EverydayTasks

Group 13:

Richie Yoseph Wijaya - 1906355812

Rodriguez Breil Soenoto - 1906355592

COMPUTER ENGINEERING

FACULTY OF ENGINEERING

UNIVERSITAS INDONESIA

TABLE OF CONTENTS

CHAPTER I
INTRODUCTION

1.1 Background

The Covid-19 pandemic has made the Internet even more essential for work and education. As a result, various activities may be done at the same time, and it can be hard to keep track of everything that needs to be done for one day. This is especially apparent when dealing with multiple calendars that need to be tracked.

Outside of work and education, productivity may decrease due to outdoor restrictions. The decreased productivity is caused by the limited activities a person may do in a day, outside of assignments or work. As a result, one's life may be disorganized, negatively impacting one's motivation and thus output.

In order to solve disorganization, one could maintain a short list of what was done and what to be done for a certain day, or for a certain week. However, since most people's time during the pandemic is spent in front of a digital screen, it would then be more practical to implement a solution that uses it. People may use productivity apps to help organize daily tasks.

Most productivity apps, including to-do apps and calendars, only give statistics on how many activities left are to be done soon. While they provide an easy way for users to track them, it isn't enough to gauge daily productivity. Journals or diaries may help, but it will not be very practical to look at each entry and quickly determine when someone is the most productive.

1.2. Project Aim

Our project proposes an activity record format. Users may write short descriptions of what they have done after they have done a certain activity, and it will be automatically recorded. As the user adds records every day, a user's productivity may be easily assessed daily, weekly, monthly, or over other time periods.

Our project will not simply record activities, but it will also record to-dos as well as its associated activity. In contrast, most to-do applications simply let the user mark an activity as done by simply clicking on a check box. This design allows the user to keep track of tasks as well as ensure they have done them, and to distinguish between a task that was cancelled and one that is actually done.

This project should ideally run across multiple devices, and across different platforms, so that the user may handily access it anywhere. For that, an ideal solution would be to implement the project as a web application. Web applications can fulfill this requirement, as they only need a web browser to access them. Most devices currently in use today have web browsers. Creating web applications, therefore, would reduce the work needed for the project to be cross-platform.

1.3. Literature Review

Personal productivity varies in measurement, both qualitatively and quantitatively. An individual's factors in measuring productivity will differ on another's, depending on what they do or what they work in. One common way of measuring personal productivity uses a two step process - time tracking and measuring output[1]:

Time tracking is the practice of recording an individual's activities throughout the day. This may be done on an hourly basis, or after every activity is done. Because it can be used to track break times versus work times, time tracking functions as a reference point for an individual to make changes regarding his daily life.

Measuring output refers to creating a set of goals and then measuring productivity from how many goals that are completed. This may be done using a to-do list, in which the individual checks off tasks in the list that are completed. This may result in an incentive to try and complete many tasks in a day.

1.4. Similar Competitors

1.4.1 Trello

Trello is a web application designed to organize Kanban-style lists. It is used for many projects whose completion is divided into many tasks.

1.4.2 Google Calendar

Google Calendar is a web app that lets the user keep track of events. It is integrated with Gmail, as well as having a mobile app that runs on Android and iOS. Users' events can be categorized into several groups, with the ability to toggle each groups' visibility.

1.4.3 Kanboard

Kanboard is similar to Trello in that it lets users manage Kanban lists. However, Kanboard is free and open source, and can be easily deployed locally.

1.5. Functions/Module

1.5.1 Activities

Everything the user does will be recorded as a unit of activity. Each activity is recorded with the date and time that the user has finished an activity, and has a short title or description.

1.5.2 Task List

A list of activities to be done. They may or may not have time and date deadlines. Unlike traditional to-do apps, an entry's completion must have an activity associated with it. Therefore, in order to mark an entry as done, the user must record an associated activity.

1.5.3 Projects

A set of tasks that are necessary to complete a certain goal. This may or may not have deadlines. A project can be marked as complete only if the tasks contained within are completed.

1.5.4 Dashboard

If implemented, this will show a summary of the user's completed activities, tasks to be done, current projects, or anything else the user needs to gauge their activity. The view can vary, for example there may be a graph showing daily completed activities for the week.

1.6. Tools

1.6.1 PHP

PHP is one of the languages that is used in programming. In our project, PHP will be used for:

1. The back-end that processes user requests
2. A templating system which will be used to prepare the final display

1.6.2 MariaDB

MariaDB is one of the tools that is used for databases. In our project, we use MariaDB because:

1. MariaDB is a database engine compatible with MySQL
2. MariaDB is a database that is more actively maintained

1.6.3 Figma

Figma is a popular tool often used for prototyping and general graphic design. It runs on the web and has collaboration features. In our project, Figma is used because:

1. Has convenience features such as frames and components, to make GUI prototyping easier
2. Able to collaborate with other team members for reviewing
3. Has a rich vector graphics editor

1.7. Resources

1.7.1 NGINX

NGINX is one of the resources that is used for servers. In our project, we use NGINX because:

1. It is easy to configure than Apache
2. It is a server that doesn't take a lot of resources

1.7.2 Ubuntu and Windows

Ubuntu and Windows are the resources that are used for the Operating System. In our project, we used Ubuntu and Windows because:

1. Ubuntu and Windows are operating systems that can be optimized for the server's needs.

2. Ubuntu is one of the operating systems that is open-sourced and can be downloaded for free.
3. Windows is one of the operating systems commonly installed and used on many computers, laptops, and notebooks.

1.7.3 Heroku

Heroku is a cloud application platform that offers hosting of web applications in various programming languages and environments. We are considering using Heroku for deployment due to its easy setup.

1.8. Risk Analysis

1.8.1 Software

- **Privacy concerns** - Because the project runs as a web app using a server, the user's daily activities are logged on the server. This may include potentially sensitive and private information. Solutions include:
    - Make the project open source and self-hostable. The downside to this approach is that most people do not have the technical understanding needed to self-host the project. However, this approach can make it possible for anyone who has the appropriate skills to run instances for their friends.
    - Increase security of the web application by fixing various vulnerabilities and reducing vectors of attack.
- **Resource usage** - As a web application, it requires more resource usage than a regular desktop application. Resource usage varies depending on the browser. It also requires an internet connection, except in cases where the user self-hosts the application.

1.8.2 Tools

Programming Language

PHP has some risk/weakness that might affect the software as listed below [2]:

1. Not sharing resources between process, leads to high resource usage, and also more difficult to use on larger system
2. Flexibility.

- Maintaining a PHP project might be hard due to differences in understanding of the code for each person
- PHP has too many external libraries that making it worse to maintain

3. Extension Dependent. Processes on PHP sometimes rely on external extension libraries that makes the connection between it and the database slower. Although there is a feature called persistent connection that boosts the connection, ensuring the performance won't be possible compared to other languages.

Database

MariaDB has some risk/weakness that might affect the software as listed below [3]:

1. Liable to bloating, leading to slower performance due to the increasing size of central IDX log file after protracted use.
2. Slow caching.

1.8.3 Resources

Server

If we misconfigure the NGINX server, it will become a risk/weakness that might affect the software as listed below [4][5]:

1. Off-by-slash misconfiguration, making it vulnerable to path traversal
2. Unsafe variable use, might lead to XSS, HttpOnly-protection bypass, information disclosure, and RCE
3. Usage of $uri or $document_uri instead of $request_uri, might lead to a CRLF injection
4. merge_slashes set to off, might result in vulnerable to local file inclusion

Operating Systems

Ubuntu and Windows have some risk/weakness that might affect the software as listed below [6][7]:

1. **Ubuntu**

- Software and hardware incompatibility. Some user might cannot use this app on Ubuntu OS because the incompatibility between their hardware and Ubuntu OS, although Windows OS might solve this weakness

- There are other Linux distributed OS that is better and use lower system requirement like Linux Mint OS and Debian OS.

