

Software Requirements Specification for EasyWorkload

Final Proposal

Prepared by

Ray Rafael Abenido

Geoffrey Co

Teodoro Jose C. Cruz IV

Marc Gerald Simeon

Ansley Paul Sze

WindowsIsSuperiorToMachintosh

February XX, 2023

TABLE OF CONTENTS

Revision History

1. Introduction	1
1.1 Purpose	1
1.2 Definitions	1
1.3 Overview	1
1.4 Scope	1
1.5 Reference Material	1
2. Overall Description	2
2.1 Product Perspective	2
2.2 Product Functions	2
2.3 User Classes and Characteristics	3
2.4 Operating Environment	3
2.5 Design and Implementation Constraints	3
2.6 User Documentation	3
3. External Interface Requirements	4
3.1 User Interfaces	4
3.2 Hardware Interfaces	5
3.3 Software Interfaces	6
4. System Features	6
4.1 User-Managed Task Manager	6
4.2 Progress Tracker for Tasks	7
4.3 Daily Reminder and Notifications	8
4.4 Provide a Calendar or Dashboard for Bird's eye view	8
4.5 Daily To-Do List Generator	9
5. Other Requirements	10
Appendix C: Team Organization and Roles	10

Revision History

Name	Date	Reason For Changes	Version
Abenido, Co, Cruz IV, Simeon, Sze	February 02, 2023	Initial document	Proposal

1. Introduction

1.1 Purpose

This document covers all relevant details about the purpose, goals, and proposed features of the application *EasyWorkload*. No actual development has taken place, but development will start shortly after the release of this document.

1.2 Definitions

The following are some of the acronyms commonly used in this document:

1. Workload - refers to all the things a student must do.
2. Task - a generic umbrella term for some requirement a student must accomplish. Differentiated further to other more specific terms such as 'Exam', 'Homework', and so on.
3. Dashboard - home page of the app that gives the user access to the app's functionality.
4. Progress - refers to how much work has been done for a specific task.

1.3 Intended Audience and Reading Suggestions

This document is for software developers who intend to work on *EasyWorkload* or even fork *EasyWorkload* and develop separate applications.

This document contains five sections. The first section provides an idea of the product scope of *EasyWorkload* and the context of the document is written. The second section 'Overall Description' provides a bird's eye view of the application. The third section 'External Interface Requirements' describes the GUI and its interactions with external hardware and software. The fourth section 'System Features' describes the features *EasyWorkload* will have. The fifth section 'Other Requirements' contains the appendices and miscellaneous information.

Developers and testers are encouraged to read each part thoroughly, especially the fourth section.

1.4 Product Scope

At its core, *EasyWorkload* is a desktop application designed to assist busy students in better organizing their workload, helping improve time management, and accomplishing all their tasks. It performs this by integrating a task-management system, a calendar system, and regular notifications

function. This application aims to benefit students by making workload organization easier, more streamlined, less time-consuming, and less stressful.

1.5 Reference Materials

The GitHub repository contains a file named 'git_conventions' that provides instructions on utilizing Git's features and writing commit messages for this project. This file aims to standardize commit messages and maximize Git's features to benefit the project. **All developers are required to read and comply with this file.**

The repository also hosts a file named 'coding_conventions' to instruct programmers on the programming style that will be used. Just as with git_conventions, **all developers are required to read and comply with this file.**

2. Overall Description

2.1 Product Perspective

The ability to successfully organize one's workload is critical to succeeding in school. And oftentimes students are faced with the daunting need to organize tasks, assignments, exams, extra-curricular activities, and deadlines. The sheer volume of the workload can be somewhat pressuring and maybe even crushing under difficult circumstances. However, good organization can help distill a large workload into manageable and achievable chunks.

Unfortunately, it is somewhat common that students tend to poorly organize their workload for many reasons. Some students have no system to organize their workload and usually end up cramming assignments at the last minute. Others have difficulty organizing and managing their workload and its deadlines. In some cases students do not have the time to organize their workload at all. However, regardless of the reasons, the result is the same: the lack of organization or poor organization makes a student's life much more difficult than it should be.

The application *EasyWorkload* is meant to overcome these challenges and difficulties and help students organize and manage their workload. In essence, *EasyWorkload*'s primary objective is to **make organizing workload** easier. A list of subgoals that incrementally achieve the primary goal is listed below:

1. Provide a simple and easy system for students to manage their workload and its deadlines.
2. Help students quickly and efficiently organize their workload, such that they will be able to (ideally) finish organizing in thirty to sixty minutes.
3. Distill large workloads into a simple daily or weekly to-do list to allow students to make progress to finishing the workload.
4. Track a student's progress in accomplishing their tasks
5. Provide a dashboard or calendar so students can have a bird's eye view of their workload.
6. Provide daily reminders or notifications about upcoming deadlines.
7. And more importantly, make workload organization less stressful and more streamlined.

