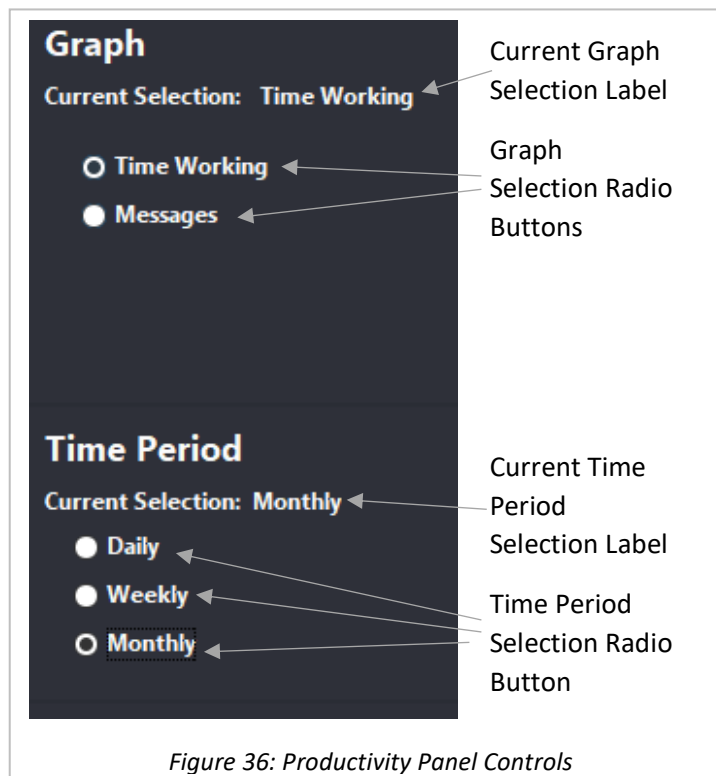
The productivity feature displays information to the user to allow for self-evaluation. This information includes: the number of messages sent in each login record within a given time frame, and the amount of time spent working within an outlined time period. This information is displayed in graphs, with the type corresponding to the type of information being presented.

Separate classes are used to generate the controls and graphs for the feature. This results in a dynamic method of generating updated graphs. To create the interface for the entire feature a Grid Bag layout is used to have the controls on the left and the displayed graph on the right.

10.7.2 Libraries Used

To create charts for the data visualization aspect of the application, “JFreeChart” is utilized. This is an external library used to represent data in numerous variations of typical graph formats. From this library, objects allowing for pie charts and bar graphs were utilized. Also, data set objects were used from the library as they have been created for the purpose of loading data into these charts.

10.7.3 JPanel: Controls Interface



The controls panel consists of two child panels; one for the time period selection, the other for the information set to be displayed. Both sections contain a set of radio buttons, each grouped together. Radio buttons are grouped to allow for only one, from within said group, to be selected at any given time, thus ensuring only a single time period and graph type has been selected. It is important to note that the

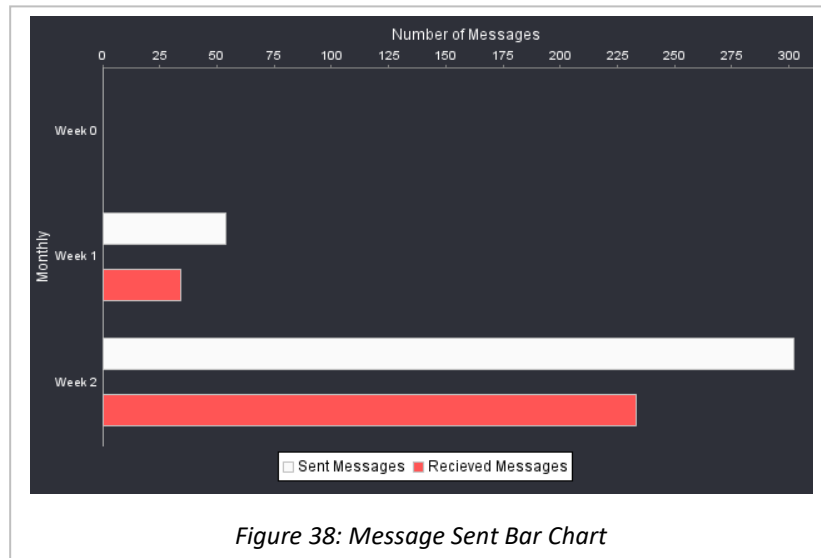
time period has a default assignment. This results in the user only being required to select a type of information for the graph to load.

```
if (e.getSource() == controls.timeWorkingRadioButton){//if the source
    graph = "TimeWorking";//set selection to time working
    controls.chosenGraphLabel.setText("Time Working");
}
```

Figure 37: Radio Button Click Listener

A click listener is used in the main class to listen for activity from the user when selecting the radio buttons. As in the figure above the listener checks the source and since each radio button corresponds to an individual selection, the selection variables are updated to state the new choice. After an update is made, no matter the source, the listener will trigger the graph panel to be repainted to display either then new graph, the new data, or both.

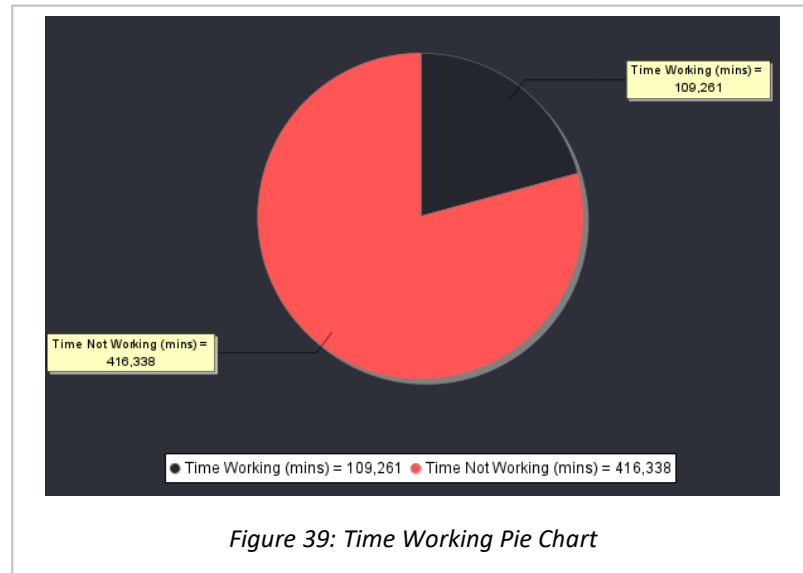
10.7.4 Messages Chart



The messages chart loads its information by calling a method which returns a filled “Default Category Dataset” object. This object is the data set container found in the external library and is filled by the following steps.

1. Create a local copy of the login records table. Allowing for easier reading of the table.
2. Call restrict method to reduce the login records table down to a state where only records within defined parameters are accepted. This restrict method does the following;
 - a. Depending on time period selection, a variable indicating either a day, week, or month in milliseconds is created. Also, a variable indicating the parameter in milliseconds is created.
 - b. Then two more variables are created; one to indicate the current time in milliseconds, and another stating the parameter in milliseconds.
 - c. Then a loop is started, iterating through the login records table.
 - i. If the indexed records start and end dates are within the two parameters, then the record is added to a temporary table.
 - d. With all useable records included in the temporary table, return it to the point of origin.
3. With only useable records included in the new table a list is created to hold the dates from these records. These dates are extracted in a loop which iterates through all records in the reduced table.
4. A switch statement is used, based of the time period selection made by the user. Depending on the selection the dates are grouped from the newly created dates list. Group these dates into a nested list (a list holding separate lists) as this allows for clear identification of the separate groups.
5. Once all records are grouped, another loop is run. This loop iterates throughout the list retaining the separate lists and then loops through the group lists. From this the number of messages sent and received are extracted from each record and totaled up in each group. With each group having a variable holding the total number of sent and received messages, the values are added to the dataset.
6. Finally, the dataset is returned to the point of origin.

10.7.5 Time Working Chart



The time working chart loads its information by calling a method which returns a filled “Default Pie Dataset” object. This object is the data set container provided through the external library. To fill it the following steps are taken.

1. Create a local copy of the login records table. This makes it easier to read through the tables.
2. Create a calendar object and then obtain the current time. When a calendar object is created, it is set to the current date and time by default.
3. Filter through a switch statement based off the selected time period.
 - a. If set to “monthly” then set the calendar object to one year in the past. This will allow for 12 groups of data to be displayed in the bar chart as each group is representing a month.
 - b. If set to “weekly” then set the calendar object to one month in the past. This will permit 4 groups to operate in the bar chart. Since the time period is weekly, having a group for each week from the last month is acceptable.
 - c. If set to “daily” then set the calendar object to one week in the past. This allows for 7 individual groupings due to there being seven days in a week. Therefore, the bar chart will have seven record sets of data applied.
4. Now that the furthest back date is now indicated via the calendar object this becomes our parameter. Any dates from the login records table found past this parameter will not be included in the data set. A loop is ran for the length of the records table to uncover the useable records, with the following occurring internally;
 - a. Create a date object and store the currently indexed login records start date.
 - b. Create a date object and store the currently indexed login records end date.
 - c. Create a date object and store the time from the calendar object. This fetches the time from the calendar object that defines the parameter. This is done to allow for comparisons between the record dates and the parameter dates.