2. Windows

- High resource system requirements, same problem with the Ubuntu OS
- Poor Security
- Virus Susceptibility

Cloud

Heroku have some risk/weakness that might affect the software as listed below [13]

- Poor network performance
- High inbound and outbound latency
- High chance of the dynos being inaccessible due to various reason

CHAPTER II
PROJECT MANAGEMENT

2.1 Members

2.1.1 Richie Yoseph Wijaya

In this project, Richie will responsible for:

1. Creating the report
2. Reviewing/Maintaining the report
3. Reviewing the code/program
4. UI design
5. Drafting the project agreement

2.1.2 Rodriguez Breil Soenoto

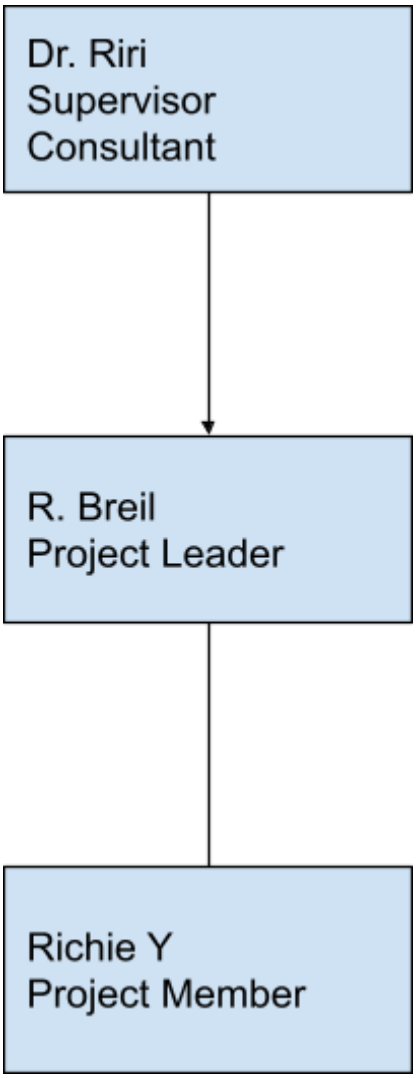
In this project, Rodriguez will responsible for:

1. Creating the report
2. Creating the code/program
3. Maintaining the code/program
4. UI design
5. Drafting the project agreement

2.2 Project Management

Structure

For the project's execution, the team structure is as follows:



CHAPTER III
PROJECT PLAN

3.1. Project Schedule

Our project's schedule from initial planning, prototyping, design and implementation is roughly as follows.

<table>
<tr><td>Month</td><td colspan="4">Oct</td></tr>
<tr><td>Week</td><td>1</td><td>2</td><td>3</td><td>4</td></tr>
<tr><td>Milestones</td><td colspan="2"></td><td>Progress Report</td><td>Alpha</td></tr>
<tr><td rowspan="3">Richie</td><td rowspan="4">Initial Planning</td><td>Implement Activity API</td><td rowspan="4">Midterms</td><td>Make Suggestions</td></tr>
<tr><td>API Testing</td><td rowspan="3">GUI Testing</td></tr>
<tr><td>Review Report</td></tr>
<tr><td rowspan="2">Breil</td><td>GUI Design</td></tr>
<tr><td>API Prototyping</td><td>Make Report</td><td>Implement Activity GUI</td></tr>
</table>

<table>
<tr><td>Month</td><td colspan="4">Nov</td></tr>
<tr><td>Week</td><td>1</td><td>2</td><td>3</td><td>4</td></tr>
<tr><td>Milestones</td><td colspan="3"></td><td>Beta</td></tr>
<tr><td rowspan="3">Richie</td><td rowspan="3">Implement Task API</td><td>Make Suggestions</td><td rowspan="3">Implements Project API</td><td>Make Suggestions</td></tr>
<tr><td rowspan="3">GUI Testing</td><td>Surveying and Beta Testing</td></tr>
<tr><td rowspan="2">GUI Testing</td></tr>
<tr><td rowspan="2">Breil</td><td>Make Suggestions</td><td>Make Suggestions</td></tr>
<tr><td>API Testing</td><td>Implement Task GUI</td><td>API Testing</td><td>Implement Project GUI</td></tr>
</table>

Month	**Dec**			
Week	**1**	**2**	**3**	**4**
Milestones		**Progress Report**		**Release Candidate**
Richie	Implement Authentication API	Review Report	GUI Testing	
	Make Report			Make Suggestions
Breil	Make Suggestions		Implement Authentication GUI	Final Adjustments and Tweaks
	API Testing		Implement Dashboard GUI	

Month	**Jan**		
Week	**1**	**2**	**3**
Milestones		**Finishing**	
Richie	Make Report	Review Report	Finals
Breil	Final Adjustments and Tweaks		

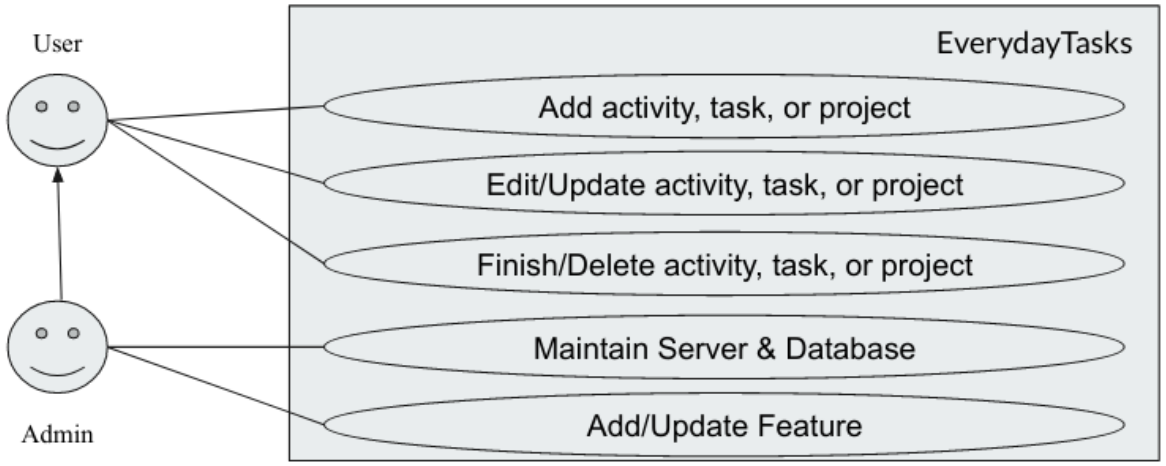
3.2 Risk Analysis

#	Risk	Likelihood	Impact	Retire cost	Priority	Mitigation
1	Suboptimal code may affect performance	**3**	**5**	**4**	**192**	Ensure each component of project design is planned in order to reduce the chance of needing to quickly patch in suboptimal code.
2	Implementation may be hindered by inadequate understanding of PHP.	**5**	**8**	**4**	**72**	Learn from tutorials and documentation, then apply to the project accordingly.
3	Web applications may have a higher chance of not being secure	**5**	**5**	**6**	**216**	Mitigate vulnerabilities by using tools such as **OWASP ZAP** (a security testing tool) to check for application errors caused by malformed input and random queries.
4	Difficulties in testing various aspects of the project	**2**	**1**	**1**	**90**	Use tools such as **Postman** (for testing HTTP API) and **Ngrok** (A web proxy to make localhost accessible) to make both API and GUI testing convenient.