It is hoped that through this application, students will be able to successfully manage their workload. And in the process, reduce their stress about class requirements and make their stay in school easier.

2.2 Product Functions

To achieve the subgoals listed above, the following functions will be implemented in the function:

1. Features the ability to let students input the details of a task (name, description, deadline, which class this assignment is a part of, and other misc. information).
2. Features the ability to let students label a particular task as a requirement under a class

Software Requirements Specifications for *EasyWorkload*

(e.g. all Philosophy tasks are labeled as part of a Philosophy class, and math tasks are labeled as part of a Math class).

3. Features a calendar and or dashboard showing the upcoming deadlines of tasks so students can have a 'bird's eye' view or 'at a glance' view of their workload.
4. Track and provide visual progress done by students for each task.
5. Prepare and present a daily or weekly to-do list to a student at the start of the working day. The exact start time of the working day will be specified by the student at the initial set-up of the application.
6. Provide regular notifications to the student about their to-do list and any close deadlines.
7. Features such as notifications of motivational messages when you cross off an item in the to-do list.

2.3 User Classes and Characteristics

The target audience of *EasyWorkload* are generally **students**. Students, ranging from high school to graduate students, can use this application to plan and manage their school activities. However, students who have poor management skills or have difficulty managing their workload are best positioned to benefit from the application.

2.4 Operating Environment

The computer that will run the application must be able to have Windows 8 or above as its O/S.

2.5 Design and Implementation Constraints

There are no design and implementation constraints as of proposal.

2.6 User Documentation

A tutorial will be made available to the students on start-up on how to use the application. Furthermore, a help function will also be implemented to provide further assistance to the students.

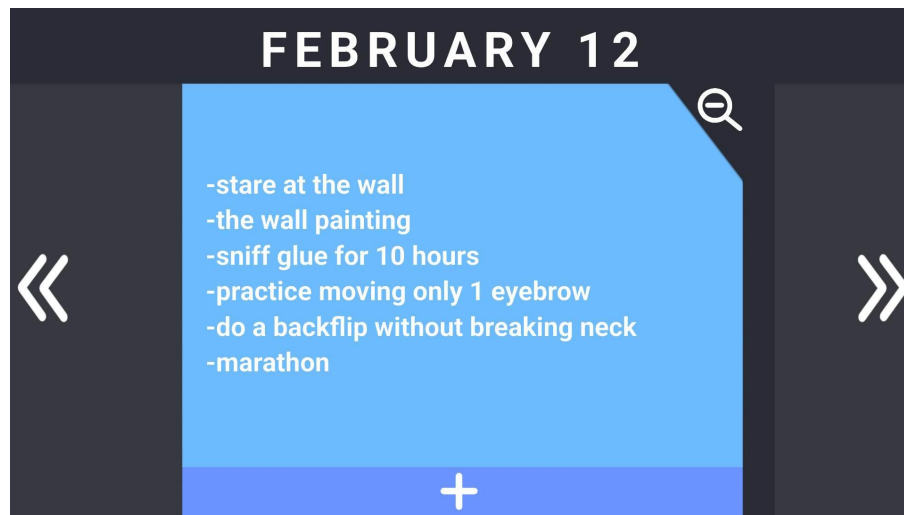
3. External Interface Requirements

3.1 User Interfaces

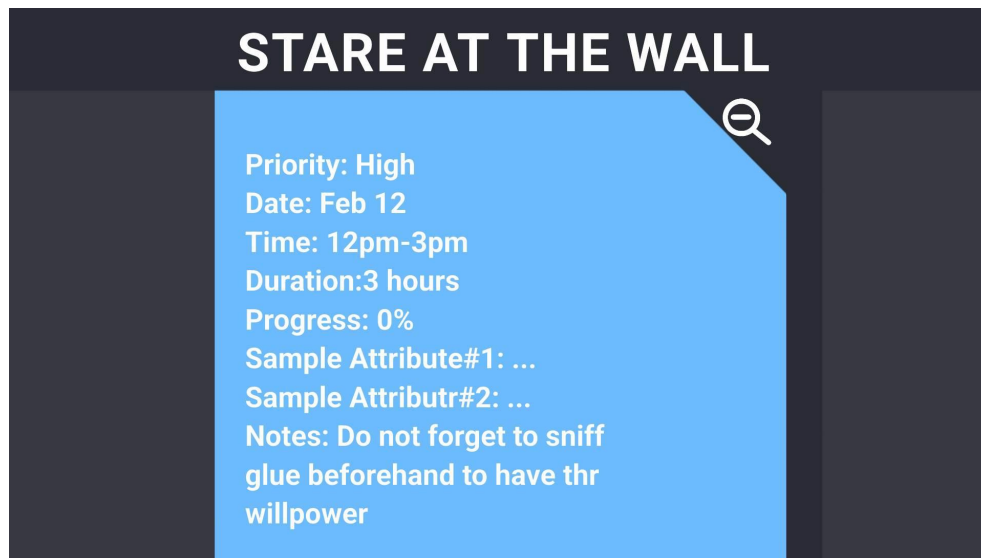
As of now, there are no specific details for the user interface as we are not sure yet of what features we want to implement are possible in the GUI. However, we have made sample screen images and styles that we will use as a reference and goal to build towards. But we know that the actual user interface we will build will not exactly be the same. Moreover, some features that are implemented in the sample screen images may be removed or edited in the actual implementation of the GUI.