- d. Check if the record start date is greater than the parameter date. By using the “getTime()” method on these dates its simple a comparison of milliseconds. In the event the outcome is a positive number, this means success so perform the following.
 - i. The length of the login record can be calculated now by taking the end time and subtracting the start time. As this is in milliseconds, we are left with the length of the login record.
 - ii. Add this time difference to a list of useable records.
5. Now loop through the list of useable records and add the individual times together into a total time variable. This variable holds the duration of all the login records combined.
6. Retrieve the total time that has elapsed from the parameter till the current time. This includes time worked and not worked.
7. Calculate the time not worked by removing the total worked time from the total time elapsed. This results in having two variables to utilize; the time worked, and the time not worked.
8. Finally add these two values to the dataset and during this process divide them both by 60,000. This is done to convert milliseconds to minutes.
9. Return the data set to the point of origin (end the method).

10.7.6 Implementation Issues

An issue with the development of this section of the application was attempting to make use of dates. Doing calculations with date and calendar objects was difficult due to these being dates rather than values, like found in typical math. To fix this the date objects internal method “getTime()” is used to obtain a long variable which is the date represented in milliseconds.

10.8 Chat Feature

10.8.1 Overview

The chat feature is key to this application as it solves many of the problems that exist with modern remote working tools. The chat allows for users to communicate with other users within the same company. Like most other communication services, the chat window is dynamically updated to show the most recent version of the communication between two users.

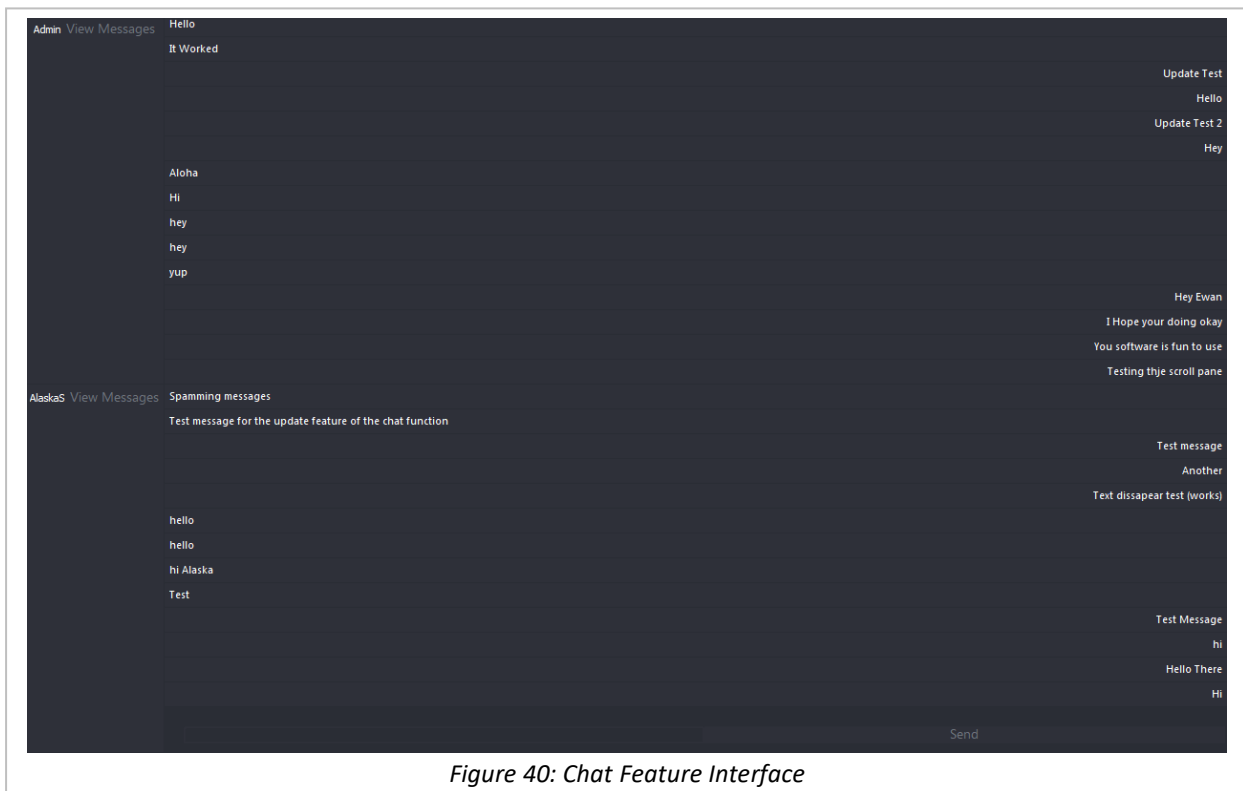


Figure 40: Chat Feature Interface

This class extends a JPanel allowing it to contain multiple components. To create this page a grid bag layout is used. This allows for smaller jpanels to be added resulting in having sections for different aspects of the feature.

10.8.2 JPanel: Friends List

The friends list resides within a JPanel. It utilizes a vertically oriented grid layout, this creates a single column list that sets each container, holding data on a user, to be the same size. The container for each user contains the users name, a link to any provided socials, and a button that the user can press to expand

```

JButton change = new JButton("View Messages");//creates a new button which will change the currently displayed
buttonStyles.applyButtonStyles(change);
change.addActionListener(new ActionListener() { //create listener for the view change button
    public void actionPerformed(ActionEvent e) {
        chatPanel.removeAll();//clear the chat panel
        createChatPanel(thisUserID);//recreate chat panel with the new selected userID
        chatPanel.revalidate();//revalidate the contents of the panel
        chatPanel.repaint();//repaint the components
    }
});

```

Figure 41: View Messages Button Action Listener

the messages for that specific friend. In pressing the button labeled “View Messages” the messages are loaded from the database and loaded to the screen.

An action listener activates once any action has been performed on the button, such as a press. The listener will; clear the current chat panel of all components, call method to recreate a chat panel with the indexed users unique ID being passed as a parameter, and will then repaint the chat panel to update its display.

10.8.3 JPanel: Message Display Panel

The messages panel is encapsulated within a scroll panel. This allows for the messages list to be scrolled through instead of all the messages being crammed into a single space. This section of the feature utilizes the array list, created by the database class, containing the database tables. The chat table is extracted first, then any records that contain either the selected friend's unique id or current users id. These ids can be found in both the “SenderId” field and the “RecipientID” field as seen in the figure below.

ChatID	SenderId	Message	RecipientID	Time Sent
□[?3□	IYOXL	yup	BKKGF	2021-02-22 17:39:24
□??VP	IYOXL	Test message for the update feature of the chat fu...	BKKGF	2021-02-22 17:56:52
□?□[?	BKKGF	You software is fun to use	IYOXL	2021-02-22 17:51:02
□????	EwanD	Hello	BKKGF	2021-02-09 22:35:25
□□??	EwanD	Hello	BKKGF	2021-02-09 22:41:30

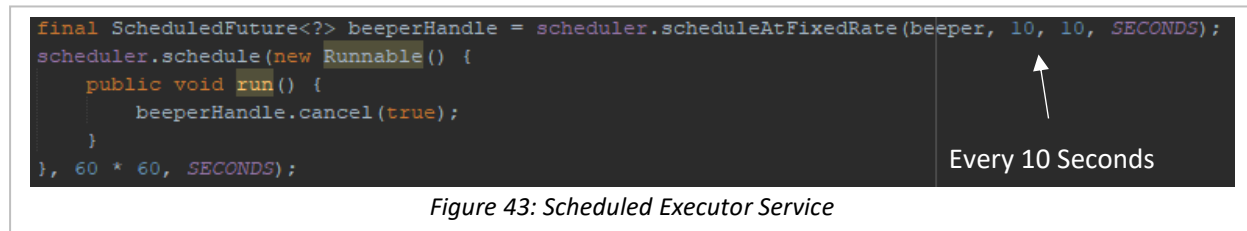
Figure 42: Database Chat Table Records

When a record is found, it is added to the messages list to be displayed on screen. Once the entire chat table is search through, with any desired records being added to the list, the chat panel receives an update to correctly display the message. Since the database class loads the chat table ordered by the time the message was sent, these messages appear in the correct order without any additional manipulation.