3.3 Project Design

The design of our project can be illustrated as follows:

3.3.1 Use Case



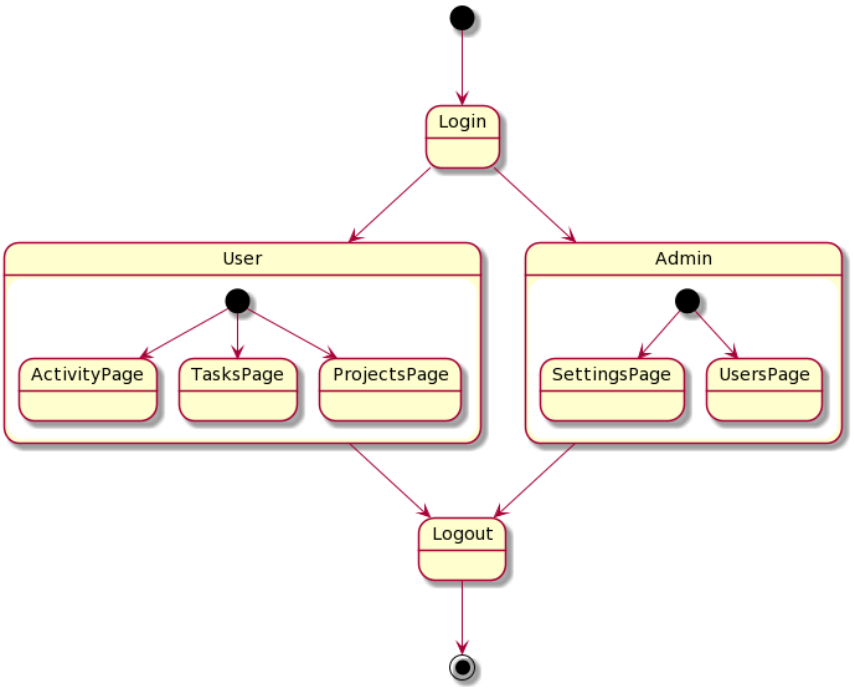
- User

  The user may add, edit, update, finish, or delete their activity, task, or project.

- Administrator

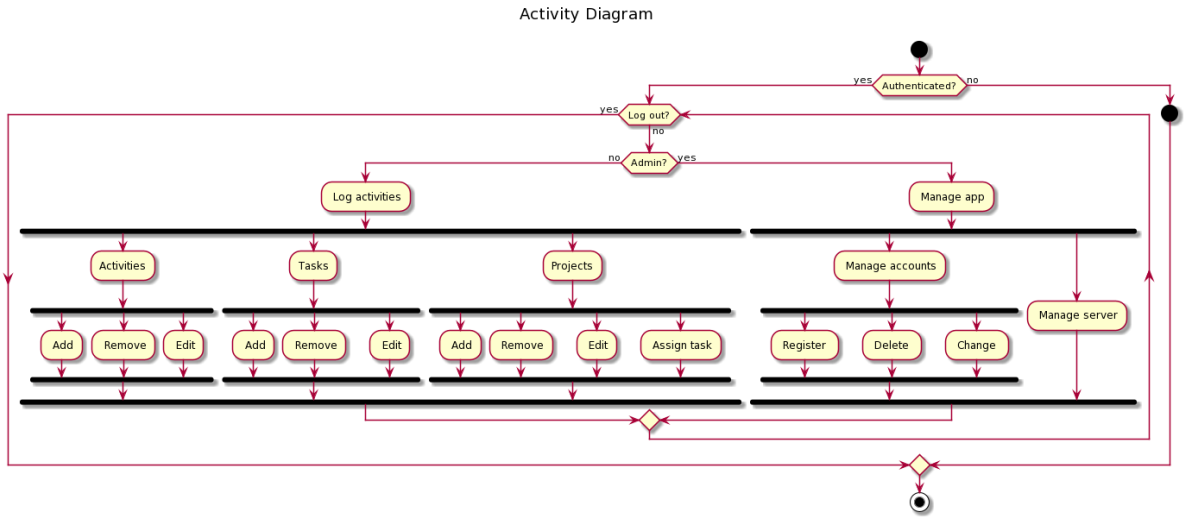
  The administrator may add features, update features, maintain the server, or maintain the database.

3.3.2 State



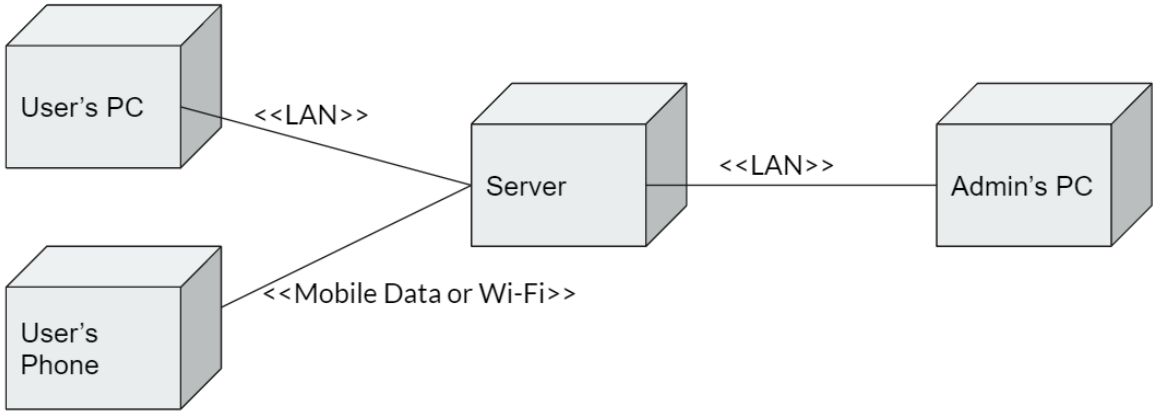
The application consists of several states that can be reached by either the user or by the administrator. They are essentially pages that can be accessed after reaching a certain state.

3.3.3 Activity



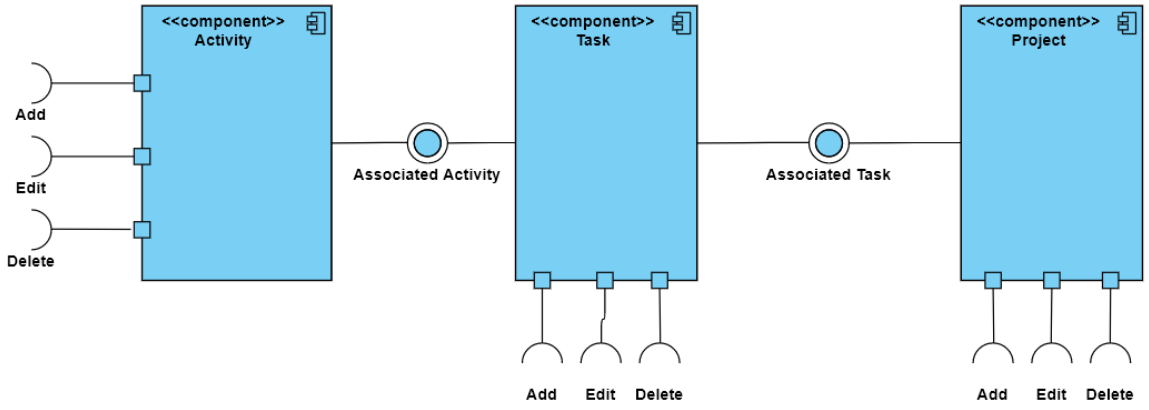
This illustrates the path in which the application can be navigated. Once logged in, the user may log his or her various activities, while the administrator can manage accounts and the server itself.