Although the user interface design is still in progress, we have an idea of what we want it to look like and what we want it to do. The user interface will primarily revolve around 3 separate views. The calendar view, the day/date view, and the event view. When opening the application, the user will first see the calendar view. In this view, the user sees to see each day of the current month and things to do for those days. The user will also have a choice to go to other months by clicking the arrow buttons on the left and right. The left arrow button goes to the previous month and the right button to go to the next month. Since the calendar view cannot fit most of the characters in the names of the events, the event names will not be fully displayed. A user must click on a specific day, for example, they can click Feb 12. When a date is clicked such as Feb 12 in this image, a new view will be shown, the day view.



The day view will simply show the events for the day that was clicked on. A user will also have the option to add more events in this view by clicking the plus button at the bottom of the event list. We can also go back to the calendar view by clicking on the zoom-out button at the upper right corner of the event list. The user can still press the left and right arrow button here to go to the day before and after the date that is currently being viewed. If an event is clicked in this view, the view will change to the event view. For example, if we click on the stare at the wall event. We will see this view.



In this view, we get to see the attributes of an event. These attributes are simply placeholders. We have not yet finalized what attributes we want to be displayed in the event view. We will be able to edit the attributes in this view by clicking on a specific attribute that we want to edit. To go back to the event view, simply click the minimize button at the upper right corner of the attributes list.

3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and the communication protocols to be used.>

The supported device type will be desktops with windows operating system that is of version windows 8 or any version that is more updated. To control data within the app, a user just needs a mouse or a keyboard. As all the buttons in the U.I. such as the minimize button, add button, and left or right arrow buttons only need a click to use their functions. And any string data such as attributes of an event can be edited with a keyboard after being clicked on.

3.3 Software Interfaces

SQLite will be used to store application data. SQLite is similar to the large centralized databases supported by MySQL and PostgreSQL. It can perform CRUD operations (create, read, update, delete) and accept SQL statements. However, a key difference is that SQLite is much smaller in scope, does not require expert human support, and can be easily embedded into an application. In essence, it can serve as a miniature database/storage for the application.

The SQLite database will be connected to the application via the Java Database Connectivity (JDBC) driver for SQLite. Queries will be performed using the modules provided by the java.sql library found in native Java.