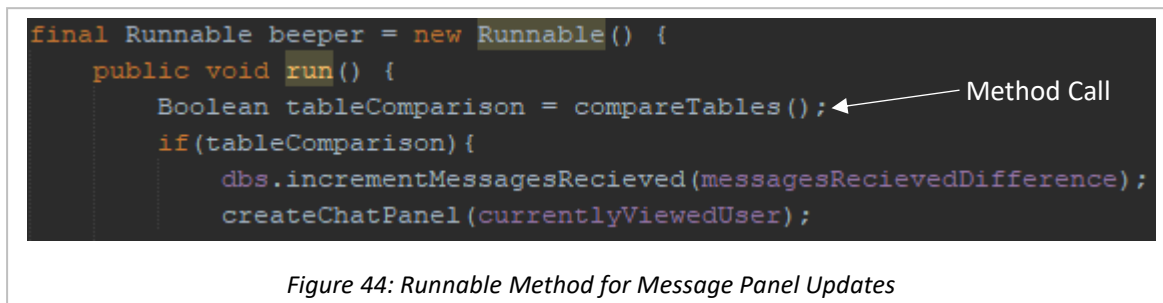
10.8.4 User Input

This panel takes the users input and allows them to send it to the database to be stored. Two components exist in this section, a text field for the user's input, and a submit button to send the message. When the message is submitted, a connection is made to the database using the typical method for connecting. This is outlined in the database class section of the report. If the connection was made successfully then the new record is added to the database.

10.8.5 Update/message Checking



To check if the messages panel needs updated a check must be ran every few seconds while the chat window is open. To conduct this a “Scheduled Executor Service” is used. This allows for code to be ran every defined number of seconds, in this case every ten seconds. The benefit of using this method instead of a more typical method, such as a thread with the sleep method, is the rest of the application can still run code rather than freezing while waiting for the next check.



To check for an update a method call is used to return a boolean. This method is called to obtain the most recent version of the chat table from the web server database, and to then compare it against the locally stored chat table. The returned boolean represents whether the local copy of the table is different to the one stored on the web server. If there is no difference, then the next wait and check is started. If a difference does exist, the local chat table is updated along with the messages panel. This results in the new message appearing on the screen.

10.8.6 Error Handling

When a user submits a message to be uploaded to the database the message will only send if the text field is not empty. This is to prevent any empty records being uploaded, and in turn to prevent any empty messages appearing on the message panel.

10.8.7 Implementation Issues

An initial problem was the entire feature being updated when a new message was received. The issue is this make the user interface change in size, causing the entire chat window to visually look different with every new message received. The fix for this was to separate each section of the feature into individual sections. This allows for only the messages to be updated.

10.9 Drop Box Feature

10.9.1 Overview

The drop-box feature makes use of a library for accessing the api (application programming interface) provided by the Drop Box tool. This tool is created by a third party and grants access to a free file system. This file system is created separately to the company database, so to make use of the feature all companies must register their own account with Drop Box.

10.9.2 Connection

To connect with the drop-box file system three important components are required.

- A drop-box client object for accessing the files within an account.
- A drop-box request object for connection to the file system.
- An access token, provided by the registered account, that acts as a unique connection code.

A connection is created by initializing the request object to indicate the path required to make a connection to the Drop Box service. The client object is then initialized with the request and access token resulting in the client being a direct connection to the file system.

10.9.3 Reading and Writing

Once the connection is created, using the client object, a list of meta-data is produced. This list contains records of information on the contents of the file system such as files and folders. With this list information can be drawn out such as the name and type of the file. This information can then be displayed in a visual form for the user.

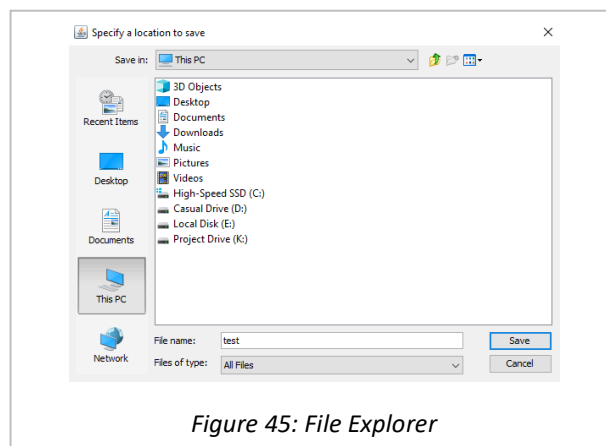


Figure 45: File Explorer

To upload a file the file manager on the user's system is opened, this allows the user to select a file. Upon selection the file is passed to the "Upload File" method which puts the file into a "File Input Stream". This reads the data from the file and holds it internally. The client object is then accessed, along with methods from within the drop-box library, to upload the data as a new file on the file system. It takes the old files name, essentially creating a duplicate version.

To read a file the user is able to select the desired file and select the download button. This will open the file system to the user and allow them to select a location and new name for the file. This allows users to easily access documents that any user requires. Utilizing this in tandem with the ping feature grants all those involved on a task a means to seamlessly perform code reviews.

```
//obtain access to the drop box file
String filename = panel.name;//obtain the name of the file via the panel object
URL download = new URL("https://www.dropbox.com/s/jo5nx5risfa6vte/" + filename);//create url for
ReadableByteChannel rbc = Channels.newChannel(download.openStream());//create a channel to read
```

Figure 46: Dropbox Connection

The downloading of files and folders is done by establishing a connection to the Dropbox account, this requires a URL and a “Readable Byte Channel” which allows for reading at the indicated link. Afterwards reading the selected document into a “File Output Stream” which allows for a quick way to copy documents. Utilizing this with a popup showing the users internal file explorer grants the means of saving the document to any location.

10.9.4 JPanel: Interface

To display the available documents to the user a jpanel is created for each file and folder. This panel contains the type and name, these are represented by labels. To create this panel another class is used with all panels extending it to maintain consistent styling. This class extends a jpanel and, using the

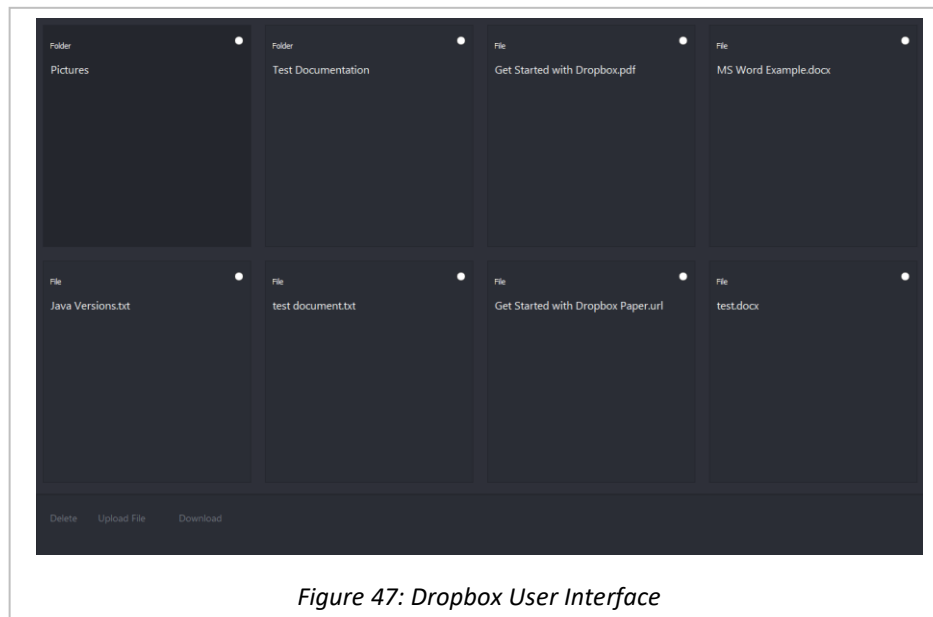


Figure 47: Dropbox User Interface

NetBeans editor, has public components that allow for relevant information to be inserted. By having the labels set to public the content of the labels can be dynamically updated through a reference to the object.

With the individual panels created they are then displayed in a grid layout. This layout only defines the number of columns, this allows for any number of rows to be created depending on the number of documents added to the layout. To accommodate the random number of documents that could exist in this user interface the content is enclosed within a scroll pane. The scroll pane allows for the displayed documents to be scrolled through if the given number exceeds the amount able to fit within the screen

bounds. In the case of this layout, the maximum documents that can be displayed without the scroll pane being required is eight.

10.10 Feedback Feature

10.10.1 Overview

The feedback section exists to display all the feedback provided for any tasks completed by the user. This is achieved by loading an interface to hold all the records, with a separate class being used to generate the visual copies of the records. The ability to edit the feedback is also given to the user.