3.3.4 Deployment



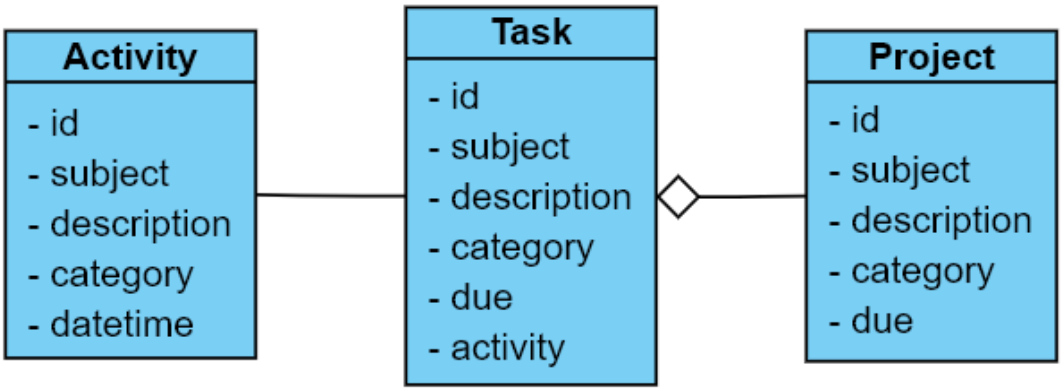
In the deployment diagram, we can see that all of the devices are connected to the server by LAN, Mobile Data, or Wi-Fi. The project may be accessed by multiple users at the same time.

3.3.5 Component



The component diagram visualizes at a glance the interfaces and the components of the application.

3.3.6 Class



The class diagram outlines more closely how the structure is to be implemented in the final product.

CHAPTER IV
IMPLEMENTATION

4.1 Overview

The application will be implemented in the form of a cloud-based web application. It will be written in PHP, and will offer both a GUI for the user-facing side, as well as an API to help with development.

4.1.1 API

The API will be a REST-like API accessible through a URL prefixed with “api/”. In this project, the objects accessed through the API are represented as JavaScript Object Notation (JSON). This allows the object to be easily parsed by most applications. Due to its conciseness, it is also preferable compared to Extensible Markup Language (XML).

Our API implements a form of Hypermedia as the Engine of Application State (HATEOAS)[12], which allows the API to be self-documenting through links provided in the object output. The advantage of this approach is that it reduces the need for hard-coded URLs, therefore making non-breaking API changes easy to perform.

Each URL can have operations done by sending the appropriate HTTP requests:

- **GET**: Fetches information about an object[8].
- **PUT**: Creates or updates an object’s information[9].
- **DELETE**: Deletes the object from the server[10].
- **POST**: Creates a new object on the server[11]. In this project, due to the fact that POST is more widely supported than PUT (for example, forms only support sending GET and POST), we implement Idempotency Tokens. They are identifiers that are used to prevent duplicate requests, such as in the case of multiple retries during a slow connection. It is used such that multiple requests in succession will only change the state of the application once, hence it is idempotent.

4.1.2 GUI

As with every other web application, the GUI will be implemented primarily in HTML. CSS is used to define the look and feel of the application, while Javascript is used to implement

various functionality that enhances the application (such as AJAX support and single page functionality). The look and feel of the application is prototyped using Figma, and will serve as a reference for full implementation.

The GUI facilitates the user or the administrator to carry out their various needs according to the Activity and State diagrams shown in Chapter III. It will be designed such that it will be easy for them to access the various functionality of the application.

4.2 Activities

In this application, **activities** are data added by the user to note that he or she has done something in a day. The information contained within include a short summary of the activity ("subject"), the time and date in which the activity was committed, an optional detailed summary of the activity ("description"), as well as an optional category in which to place the activity in.

4.2.1 API

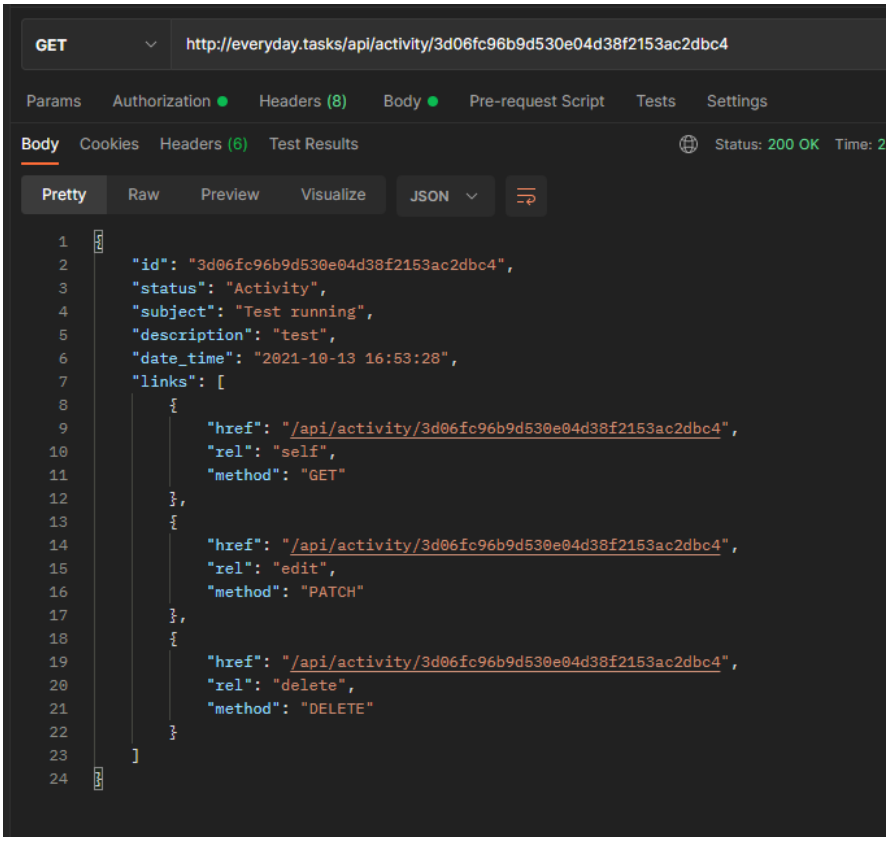
A working Activity API is implemented, and can be used to perform create, read, update and delete (CRUD) operations on a single Activity object. The Activity object itself has three forms:

- As a PHP object in the server;
- As a database table;
- As a JSON object retrieved by the user.