4. System Features

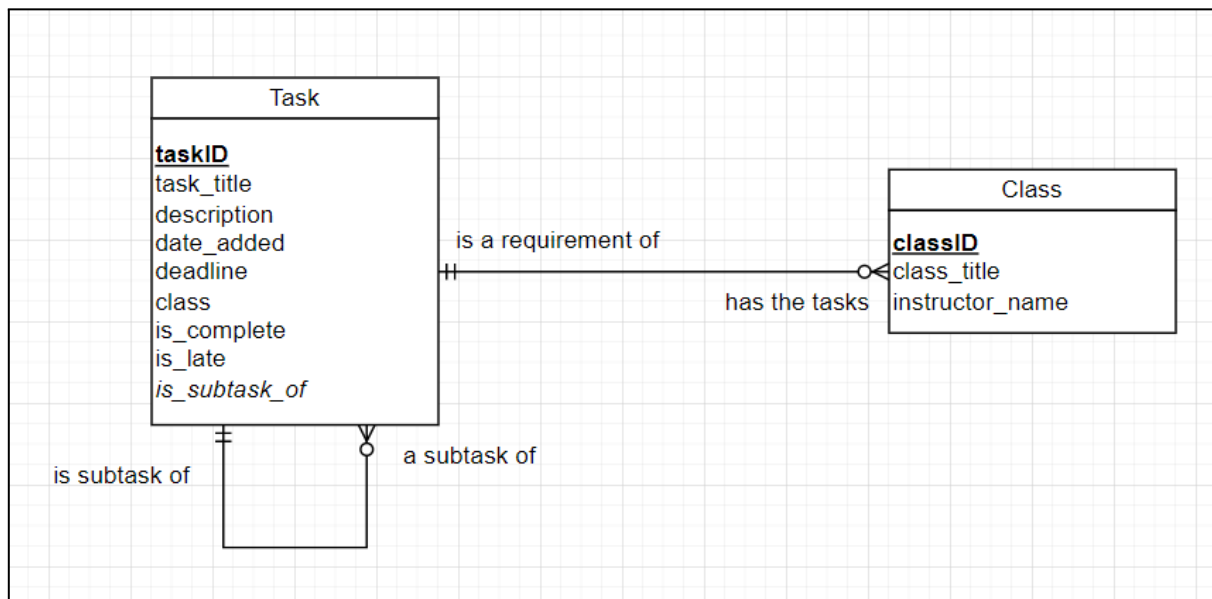
4.1 SQLite API

4.1.1 Description and Priority

SQLite API is an interface that enables the application to communicate with the integrated SQLite database. The goal of this API is to abstract all database interactions with SQLite and provide other modules included in the application an easy means to query for data. More specifically, it provides the following subfeatures:

1. Enables the application (or more specifically other modules) to execute SQL queries on the database.
2. Enables create, read, update, and delete operations (CRUD) on any row(s) on any table(s) on the database.
3. Provide abstracted data storage, writing, and retrieval operations to the rest of the application.

It is important to note that the API *will not allow for new tables to be created*. This API will only allow for the data on pre-existing tables to be changed. All necessary tables and attributes needed will be created at initial set-up of the application. The following ERD represents the tables, their relationships with other tables, and the attributes that will be created at initial-set up. Note that these are the only tables needed for the application to work with:



This is not strictly a feature in the sense it offers the user some functionality that they can make use of, but it remains critical to the functioning of the software as a whole because this handles the data storage and retrieval functions of the application. Without data storage and retrieval functions the application would not be able to hold any data for the long-term. Therefore, this feature has the **highest priority**.

4.1.2 Stimulus/Response Sequences

Because this feature is in the back-end and therefore has no interactions with the user, there are no stimulus-response sequences. Instead, this API is implemented by the various modules who needs access to user data. The exact details of implementation vary from module to module.

4.1.3 Functional Requirements

REQ4.1-1: API must be able to insert a new row of information on a table. This information and on what table it will be added will be specified on the parameter.

REQ4.1-2: API must be able to modify a value on a specified row and column on a table. The row, column, and table it will modify will be specified on the parameter. Details on how this should be implemented will follow.

REQ4.1-3: API must be able to delete a row on a table given some information passed into the parameter. Details on how this should be implemented will follow.

REQ4.1-4: API must be able to update a row on a table with new values given some information passed into the parameter. Details on how this should be implemented will follow.

REQ4.1-5: In the event of a failed attempt to perform the above operations, the API must 'catch' the error and display the SQL query in the console. Other relevant and useful information that can help trace the error can also be displayed.

4.2 User-Managed Task System

4.2.1 Description and Priority

Although the feature and exact details have been established, It is worth mentioning that the implementation can change in the future because there are many ways in which this can be implemented. This feature might even be potentially merged with other features in the future.