10.10.2 Jpanel: Interface

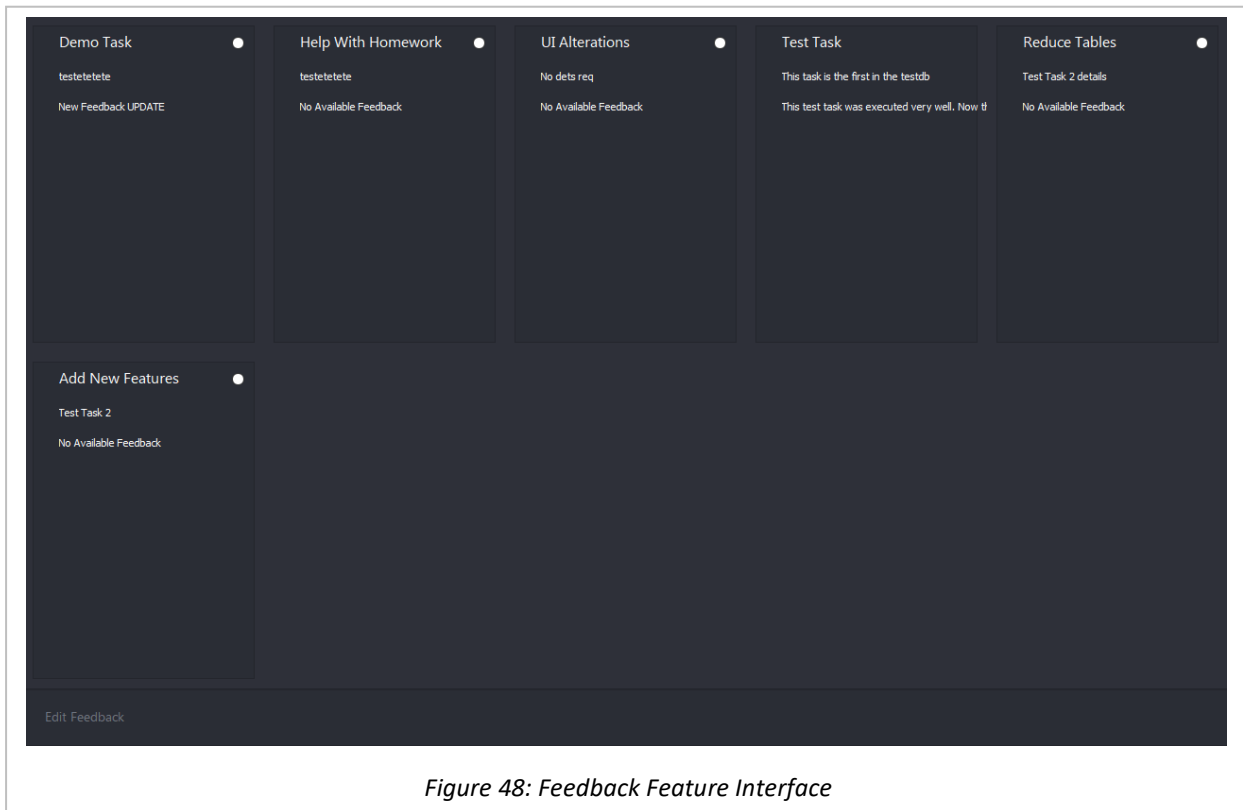


Figure 48: Feedback Feature Interface

Each feedback record is obtained through the feedback field in the task table. The feedback records are created by accessing a separate class that extends a jpanel. The name, details and feedback for each completed task is passed into the constructor during the creation of each panel. An internal method then is called by the constructor to update the labels that exist on the panel. These labels contain the parsed information. If no feedback has been assigned to the task, the label is set to “No Feedback Available” to inform the user. A radio button is also attached to the record panels to allow for individual selections. These radio buttons allow the user to edit any selected records.

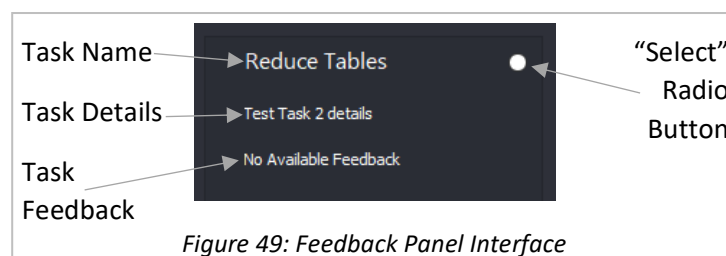
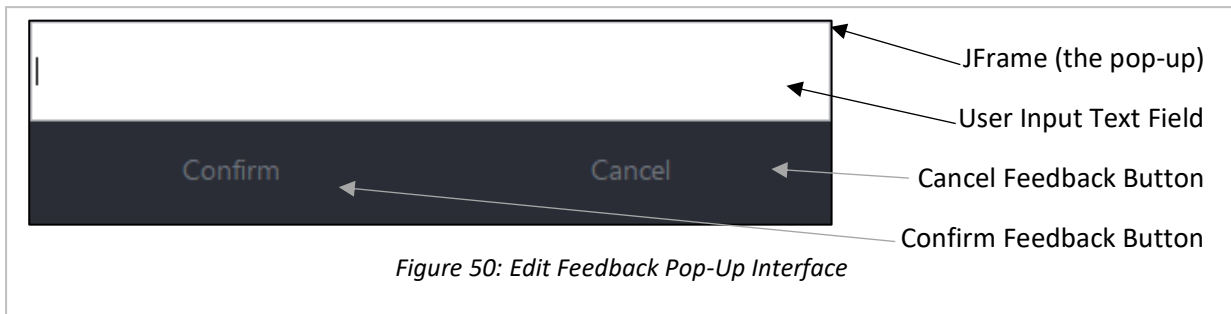


Figure 49: Feedback Panel Interface

10.10.3 Edit Feedback Pop-Up

To edit a feedback record, the user can select the radio button contained within the panel and then press the edit button that exists on the controls panel. This opens an external JFrame that acts as a popup window. This contains a text field for the users input and two buttons; one to cancel the update, and another to confirm the user input and update the feedback.



Once the confirm button is pressed an update method is called to update the specified record on the task table. This connects to the database and then calls an update query to edit the feedback field. If the update is successful, the indexed panel is then removed from the user interface. This is done to temporarily amend the displayed records until the local copy of the database tables is updated again.

10.11 Settings Feature

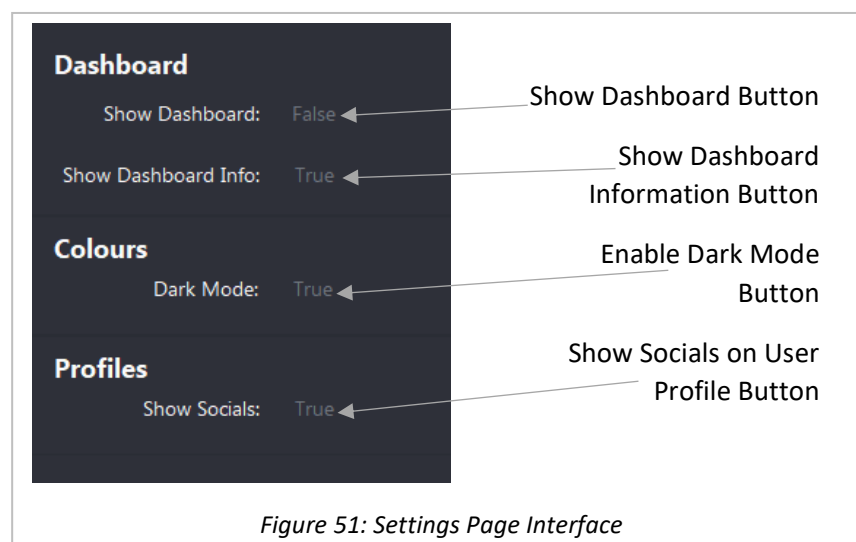
10.11.1 Overview

The settings page exists to allow the user to tailor parts of the application to their needs. Settings are saved in a text file which is read at the start of the applications lifetime. This file is then updated and saved whenever one of the settings is changed by the user. The available settings include the following.

- Skipping past the dashboard section of the application.
- Disabling the information section of the dashboard.
- Enabling dark mode on the application.

The ability to remove the dashboard allows for the user to navigate directly to the login page upon the applications initialization as the dashboard provides nothing to them. Disabling the information section on the dashboard is possible to hide data regarding the newest updates in the tool. This exists for users who have no interest in the newer features available. Switching between the default dark mode and light mode exists to give users the ability to customize the tool to fit their visual requirements.

10.11.2 Jpanel: Interface



The user interface for the settings page consists of a list of labels representing the settings from the file with its current value alongside it. The settings value is displayed in a button, indicating to the user to press it to flip the value. When a button is pressed the value displayed on the button is flipped and the value is updated within the file.

10.11.3 File Storing

The settings are saved locally to the user's computer within a text document. This document is referenced through a file object with a scanner being used to read through the document line-by-line. If the application cannot find the settings document in the root directory of the project, then it is assumed that the file does not exist. In this scenario a new file is created containing the settings with their default values being assigned.

The document contains settings that are held within strings. Each setting occupies a single line in the file, with all settings maintaining the same style. This style consists of having the setting and its value separated by an equals sign.

When the user clicks an available settings button, to flip the boolean value correlating to the desired setting, a method is called to search for the indexed setting. Once the setting is found the boolean value is read, an additional method is then called to flip the value of that setting, and to then save this updated setting to the file.