The API translates the object between these three forms as needed, and can be operated on by the user using the methods described in section 4.1.1. The URL for manipulating activities is:

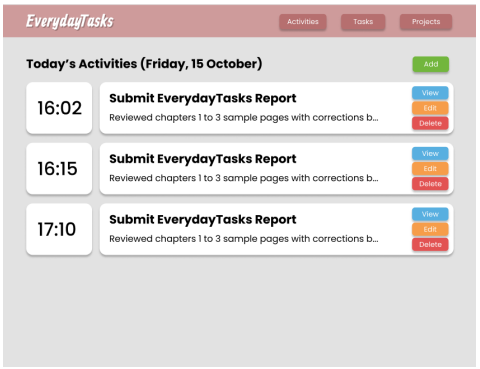
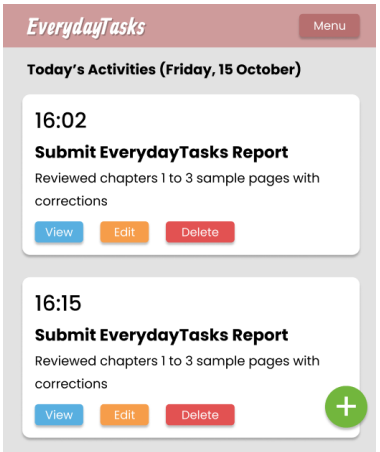
- /api/activity: For manipulating an activity or to create a new activity
- /api/activity/<id>: For manipulating a single activity (editing or deleting)

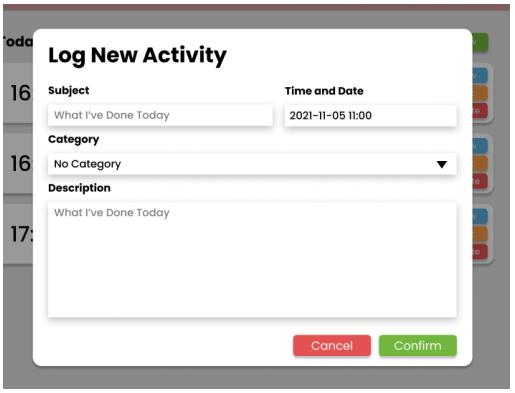
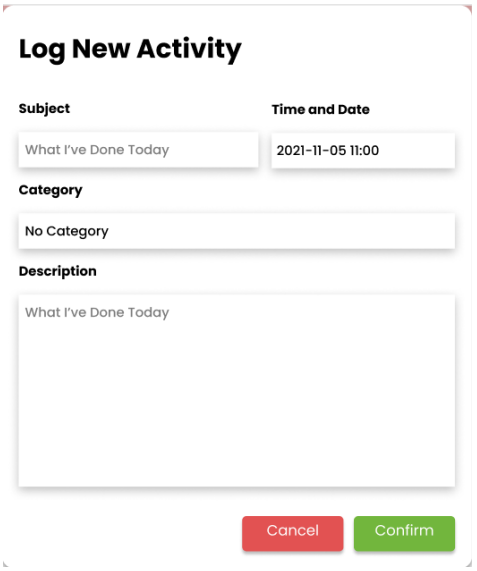
We have tested the API with the help of the Postman software, an application designed to ease testing web API's. Below is a screenshot from one of the test sessions. The custom "everyday.tasks" domain shown is not a real public domain; it redirects to the test system's *localhost*.



4.2.2 GUI

The prototype of the activities page is as shown below:

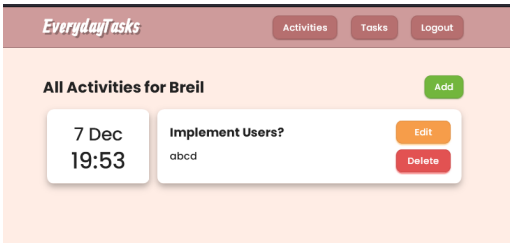
Mockup	Desktop	Mobile
Main Activity page		

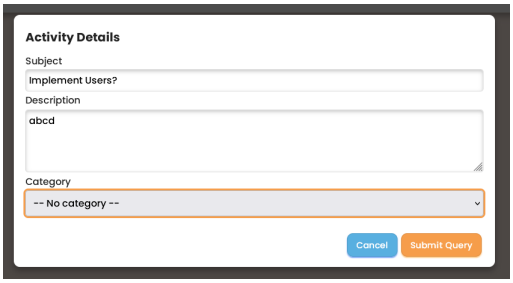
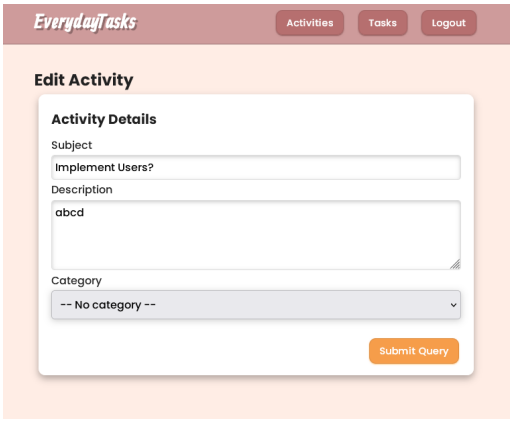
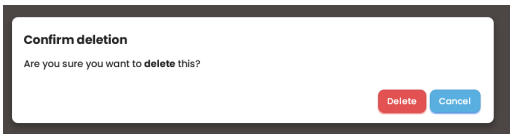
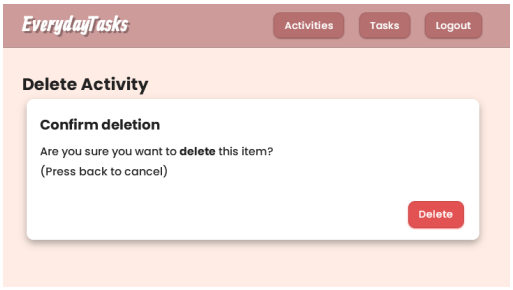
Add New Activity		

The implementation is done in a hybrid way. If the activity edit and delete URLs are loaded in isolation: "/activity/<id>/edit" or "/activity/<id>/delete", the page loaded is the static dialog written only in HTML. However if it is accessed through the main "activity" page, then it is implemented inside of the site's Javascript.

The advantage to this approach is that users unable to load Javascript will still be able to perform activity logging, editing and deleting. Meanwhile, regular users will have a smoother experience in using the website, since the dialog does not need to be loaded as a separate page. However, the disadvantage is that since we are not using a specialized library for single page applications (such as React), the Javascript and the HTML code need to be updated separately, which adds complexity to the code.

The static version of these pages are written in PHP and HTML, and any requests made will go directly to the backend. However, the Javascript version will create requests to the EverydayTasks API, and upon a successful request, the page will update itself to reflect the new state of the database.

Implement	**Desktop**	**Mobile**
Main Activity page		(To be implemented)

Add New or Edit Activity	 	(To be implemented)
Delete Activity	 	(To be implemented)

4.3 Tasks

Tasks are a list of activities that the user plans to perform at some time in the future. They will have a short summary describing the task (“subject”), and optionally include longer descriptions (“description”) as well as an optional category.