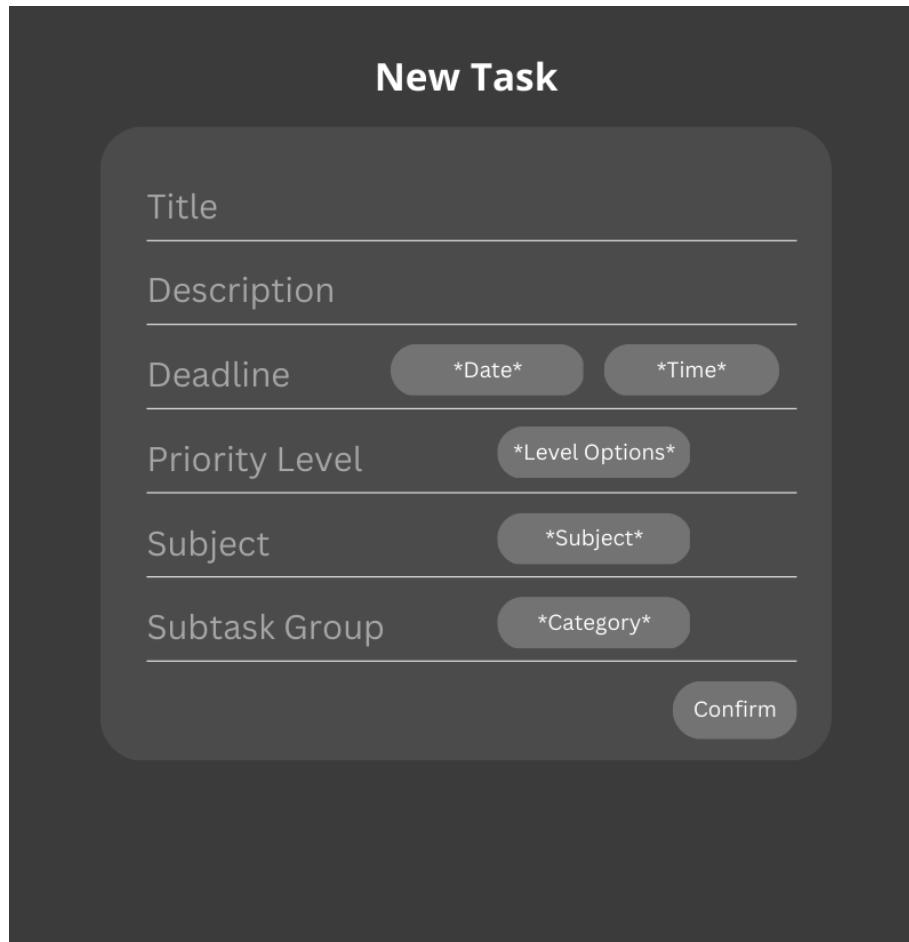
This feature allows the user to insert, delete, and modify Tasks in the application. Task here is defined as a general activity that the user must perform to meet their class requirements. This general activity may be an assignment, homework, exam, a study session, etc. Each Task contains the name of the Task, a description, deadline, priority level, and the class/subject a Task is associated with. The Task added may be standalone or a subtask to another existingTask (e.g. The main Task to 'Finish thesis paper' may have 'Finish review of related literature section' as its subtask).

This feature also allows users to create a Label to categorize their Tasks under different subjects/classes. For instance, a user creates a Label named "Philosophy". All Philosophy Tasks then are grouped under this Label instance.

This feature is critical because it enables the user to interact with the application and is the only feature that allows the user to insert any input into the application. Without this feature, the application will have no information to process at all. Therefore, this feature has the **highest priority**.

4.2.2 Stimulus/Response Sequences

When adding a Task, a user presses a button and a window pops up with fields for name, description, deadline, priority level (low, medium and high), and the class a Task is associated with. An additional field will be added to let the user determine if the new Task is a subtask to another Task instance. User fill each of the fields and press a 'Confirm' button on the bottom right of the screen. Upon pressing the 'Confirm' button a new Task instance is created and stored in the computer.

A screenshot of a 'New Task' form. The form is a light gray rounded rectangle centered on a dark gray background. It contains several input fields: 'Title', 'Description', 'Deadline' (with sub-fields '*Date*' and '*Time*'), 'Priority Level' (with a '*Level Options*' dropdown), 'Subject' (with a '*Subject*' dropdown), and 'Subtask Group' (with a '*Category*' dropdown). A 'Confirm' button is located at the bottom right of the form.

New Task

Title

Description

Deadline *Date* *Time*

Priority Level *Level Options*

Subject *Subject*

Subtask Group *Category*

Confirm

When deleting a Task, a user selects the Task to be deleted and a window pops up with all the details of that task. At the bottom right of the window, a trashcan icon will be displayed. User clicks on the icon then a prompt will appear confirming the deletion with the buttons 'Yes' and 'No'. Upon clicking 'Yes', the Task will be deleted from the computer. Upon clicking 'No', the prompt will be closed.

Delete Task

Deliverable 3

do and submit deliverable 3

Deadline

February 9, 2023

11:59 PM

Priority Level


High

Subject

CSCI 42

Subtask Group

Deliverables



When modifying a task, a user selects the Task to be modified and a window pops up with all the details of that task. The fields indicated earlier are editable and the user can change them as he/she pleases. Once the user is done changing the values of the fields, the user presses a 'Confirm' button on the bottom right of the screen. Upon pressing the 'Confirm' button a new Task instance is modified and stored in the computer

The screenshot shows a 'Modify Task' dialog box with a dark background. The dialog contains several input fields and buttons. The fields are labeled 'Deliverable 4', 'do and submit deliverable 4', 'Deadline', 'Priority Level', 'Subject', and 'Subtask Group'. The 'Deadline' field has two sub-fields showing 'February 16, 2023' and '11:59 PM'. The 'Priority Level' field shows 'High'. The 'Subject' field shows 'CSCI 42'. The 'Subtask Group' field shows 'Deliverables'. A 'Confirm' button is located at the bottom right of the dialog.