10.11.4 Error Handling

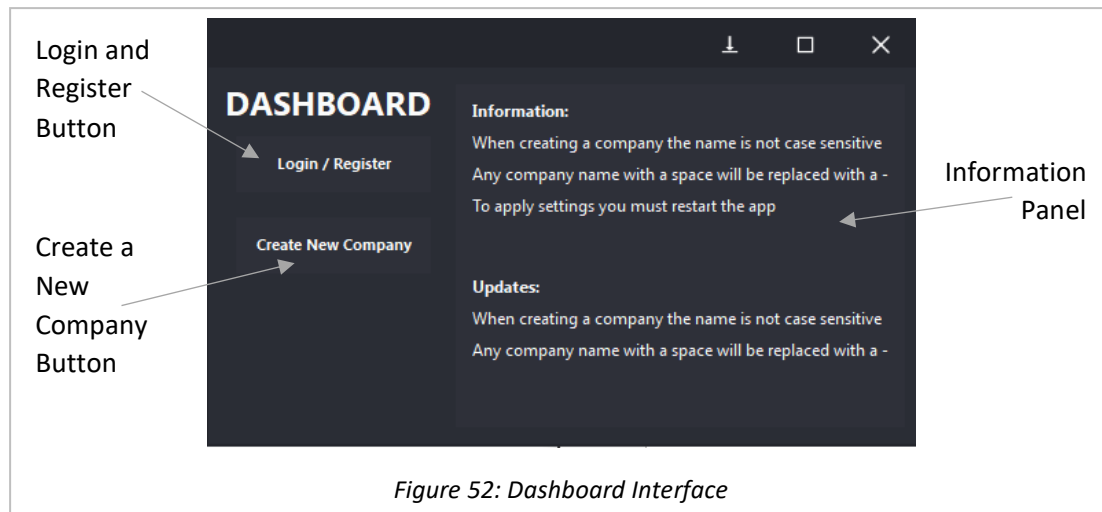
When a user submits a message to be uploaded to the database the message will only send if the text field is not empty. This is to prevent any empty records being uploaded, and in turn to prevent any empty messages appearing on the message panel.

10.11.5 Implementation Issues

An initial problem with this feature was the entire feature being updated when a new message was received. The issue is this make the user interface change in size, causing the chat to visually look different with every new message received. The fix for this was to separate each section of the feature into individual sections. This allows for only the messages to be updated.

10.12 Dashboard

10.12.1 Overview



The dashboard class extends a jframe resulting in it being an independent interface. It is the first user interactive panel that the application loads. This section allows for the user to navigate to the login and register forms, or to the “create company” form. On the dashboard an information panel is displayed, this informs the user of any updates that have been made to the application. As the dashboard provides the ability to create a new company database it may be seen as a needless step to take while navigation to the login page. To fix this, in the settings page the user can disable the dashboard. This results in the application skipping past this section and navigating directly to the login panel.

10.12.2 JPanel: Information Panel

The information panel exists to inform the user of any changes that have been made to the application. Ideally this panel would take in information from a table that exists out with the companies database. This allows for multiple records to be read and dynamically inserted into the application without the companies having any access to them. This also allows for the information to be easily updated and inserted into the dashboard. However, the current method of inserting the information is by editing the text area in the NetBeans interface editor.

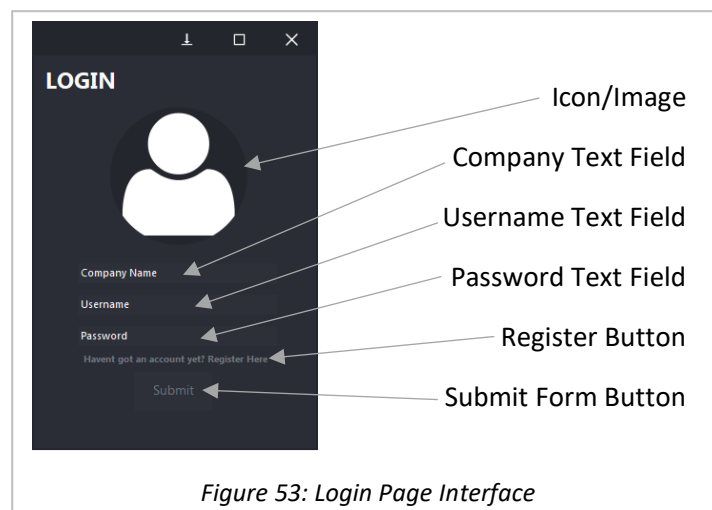
10.13 Login and Register

10.13.1 Overview

Both the login and register forms exist in the same JFrame. This is to keep the application access forms in the same space. To load this complete JFrame both the register and login Jpanels are created fully with all components initialized. Then only the login panel is applied to the frame, while the register panel is left inactive. This can be switched when the user interacts with the button asking if they require an account.

10.13.2 JFrame: Login Form

The login form is used to obtain access to the application. From the users input an attempt is made to connect to the web server's database management system. If the user is successful, they are able to further navigate into the main application. If unsuccessful, a notification is displayed to the user in red.



To make a login attempt a connection is made to the company database through the web server. This means first the user must login to the web server using the entered information, if a matching account is found then the process is conducted on the company database. If the account is found in both the web server and the database, then the application continues to the next stage.

10.13.3 JFrame: Register Form

The register form takes user input to create a new account in the indexed companies database. The first step taken is to verify the user's input. This is done by checking that the entered company code exists, and that both entered passwords match. If either input checks result in an error the user is informed using a red text notification to clearly inform the user to make another attempt. If the check were a success and all the user entered information is okay, then the application tries to register an account. This attempt requires a connection to the database using the typical three parsed variables. To register, an account must be created and stored on the web server and on the company database. This is to allow for users to login to both the server and then the database. To ensure that these user accounts match, the query used to insert an account into the server accesses the data already stored in the company database. This guarantees the information will match.

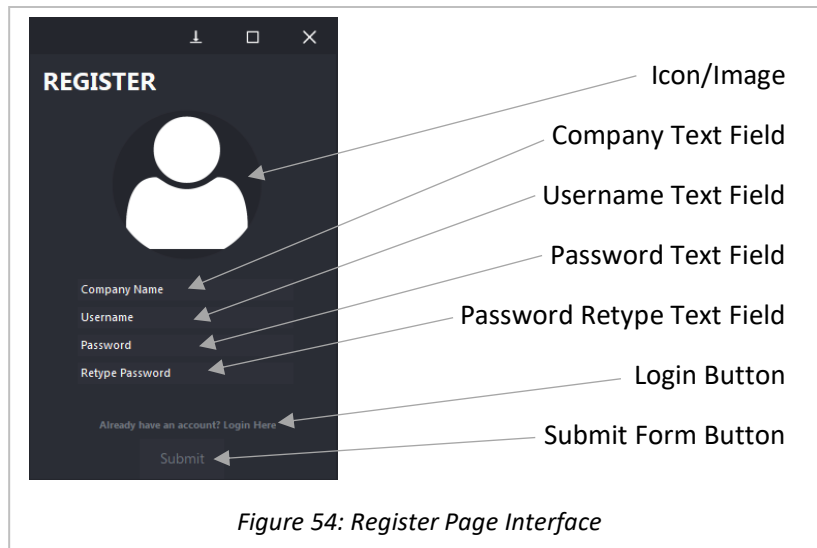


Figure 54: Register Page Interface

Once the user account is added to the server, privileges are granted to the account. These privileges allow for the account to insert, create, and update the database that corresponds to the entered company code. This is done to prevent every user having the ability to fully modify the database.

10.13.4 Responsive Components

In both forms the components react to user input. The text fields that accept user input react by clearing the default text that indicates where specific information should be inserted. This text is cleared when the user clicks on the text field. The default text is returned if the user exits the text field without having entered any data. The button for submitting user entered data also reacts to the user's cursor. When the cursor "enters" the buttons space its background is highlighted to indicate to the user that they are able to now select the button. When the cursor exits the buttons space this highlighting is undone to return the visuals to default.

10.14 Database Class

10.14.1 Overview

The database class is initialized after the user has logged into the application. Upon the classes initialization it is passed three variables of the string data type. These variables hold the required information for accessing the company database through the web server. This includes a database name, a username, and a password. These are then combined in a command to connect to the database, if successful then the remained of the code is ran, if not then the application is not loaded.

10.14.2 Loading Database

This class is tasked with loading the database from the web service and storing it in a list of multidimensional arrays. It has been designed to allow for different database structures so changes can be made without any negative implications during runtime.

In loading and locally storing the database into the program there several steps to follow. These steps include.