4.3.1 API

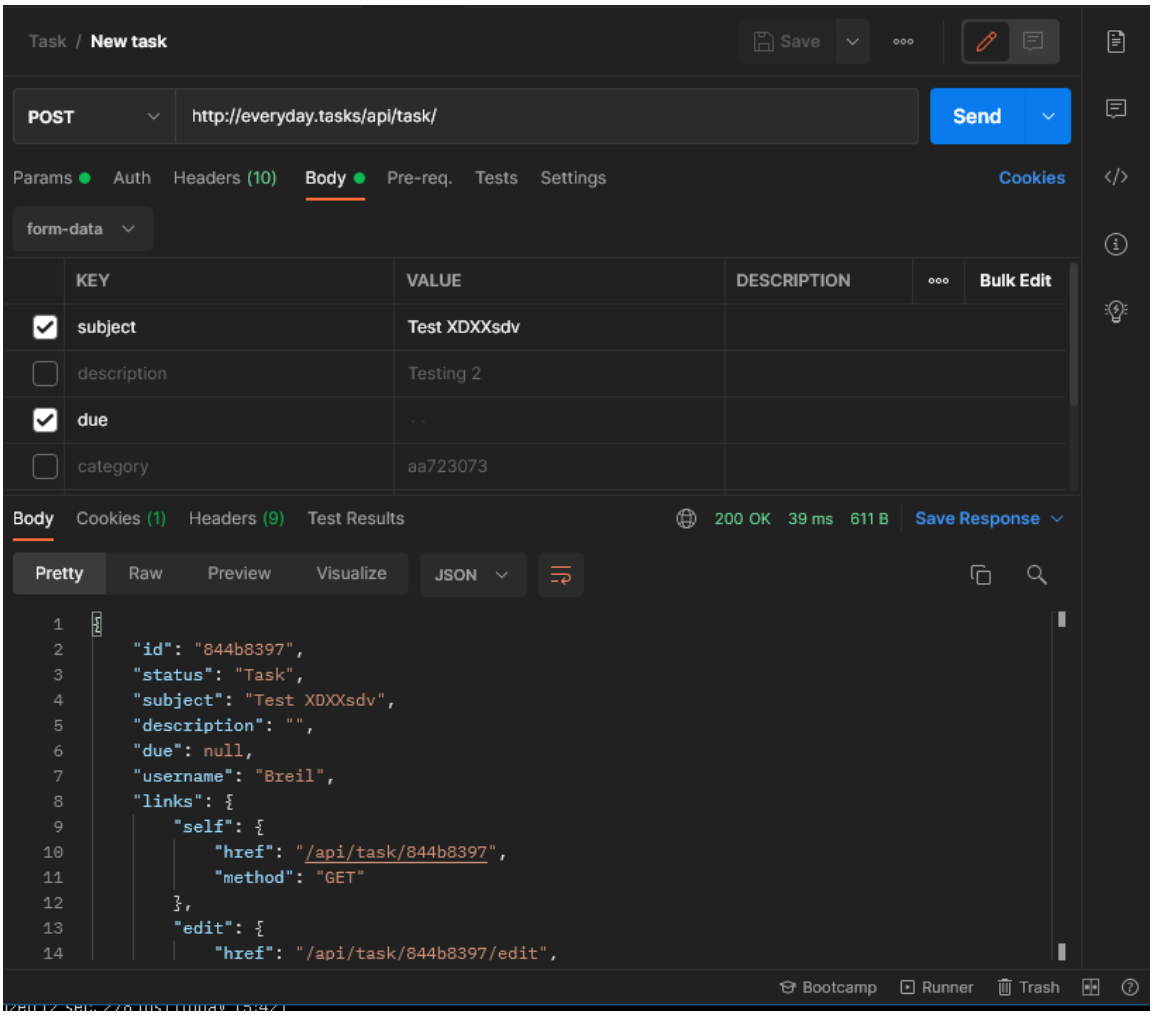
A working Tasks API is implemented, and can be used to perform create, read, update and delete (CRUD) operations on a single Task object. Like the Activity object, the Task object also has three forms:

- As a PHP object in the server;
- As a database table;
- As a JSON object retrieved by the user.

The URL for manipulating tasks are:

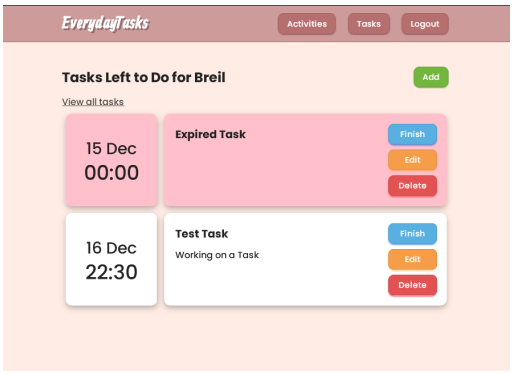
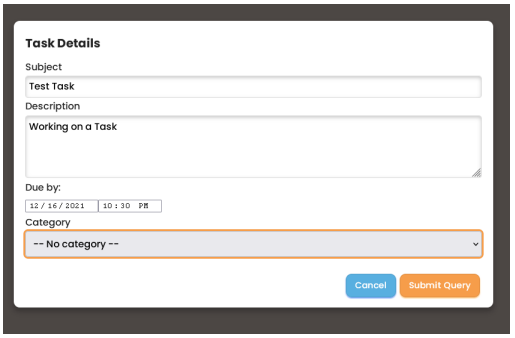
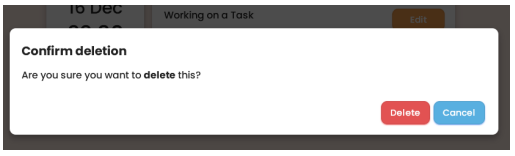
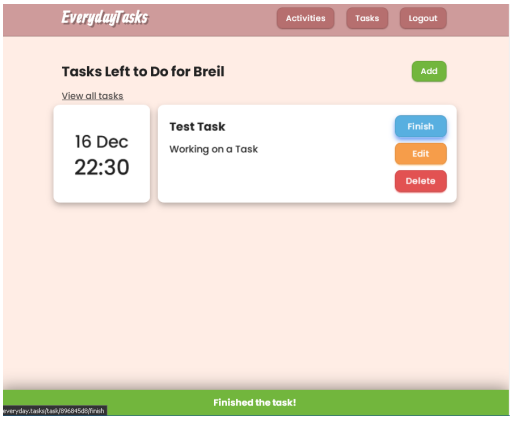
- /api/task: For manipulating a task or to create a new task
- /api/task/<id>: For manipulating a single task (editing or deleting)
- /api/task/<id>: For finishing a single task. Sending a POST request to this URL will complete the task by adding an activity, and then associating it with the task.

As before, we have tested this API with Postman. Below is a screenshot from one of the test sessions.



4.3.2 GUI

Like the activity page, the implementation is planned to be done in a hybrid way. However, we were not able to implement the hybrid page in time for integration testing and the eventual final version. The following is the implementation of the Tasks GUI:

Implement	**Desktop**	**Mobile**
Main Tasks page		(To be implemented)
Add New or Edit Task		(To be implemented)
Delete Task		(To be implemented)
Finish Task		(To be implemented)

4.4 Projects

Projects are activities the user plans to perform gradually, and is made up of a set of Tasks. In order to complete a Project, the user must finish all of the Project's Tasks. In addition to a Task list, the Projects will contain a short summary ("subject") and more detailed info ("description") of its own.

4.4.1 API

A partially working Projects API has been implemented so far, and can be used to operate on Project objects, which also comes:

- As a PHP object in the server;
- As a database table;
- As a JSON object retrieved by the user.