When adding a new Label or modifying a preexisting one, a field appears and the user is prompted to enter the name of the new Label. A field for instructor name may also be shown, but the user is not expected to fill this field out. Once the user is done making changes, the user presses a 'Confirm' button.

When deleting a new label, the user clicks a trash can icon. A prompt will appear confirming the deletion with the buttons 'Yes' and 'No'. Upon clicking 'Yes', the label will be deleted from the computer.

4.2.3 Functional Requirements

Be aware that functional requirements are still incomplete. More requirements may be added, removed or modified during development.

REQ4.2-1: The feature must enable to accept input from the user when creating new Task or Label instances.

REQ4.2-2: The feature must check user input and if invalid reject the input and inform the user which inputs provided were invalid.

REQ4.2-3: The feature must pass the details provided by the user to the database via the SQLite API.

REQ4.2-4: When an error occurs when trying to perform the functional requirements specified above, an error message is displayed to the user. The stack-trace and any other relevant and useful information is to be printed into the console for debugging.

4.3 Progress Tracker for Tasks

4.2.1 Description and Priority

This feature allows the user to track the amount of progress and work done for a particular task. It will be displayed as a bar to show the percentage of progress and work left. It can also track how much work is to be done for all tasks within a particular time frame.

Because this feature is not critical, it is considered a **low priority**.

4.2.2 Stimulus/Response Sequences

To utilize this feature, the user will need to set up a specific task with a deadline.

4.2.3 Functional Requirements

REQ4.3-1: The user will need to click a specific task

REQ4.3-2: Once they have accessed the task, they will need to adjust the progress of it as they please. This helps in starting where you left off

REQ4.3-3: The user may opt not to use this feature and just set the task to finish if their tasks are more simple and fast to do.

4.4 Daily Reminder and Notifications

4.3.1 Description and Priority

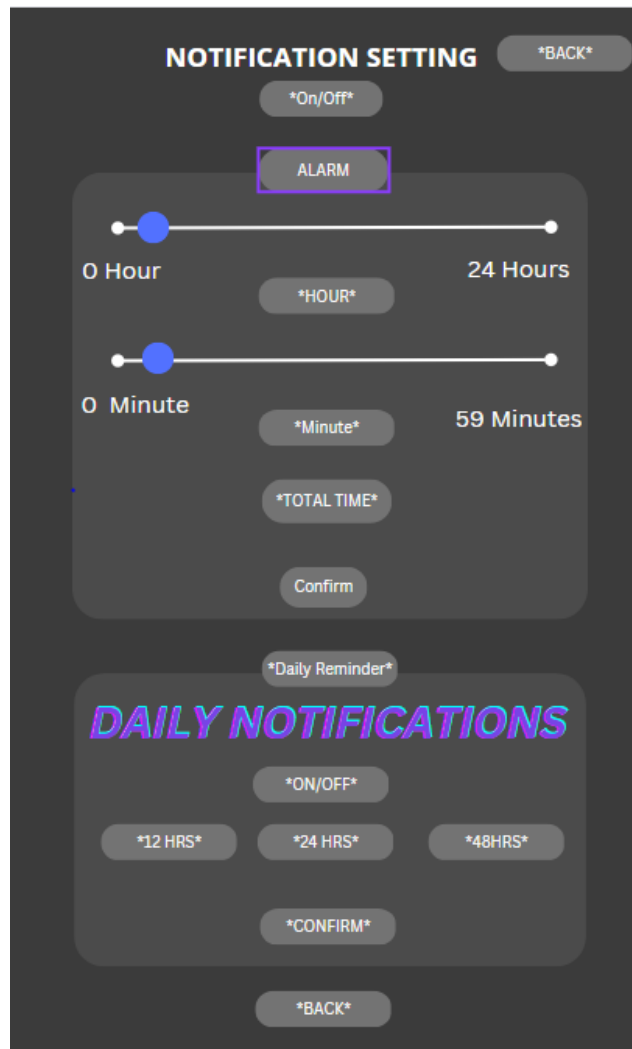
This feature allows the user to be reminded of their tasks that are about to start or tasks that have been forgotten or nearing their deadline. It will notify the user via a soft bump that will appear on their screen if they are using the device. The user could also be notified via an alarm. If the user so chooses, they can enable this feature or disable it. When working on a task, a user could leave it for a break and be reminded of that task in case they forgot. This function will run on the system clock.

To find this feature, there would be a button on each task to open the feature. Once there, the GUI will display things you can toggle for reminders. A person could toggle the reminder for a task from an hour to 24 hours and another bar for a minute to 59 minutes.