1. Connect to the company database.
2. Loading the names of the tables into an array list.
3. Counting the number of tables.
4. Begin a loop to iterate through database tables.
 - a. Create a statement object to hold the query.
 - b. Insert query to read specific table.
 - c. Run statement with query to obtain a result set with data within.
 - d. Call method to calculate the number of rows in result set.
 - e. Call method to calculate number of columns in result set.
 - f. Create a multidimensional array frame using the previously calculated number of columns and rows.
 - g. Call method to fill the 3d array with data from the query.
 - h. Add table to the array list of database tables.
5. Close statements and connections.

The result of all these steps is an array list holding individual multidimensional arrays, which each correspond to a different table from the database. This is then returned to the applications initialization to be passed and used throughout.

10.14.3 Login Records

The database class is used to retain and upload data correlating to the current login record. A login record is created for use in the productivity feature to measure multiple statistics. The class contains a method that is called to begin the login record, this is done by uploading a half-created record with a number of the fields left as null. Upon the end of the application the record is then closed by uploading the remainder of fields to the database. This is triggered in the system exit class which is described further in this report.

10.15 System Exit Class

10.15.1 Overview

This class is called when the application ends. Similar to the database class this class is passed three variables used to connect to the database being hosted on the web server. However, a fourth parameter is passed, this being a reference to the database object. This grants access to the database class's public variables for use in completing the record.

A connection is made and then the method for finishing the login record is ran. To obtain the required information for the record to be complete, such as "number of messages sent", the data is obtained through the database object. A query is then run to update the indicated record, indicated by another variable located in the database object.

10.15.2 Implementation Issues

An issue that is to be solved with this occurs in the event of a crash. If the application crashes the system exit class is never ran because the application requires a pre-ordained close. This results in the half-completed login records being left on the database. The productivity feature that makes use of these records does have internal error handling to deal with a scenario such as this, but still the records exist.

11 Testing

11.1 Methods Used

The two testing methods being used are integration testing, and exploratory testing. Integration testing is conducted by software developers to ensure that individual sections of the application meet the intended design and behavior. Typically, with integration testing the individual sections of the application are evaluated, this includes evaluating the design and functionality. To perform exploratory testing a group of individual users is selected to perform actions on the application without any prior experience. A script is provided to outline the tasks to be performed, and a questionnaire is provided for these users to review the application in relation to the tasks. Each review is in the form of a score which is then put into a graph for visualization.

11.2 Internal Testing

Dashboard	<p>This page has two buttons, both performs actions to navigate the user to another form. These buttons work as intended as the first successfully takes the user to the login page, the second takes the user to the “create new company” form. The design for these buttons is also successful as they properly react to user interactions such as mouse hovering.</p> <p>The information panel that exists also functions as desired. The retrieval of text from an individual table works as intended as the panel contains the update.</p>
Create New Company	<p>This page consists of a text field and button. The designs for both these user inputs as intended. The functionality of the text field also works as intended, more specifically the responsiveness. The submit buttons responsiveness also has the intended effects.</p> <p>In clicking the submit button, the entered information should be taken and made into a new company database, this is done successfully with the login form being loaded afterwards.</p>
Login	<p>The login page contains a group of text fields, all of which have the same designs, responsiveness, and purpose. The design of these text fields is as desired. The responsiveness is achieved as the default text is removed when clicked on and reloaded if the field is exited while still empty.</p> <p>The “Register Here” button has the intended responsiveness when the user hovers over with their mouse. It also navigates the user to the register page when click on, this is the intended purpose of this button.</p> <p>The submit button exists to send the information in the text fields onwards to establish a connection to the company database. This is successfully achieved when pressing the button. The design for this button is also as intended.</p> <p>The error message appears as intended when an issue with the user’s input is found. This message also successfully states the corresponding error.</p>
Register	<p>The register page shares the same styles as the login form. The responsive components do what is intended of them as they change appearance.</p> <p>The submit button successfully takes the users input and connects to the web server and then database to create the account. The user is then taken to the login form.</p> <p>The “Login Here” button successfully takes the user to the login form.</p>

	The error message appears as intended when an issue with the user's input is found. This message also successfully states the corresponding error.
Title Bar	<p>The title bar has three buttons. The close button does successfully exit the application. The full screen button does successfully stretch the user interface to fit the user's screen, it also allows for windowed mode to be re-enabled. The minimize button does as intended by shrinking the application down to the task bar.</p> <p>The style of all these components is as intended. Specifically, the full screen button changes the icon that exists on it to indicate a windowed mode is now available. The exit button has a colour change when the user's mouse hovers, just like the other buttons, however this button changes to red.</p>
Menu	<p>The menu panel has a list of buttons that all act the same way; however, they load different features to the content panel. The buttons all successfully load their corresponding feature. The design for these buttons is consistent with all in the menu, it is also as intended as the responsiveness functions properly.</p> <p>The task drop down arrow does function as intended as it displays the task names, and changes the icon depending on whether the list is showing. The dropdown arrow however does not have a hover effect applied to it, resulting in it not being greyed out when not in use.</p> <p>The list of tasks made available by the dropdown work as intended as they successfully load the information on the specific task.</p>
Tasks	<p>The task panel works exactly as intended. The individual tasks are displayed in the desired table format with the desired relevant information being displayed. Each label is responsive as it changes to a brighter colour as the user's mouse hovers over it. Upon selecting the label, the task it belongs to is then opened and shown to the user.</p> <p>The design of the panel is as intended as the header panel is indicated through the darker colour. The content also has dividing lines along the x-axis to outline the separate task records.</p>
Add New Task	<p>The page for adding new tasks contains multiple components such as text fields, date pickers and combo boxes. All these components have the desired styling besides the text fields. These have not got the styles that have been applied to other text fields from other sections of the application. The submit button has the planned design as the responsiveness is applied to highlight the text upon the user's cursor hovering on top. There is no responsiveness on the user input because the required data is indicated next to the input component.</p> <p>When the date picker is selected a calendar window is opened to select the date. This is planned as an external library is used for this purpose.</p> <p>The submit button will upload the new task to the database but will not navigate the user to another location.</p>
Specific Task	<p>The specific task page displays all the relevant information in labels, besides the active field. The active radio button exists to allow the user to change the status of the task. When pressed the button is flipped to un-selected, the code behind this runs as anticipated. The code updates the record on the company database and therefore changes the task page when it is reloaded.</p> <p>The designs of this page are simple, but they look as planned as they match the style found consistently throughout the application.</p>

Calendar	<p>The calendar panel contains the controls and the calendar in separate sections. The calendar is displayed in the intended manner as each day is a responsive panel which is highlighted from the user's mouse. The tasks are also displayed on their submission date, they too are responsive as they are highlighted when the user hovers their cursor over them. When a task name is selected it then opens the information page in that specific record.</p> <p>The controls of the calendar function as intended as they navigate the calendar both forward and back months or years at a time. The design of this section is not as intended. The individual components have the desired aesthetics however they aren't in the intended layout/positions.</p>
Notifications	<p>The notifications page contains an individual panel for each notification and a controls panel. The controls panel has two buttons, one to create/update and a second to remove any selected notifications. Both buttons share the same style and responsiveness as they both highlight when the user's cursor is hovering. The create/update button will, as intended, navigate the user to the edit notifications form. The remove button will remove a notification from the user interface and the database if said notification has been selected.</p> <p>The individual notification panels each contain one user interactive component, the radio button. This button looks and functions as planned as it will add the selected notification to the list of notifications to edit.</p>
Create/Update Notification	<p>Both creating and updating notifications occurs on the same page. The components of both sections have the same designs and responsiveness applied to them. The user input fields do not have any responsive attributes applied to them as no default text is utilized. The styling for these components is accepted but not fully as intended as they do not match the rest of the application. The submit buttons have the planned responsiveness as they are highlighted when the user hovers their cursor over the top. They also function as intended as they attempt to establish a connection to the company database to add or alter notifications.</p>
Productivity	<p>The productivity page is split into two sections: the controls and the display. The controls panel has the intended designs and layout for components, and functions as intended. The displayed chart changes according to the selection made in the controls.</p> <p>The charts designs are not consistent with the rest of the application. However, this is due to making use of an external library to generate them, making it harder to style.</p> <p>The layout of the entire page is as intended however does not match the rest of the application.</p>
Chat	<p>The chat page contains a list of available users to message and then a list of messages that correspond to the selected user. The messages window displays the messages in a scroll pane as desired however the scroll window does not begin at the bottom of the panel. This would be ideal as the messages start with most recent at the bottom. The interface to send messages is below the messages window and it contains a text field that has the desired styling and has a send button which also has its intended styling and responsiveness.</p> <p>The users list displays the users in individual panels, the orientation is as intended but the layout is not. The panels stretch across the downwards list to fill</p>