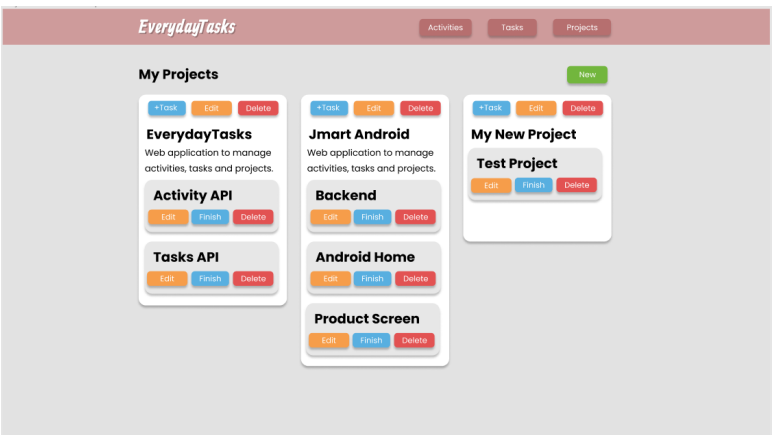
The URL for manipulating projects are:

- /api/projects : For manipulating a task or to create a new task
- /api/projects /<id>: For manipulating a single task (editing or deleting)

We were unable to test the projects API in time for integration testing and release, so it is not included at this moment.

4.4.2 GUI

The GUI is planned, however we were unable to implement it in time for testing and release. The prototype of the projects page is as shown below:

Mockup	**Desktop**
Main Projects page	

CHAPTER V
TESTING METHODS

5.1 Introduction

This part is intended to define EverydayTasks' testing methods and infrastructure. The following will be defined here: test objectives, scope, scheduling, risks and approach. We will also define deliverables (products to be tested) at what stages of testing, and what will be the scope of testing.

5.1.1 Objectives

EverydayTasks is a web application that makes it possible for users to log their daily activities using an intuitive user interface. It also makes it possible for developers to easily get their daily activity data using its API. It is written in PHP, using a MySQL or MariaDB-powered database.

The application needs to be tested to ensure proper functionality, fulfillment of requirements and that the user's needs are met. Testing will be done by the developer as well as the team members. In order to ensure EverydayTasks will be functional for the end-user, a testing methodology will have to be created. Testing will be divided into several stages, according to the project's set milestones. We will use common terminology for test releases: alpha testing, beta testing, and a release candidate.

Alpha testing will be the internal testing of the web application performed by the team members and developers. It will be used to evaluate functionality of core features. Beta version of this test will test the user experience at an early stage, and may involve a test group consisting of a few people. The final version (or release candidate) of this test will be the final testing of the app, after most features have already been implemented, and is used to prepare the web application for release.

5.1.2 Team Members

Resource Name	Role
Richie Yoseph Wijaya	Test Manager / Tester
Rodriguez Breil Soenoto	Project Leader / Tester

5.1.3 Scope

By the end of the alpha stage, the application will have all core requirements. The core requirements are that the user will be able to:

1. Log an activity
2. Edit and delete the activity
3. View activities for today and also past activities that have been logged.

By the end of the beta stage, additional requirements will have to be met, which include:

1. Create a task
2. Edit the description, time of the task
3. Delete a task
4. Finish a task by logging its associated activity
5. View all finished and unfinished tasks

Since the user base is currently small, load testing won't be considered during the initial stages. However, as the user base grows, load testing will have to be done at some point. Load testing is intended to test the scalability of the application, and can be done using benchmarking tools.

Security testing will also be considered here, including input validation and fuzzing. It will be done at the user level, the application level and the database level to ensure no obvious vulnerabilities are present.

5.2 Assumptions / Risks

5.2.1 Assumptions

This section lists assumptions that are made specific to this project.

1. Product delivery will be in a form that can be committed to the GitHub repository.
2. Core team members have the same setup, including server configuration, database configuration and local host domains.

5.2.2 Risks

#	Risk	Impact	Triggering	Mitigation Plan
1	Scope creep – as testers become more familiar with the tool, they will want more functionality.	High	Delays in implementation date	Considering each feature request extra carefully, will stick to initial requirements
2	Feature and design requirement changes may negate any test cases written up to that point	High	Loss of all test cases	Backup database before upgrade and merge it in afterwards
3	Delivery is irregular due to other assignments and tasks	Medium	Product did not get delivered on schedule	Reschedule as needed, or postpone features until the next version.

5.3 Test Approach

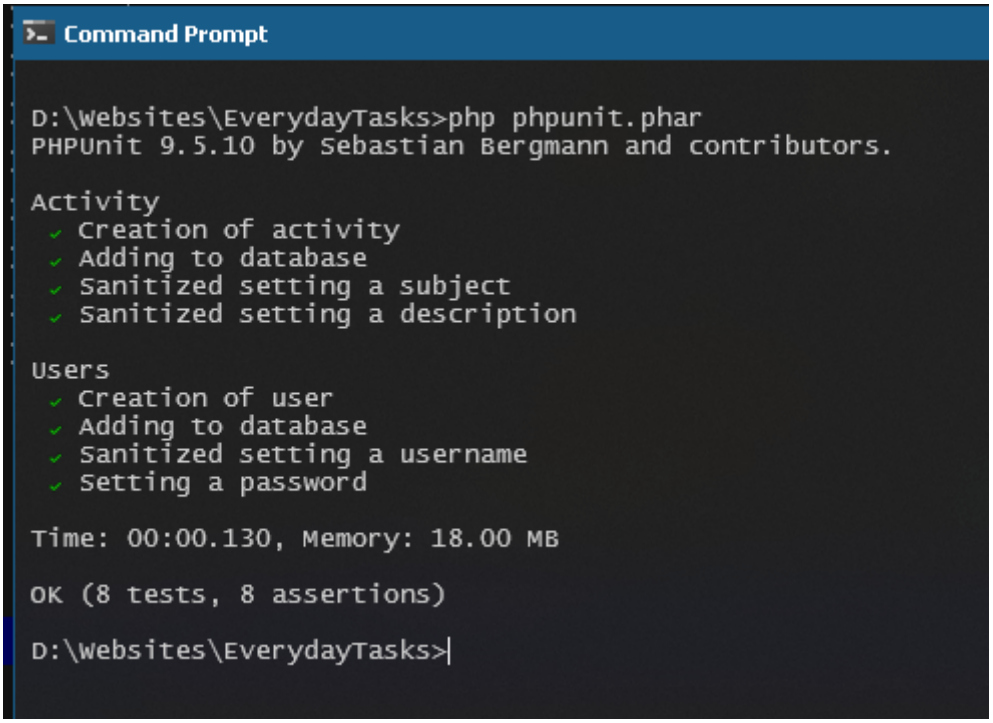
Testing will be done on both the API and the front-end. At present, testing is done manually. However test cases will be built for the various functionality in the app.

5.3.1 Test Automation

Test automation is currently employed for unit testing PHP model functions, and is currently only available to test the Activity object. We plan to create unit tests for the Tasks and Projects object, but we were not able to implement them in time. We use the PHPUnit library to create test cases for the application.

The unit test uses a separate database with a structure identical to that of the production database. Each time the unit testing session is started, this database is cleared and rebuilt to ensure a clean environment. Currently, we are only testing the Activity object and adding users into the database.

The Activity test suite includes: creating an Activity object, adding it to the database, as well as testing setting subject and description in a sanitized way (such that it is not prone to cross-site scripting / XSS attacks). For the Users test suite, it will test creating a User object and adding it into the database, as well as setting a sanitized username and a valid password. More unit tests are planned to be written as the application grows. Below is the result of the unit testing:



5.3.2 Test Environment

The test environment will use Postman for API testing, and Firefox or Chrome for front-end testing. As mentioned, unit testing will utilize PHPUnit using a separate database. The PHP version used for PHPUnit should be the same version as used in the actual application, a minimum of PHP 8.0 (which is the latest stable version at the time of writing). The database used for the test runs on the same engine as the actual application as well, using MySQL or MariaDB.