For daily notifications, the system will notify if a task is unfinished or left behind every day as needed. If the user wants to not have reminders or notifications, they could simply just turn it off in the settings. Another feature would be that when an item or task is finished, It would display a random motivational message and the user can go back to finishing the next task.

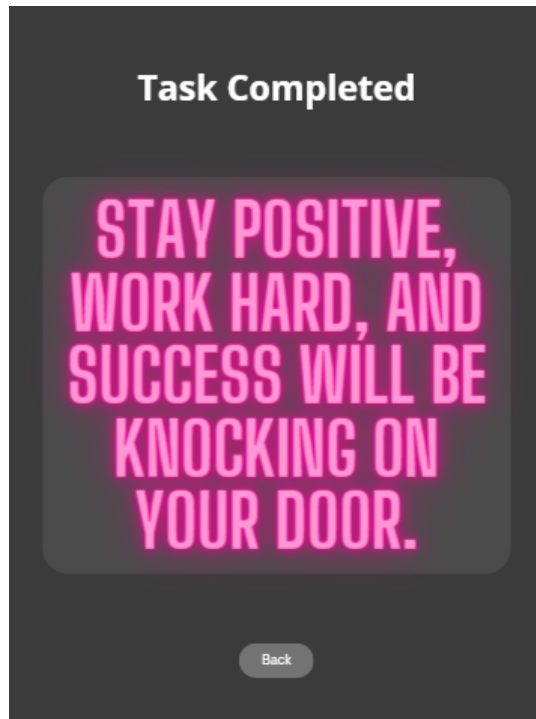
Because this feature is not critical but still integral to the software, it is considered a **medium priority**.

4.3.2 Stimulus/Response Sequences



This is a possible GUI for the notifications. This is not totally final. The first button would be the back button on the top right which leads back to the specific task page. The next button is the switch of the notifications, by default, this is turned off. A user would just press it and all the other buttons should be clickable and the screen brightens up a bit. The first panel is basically the alarm you want to set for the task. The first slider would be the hours and the second slider set is the minutes. The total time will be displayed near the confirm button. Once the user hits confirm, The alarm is set. It can be overwritten anytime. The alarm will play the sound and screen bump when the timer runs out.

The next set of buttons will be the daily notifications, this is where the task will notify the user daily or something similar. One can just turn it on and off. If the alarm and notifications overlap then only one notification and sound will play. Once turned on, the user can set it either every day or every two days. After all of this, the user can hit confirm and it will be set right away. The last button is simply another back button where the user can go back to a task



When a task is completed, a sample motivational message would appear. It would be similar to this. The user would click the 'back' button to go back to the tasks

4.3.3 Functional Requirements

Be aware that functional requirements are still incomplete. More requirements may be added, removed, or modified during development.

REQ4.3-1: In order to enable the notifications, the user must select a task from the tasks page.

REQ4.3-2: The user will then have to click on a button to open up the notification settings.

REQ4.3-3: The user should be able to see the settings and they can adjust the alarm in so they see fit. They should hit confirm to confirm the alarm

REQ4.3-4: When an alarm or notification is triggered then the system should allow the program to play a sound and show a small bump

REQ4.3-5: When finishing a task, a motivational message would be displayed.

4.5 Provide a Calendar or Dashboard for Bird's eye view

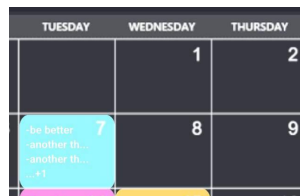
4.5.1 Description and Priority

This feature allows the user to have a calendar and or dashboard showing the upcoming deadlines of tasks so students can have a 'bird's eye' view or 'at a glance' view of their workload. This would give a student an image of what their possible schedule would be. They could select a day and then the tasks of the said day would be shown. First the Calendar will load this image.

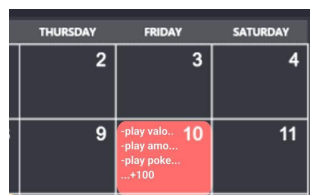
Software Requirements Specifications for *EasyWorkload*



Afterward, the current month string will be loaded at the top of the calendar. The current month can be found out by looking at the current date set in the computer in which the app is opened. After the current month has been imported, the app will create a u.i for that month. The data that will be required here are the day that the month starts and how many days there are for that month. Through this info, the app will be able to load the correct number for the day. I.e february's first day during 2023 is on a wednesday so the number 1 will be loaded on the upper right corner of that day in the GUI. This number will also serve as a button to navigate to the User-Managed Task System in 4.2.