	the container however the ideal layout would be to have a fixed size for these panels, with them not being stretched to fit.
Drop Box	<p>The drop box feature displays the content in individual panels. These panels contain components to display the relevant information. The components work as intended. The radio buttons on each panel can be selected with the delete button allowing for the removal of the document from both the user interface and the database.</p> <p>The controls panel that allows for deletion and uploading contains two buttons to perform these actions. Both buttons have the same desired style with the responsiveness working as intended. Upon pressing the upload button, as intended, the file explorer window is opened with the user being able to select a file. Once uploaded the page is refreshed to show the new document on the internal file system.</p>
Feedback	<p>The feedback page takes inspiration, for the design, from the drop box feature. Split into two sections; the controls panel contains buttons to edit notifications, whereas the display panel houses individual panels for each piece of feedback. The page does have the desired design and layout.</p> <p>The individual feedback panels have only one interactive components: the radio button. This radio button looks as planned and functions the correct way also. The controls panel has a button for editing notifications. As intended this button adds the selected records to a list in which the feedback can be altered. This edit button also has the intended design as it is responsive to the user's cursor.</p>
Edit Feedback	<p>This pop-up window contains three elements: text field, cancel button, confirm button. The text field has no designs or responsiveness applied to it; however, it does work as intended as it takes in the user input. The buttons have the intended design and responsiveness, and they function as planned. The cancel button successfully stops the editing process, while the confirm button successfully connects to the database and edits the indexed record.</p> <p>This popup does not match the style of the rest of the application but does have the intended design.</p>
Settings	<p>The settings page has a responsive button for each individual setting. These buttons all have the same design and functionality. The buttons react to having the user's cursor hover over them by highlighting the text. The text is switched between true and false after every mouse click.</p>

11.3 External Testing

To obtain testing from users a script has been created with a list of questions following. The script informs testers on what actions to attempt within this project, instructions are provided where necessary. The questions each relate to a section of the script. The users provide a score between 1(lowest) and 5(highest) to rate each section. The script and questions are in “appendix 1” at the end of this report.

Five individuals have been selected to evaluate the project in its current state. The following graphs state the results from the questionnaires provided to these testers. Each question has its own set of results. The results will be provided along with the corresponding question.

Question 1: Login to the application using the example account.

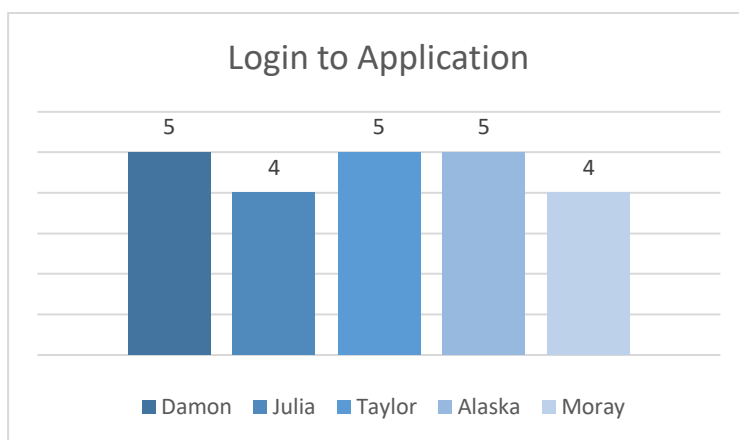


Figure 55: Question 1 Results

Question 2: Create a New Task.

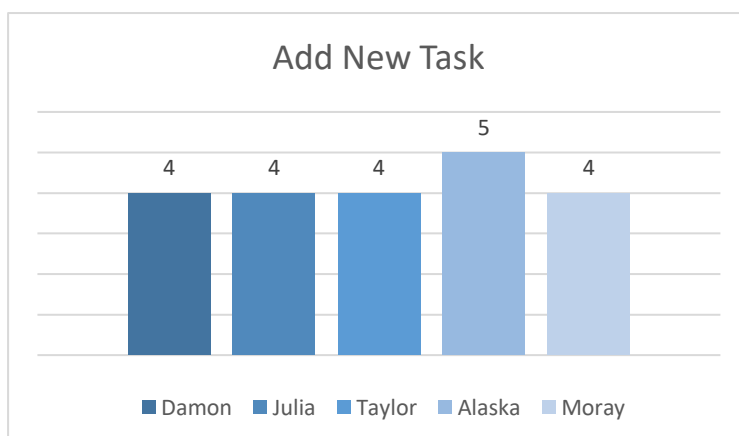


Figure 56: Question 2 Results

Question 3: View the New Task on the Calendar.

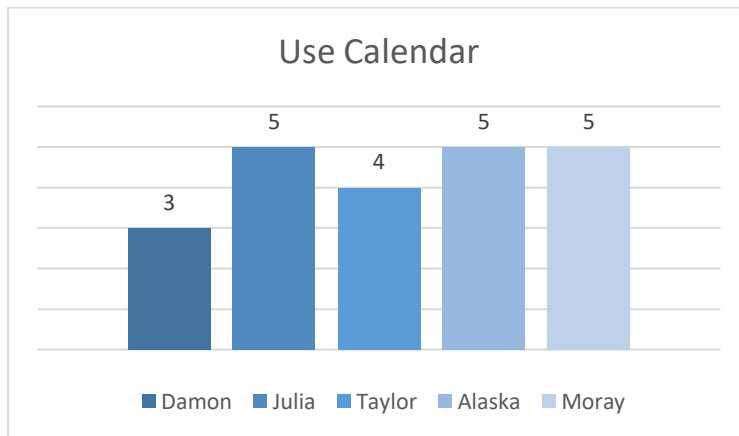


Figure 57: Question 3 Results

Question 4: Send Message to "AlaskaS".

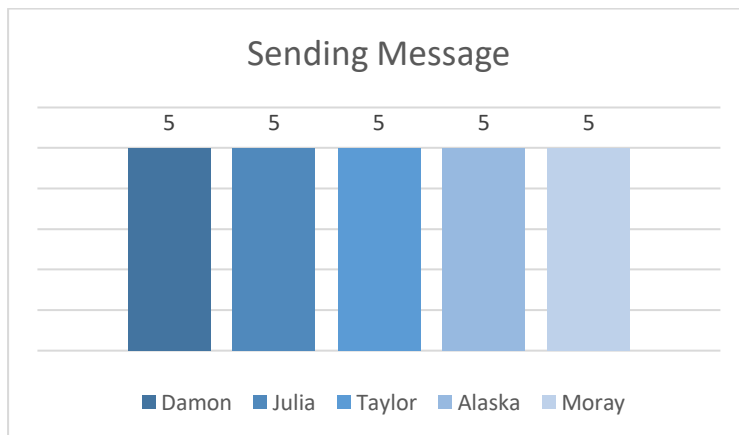


Figure 58: Question 4 Results

Question 5: View Messages Sent in Productivity Charts.

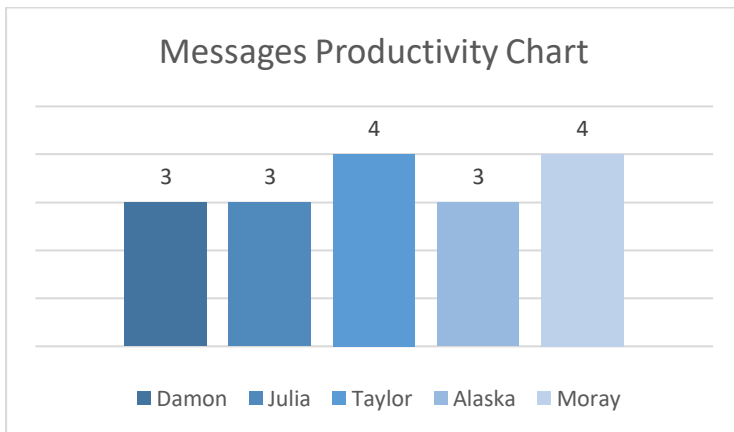


Figure 59: Question 5 Results

Question 6: Create a New Notification.

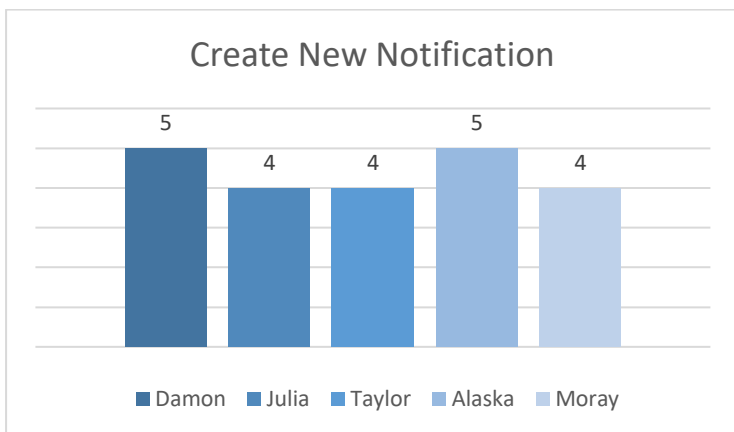


Figure 60: Question 6 Results

Question 7: Upload a File to Drop Box.

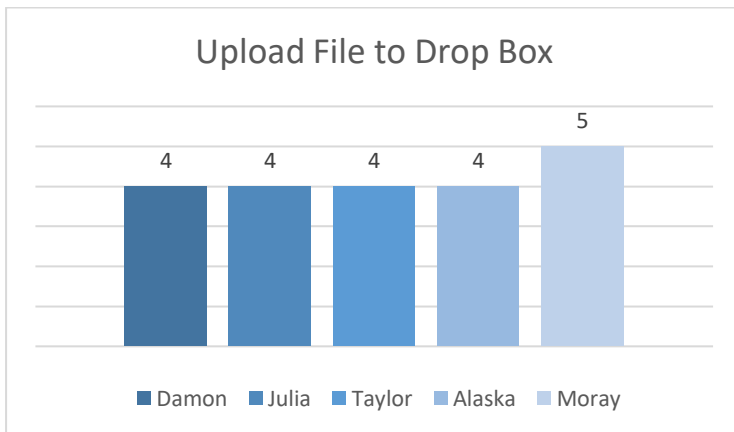


Figure 61: Question 7 Results

Question 8: Delete a File from the Drop Box.

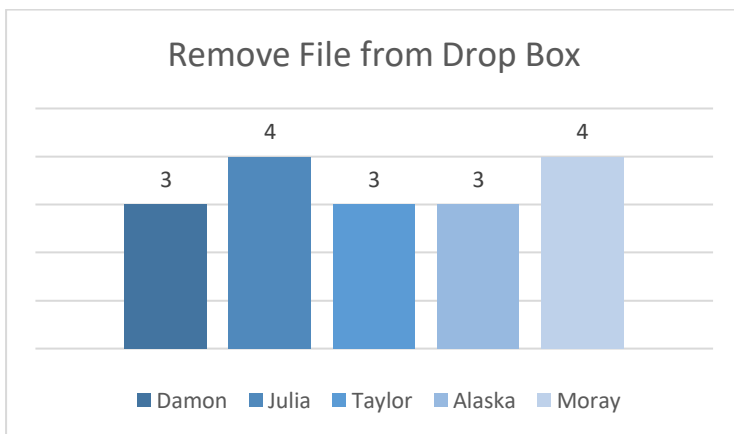


Figure 62: Question 8 Results

12 Conclusion

12.1 Evaluation

The final solution of this project is a working remote working tool that has been created with the employee being the center of attention. This application runs on windows operating systems and connects to a web server hosted on a computer system. The application is built to fit on the typical screen resolutions that are found on computer systems, however the application can be run in windowed mode at a fixed resolution. Though it is a working application it is currently not available for distribution due to the nature in which it has been developed. The application was created through a local system using Apache web server, this would need to be changed to a globally accessible server to allow for the tool to be utilized anywhere.

The problems with remote working are solved as.

- Distractions are limited due to having all required information available to the user in a centralized location. This results in having no need to navigate to third party sites to communicate or retrieve files.
- Visibility is increased as the work submitted is being reviewed by an employer through the application. The employer being able to see a list of tasks to be reviewed also aids with this process.
- Visibility is also aided by the use of the productivity feature. This feature granting users the ability to gauge their own workload and effectiveness in remote working will greatly help with visibility as increased workflow will be seen, measured in the quantity of completed tasks.
- Isolation effects are reduced due the communication channels available to others within the company. However, this could have been combatted further with introductions to alternative features.

The application successfully makes use of the Java Swing library which provides a wide array of visual components that allow for user interfaces to be made. This allows for user input to be read and inserted into the database through the use of connections made through the Apache web server. Java Swing components also allow for output to the user in various forms. "JFreeChart", an external library for displaying visualizations of data, was utilized to create various types of graphs for displaying measured information to the user. This being a strong factor in attempting to solve the problems outlined with remote working. A Drop Box library was also used to allow for connections to be made to the Drop Box server where the company file system is hosted.

Measure have been taken to guarantee the application is running smoothly by ensuring opened objects such as feature classes are closed when necessary. This keeps as few processes operating as possible to reduce the stress being put on the user's computer system. Due to the high number of database connections that are made, it was key to ensure that the following objects were closed after each attempt:

- Connection Objects (created to obtain connection to database)
- Statement Objects (created to hold a statement to make on the database)
- Execute Objects (created to hold a SQL query that is executed)
- Update Objects (created to hold an update SQL query)

Since the volume of objects required to access and operate on databases, all involved objects are closed after a successful or unsuccessful operation.

The final solution almost fully incorporated all desired features. However, the calling feature was not one of them. While calling would be a valuable feature to have incorporated into the final product, attempting to add this feature was a challenge for a simple reason; the project was developed as an ant project while all open-source libraries for a calling feature was designed for maven projects. A maven project adds the libraries direct to the project whereas ant project simple link them from the systems internal file system.

12.2 Plans

12.2.1 Changes

- Change the settings page to dynamically apply any changes made to specific settings. Currently the system only operates with the new settings once the application is closed. Altering this to apply changes in the moment would make the application more responsive.
- Having a calling feature incorporated into the application would have made it a more complete program as all features that were in the designs would have been completed.
- Developing the project as a maven project would have allowed for a wider selection of libraries to be used. This would have made development easier as completing features from scratch came at the cost of time.
- Simplifying the application down to a more specific purpose would have given the project a more refined feel. Instead of having a tool for remote working, having made the tool around making remote working easier would have been superior. Essentially making an application to complement existing remote working tools.
 - Doing this would have resulted in having more time for each individual feature.
- Making use of a visual library would make for a more consistent design throughout the application.

12.2.2 Future Work

- Mobile Application
 - Develop a mobile app version and have it connected to the computer version, so whilst working even the user's phone is being purposed to benefit them.
- Web Server Updating
 - Updating the web servers' databases to have a separate database for each user. This would result in data being easier to store and access, whilst keeping it more secure.
- Scrum Management
 - Incorporate more of the scrum management system into the application. By doing this the workflow would be improved as making the process of completing tasks smoother will benefit a user's quantity of completed work.
- Incorporate more features to combat isolation effects.
 - Additional features can be added to the application to further reduces the effects of isolation. Of the three main issues with remote working that this project set out to fix, isolation is the one that could benefit from further methods.

13 References

- [1] Deloitte. How covid contributes to a long-term boost in remote working, <https://www2.deloitte.com/ch/en/pages/human-capital/articles/how-covid-19-contributes-to-a-long-term-boost-in-remote-working.html>, March 2020.
- [2] Forbes. The Untold Side of Remote Working, <https://www.forbes.com/sites/ankurmodi/2019/12/26/the-untold-side-of-remote-working-isolation-and-lack-of-career-progression/#784c908568c7>, December 2019.
- [3] Raconteur. How Work Friendships could be the Secret to Productivity, <https://www.raconteur.net/hr/employee-engagement/work-friendships-secret-productivity/#:~:text=Work%20friendships%20can%20lead%20to,Business%20School%20Professor%20Teresa%20Amabile.&text=However%2C%20that%20camaraderie%20can%20pay,those%20with out%2C%20according%20to%20Gallup>, June 2018.
- [4] Harvard Business School. How Much Will Remote Work Continue After the Pandemic, <https://hbswk.hbs.edu/item/how-much-will-remote-work-continue-after-the-pandemic>, August 2020.
- [5] Harvard Business School. How to Keep your Team Motivated, <https://hbr.org/2020/04/how-to-keep-your-team-motivated-remotely>, August 2020.

Appendix 1 – Testing Case

Install the application and perform the following.

1. Login to the application using the example account.
 - Company Code – testdb
 - Username – EwanD
 - Password – p
2. When logged into the application create a new task to complete. Add the following information:
 - The tasks name.
 - The start date.
 - The submission date.
 - Any additional details.
 - The importance level.
3. Check the new task exists on the calendar. Find it by navigating to the date that was set for submission. (Can select the task to review the information on it)
4. Try sending messages to user: AlaskaS
 - Click the view messages button to open the chat view for that user.
 - Then type messages in the text box and send using the submit button.
5. Check the productivity chart stating the number of messages sent.
 - ❓ Select the messages chart by clicking the “Messages” radio button.
 - ❓ Select the time period, choose any.
6. Create a notification to get help from anybody who is available. Add the following information:
 - ❓ The recipient of the notification.
 - ❓ The name of the notification.
 - ❓ The content for the notification.
7. Upload a file to the drop box file system using the feature in the menu.
8. Delete a file from drop box.
 - ❓ Select the file using the radio button.
 - ❓ Hit the delete button.

Thank you for using the first full version of this remote working tool. Please answer the following questions by providing a score on a scale from 1- 5: with (1) being poor and (5) being great.

1. How easy was it to login to the application?
2. How easy was it attempting to add a new task to the list?
3. How easy was the calendar to use (navigating through the dates)?
4. How easy was it to send a message?
5. How easy was it to review the number of messages sent in one of the productivity charts?
6. How easy was it to create a notification?
7. How easy is it to upload a new file to the file system?
8. How easy was it to remove a document form the drop box file system?