After the number of the date is loaded, a to-do list at the left side of the date box if there are tasks in the database for that day. With this, a box graphic will be instantiated for that day behind the task list with the color that the user can set in the day view. Such as February 10 in this mock GUI image. The events will also have to be cut if they exceed 9-11 characters. And each date can only display a maximum of 3 tasks as the date box is not big enough to display more than that. The other tasks for a day exceeding the 3 task display limit will just be displayed as a number. I.e if there are 6 tasks, the 3 first tasks will be displayed and the other 3 will only be shown as "...+3".



This feature is of **High priority**.

4.5.2 Stimulus/Response Sequences

To utilize this feature, the user will need to simply log on. The calendar should be the first thing the user should see. The calendar should display an overview of the tasks on each day and their progress.

There are only 2 things the user can click on in this calendar. The date boxes and the arrow buttons. The date boxes such as the Box of Feb 10 when clicked on will navigate to a day view that is the User-Managed Task Systems of that date, showing all the tasks of that day. The user will also have the option to navigate to other months through the left and right arrow buttons. These buttons will simply change the information of the current month into either the month before or after the current month. The first day, number of days, current month string, task lists of each date, box graphic of each date and etcetera will all be reloaded to reflect the information of the month the user wants to navigate to. For example, if the user presses the left arrow button, the month of february will become january with all of january's data.



4.4.3 Functional Requirement

REQ4.4-1: The user will need to access the app.

REQ4.4-2: They can choose what day on the calendar to add tasks

REQ4.4-3: The user should be able to see the overview of a day's tasks and goals.

4.6 Daily To-Do List Generator

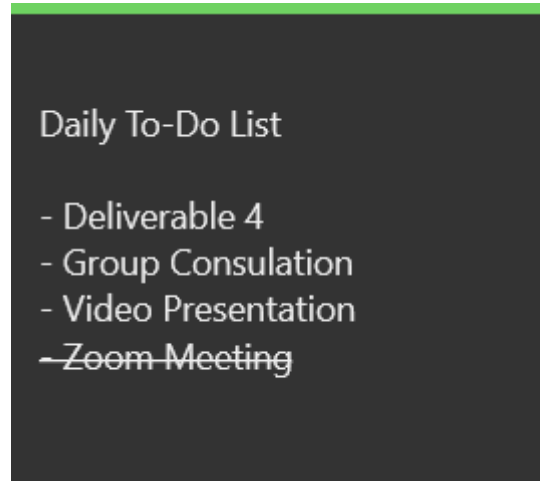
4.5.1 Description and Priority

This feature automatically distills large workloads into a simple daily to-do list to allow students to make progress in finishing their Tasks. This to-do list will be presented at the start of the working day. The exact start time of the working day will be specified by the student at the initial set-up of the application.

Because this is not integral to the application, this is considered **low-priority**. However, it is worthwhile to note that is a quality of life improvement for users.

4.5.2 Stimulus/Response Sequences

No stimulus/response sequence is needed as the daily to-do list generator is automatic. However, at first set-up of the application, the user will be prompted to set 'Working Day Start'. This will be the time the Daily To-Do List will be presented to the user. The feature will take Tasks that have been made by the user and list it based on priority and deadline. The user can cross out a Task in the list to indicate its completion or add another Task that they have already made to the list if they choose to.



4.5.3 Functional Requirement

REQ4.5-1: The feature must be able to display a list of Tasks, along with their information

REQ4.5-2: This feature must be able to process Tasks made by the user

REQ4.5-3: The feature should be able to list tasks based on priority level and deadline

4.7 Settings JSON

4.7.1 Description and Priority

This is less of a feature more of a storage for any settings about the application. This includes: Meta-information include:

1. Start of the working day
2. Maximum number of Tasks the auto-generated daily to-do list will have
3. Enable or disable Daily To-Do List Generator feature
4. Enable or disable Daily Reminder and Notifications feature
5. Other useful information.

Considering that this is not critical or integral to the application, this is considered the **lowest priority** among all the features in the application.

4.7.2 Functional Requirements

REQ-4.7-1: This feature must be able to store meta-information. Details of implementation will follow.

REQ-4.7-2: This feature must make its stored meta-information available to other parts of the application if it is needed. Details of implementation will follow

Appendices

Appendix A: Team Organization and Roles

The team will be using Facebook Messenger to coordinate. The team will be following the democratic structure in working on the project. The roles of the team members are as follows:

1. Ray Rafael Abenido - Main developer for feature 4.1
2. Geoffrey Co - Main developer for feature 4.2
3. Teodoro Jose C. Cruz IV - Main developer for feature 4.5
4. Marc Gerald Simeon - Main developer for feature 4.4
5. Ansley Paul Sze - Main developer for feature 4.3