5.4 Planned Deliverables

Deliverable	For	Date / Milestone
Test Plan (Alpha)	All Team Members	13 Nov 2021
Traceability Matrix	Test Manager	13 - 14 Nov 2021
Test Results	All Team Members	14 Nov 2021
Test Status report	Project Leader	14 & 19 Nov 2021
Metrics	All team members	

Deliverable	For	Date / Milestone
Test Plan (Beta)	All Team Members	27 Nov 2021
Traceability Matrix	Test Manager	27 - 28 Nov 2021
Test Results	All Team Members	28 Nov 2021
Test Status report	Project Leader	28 Nov & 3 Dec 2021
Metrics	All team members	

Deliverable	For	Date / Milestone
Test Plan (Release Candidate?)	All Team Members, Participant	12 Dec 2021
Traceability Matrix	Test Manager	12 - 14 Dec 2021
Test Results	All Team Members	14 Dec 2021
Test Status report	Project Leader	14 & 18 Dec 2021
Metrics	All team members, Participant	

CHAPTER VI
USER MANUAL

6.1 Introduction

This User Manual was created so that users know how to use EverydayTasks and also the features contained in it. This user manual can also be used as a guide in testing the EverydayTasks application as well as filling out surveys or test sheets for the future development of the EverydayTasks application.

6.2 Basic Concepts

Activities are details of one activity carried out in one day. The contents are when the activity was carried out (automatically filled), activity title, description (if needed) and category (if needed). For example: when a User completes a class assignment, or after exercising, the User can record it through this feature.

Tasks are activities that are planned to be carried out at a time. Tasks can have a deadline or not. There is a title, description (optional) and category (optional). In EverydayTasks, there is a difference between deleting a task and completing a task:

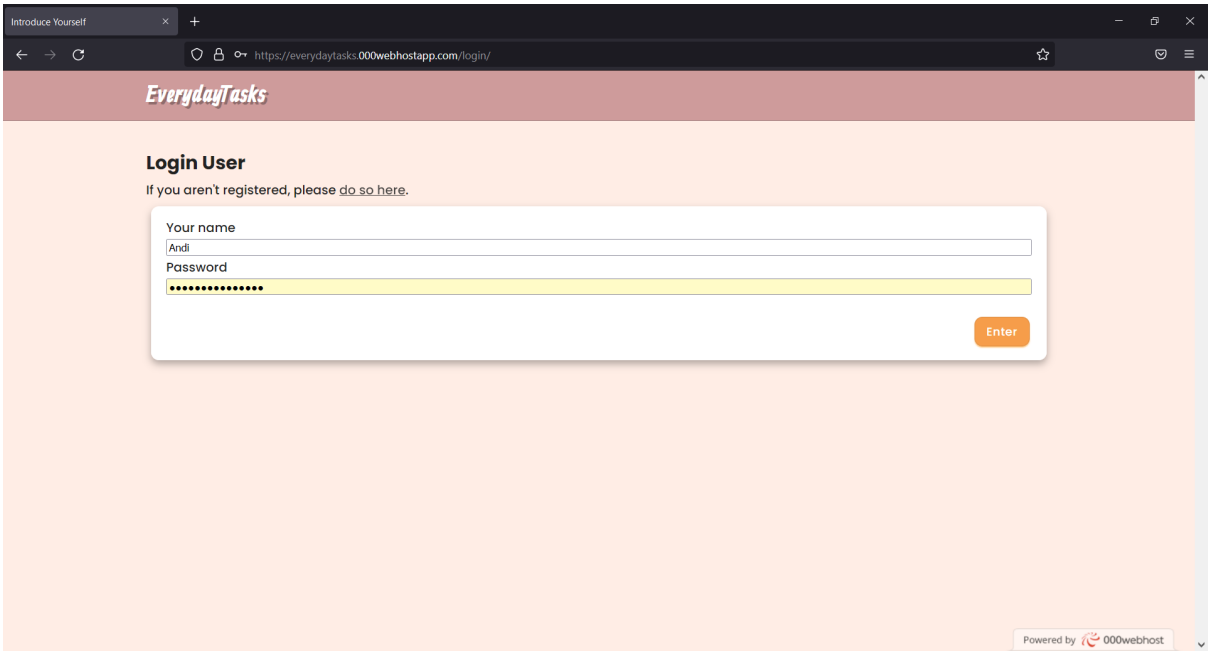
- Completing a **Task** will **create an Activity** that declares the completion of the Task, and that Activity will be associated with the Task. Users can access related Activities and Tasks (needs to be implemented).
- Deleting the **Task** will actually delete the Task. Completed tasks cannot be deleted (needs to be implemented).

Projects (needs to be implemented) is a list of Tasks that have the same topic. For example, writing a book can be made a Project. The project can have a Task for writing chapter 1, a separate Task for writing chapter 2, and so on.

Each Activity, Task and Project is owned by a User. A User can only see the Activities, Tasks and Projects they have.

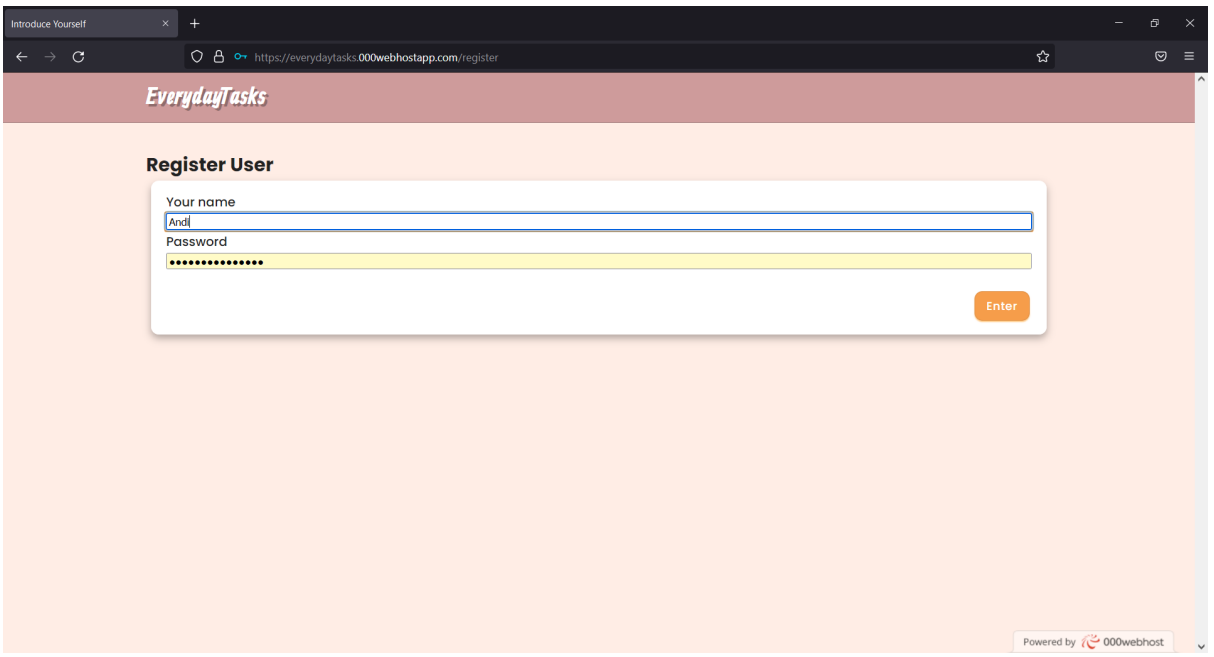
6.3 Operations

“Login” Page



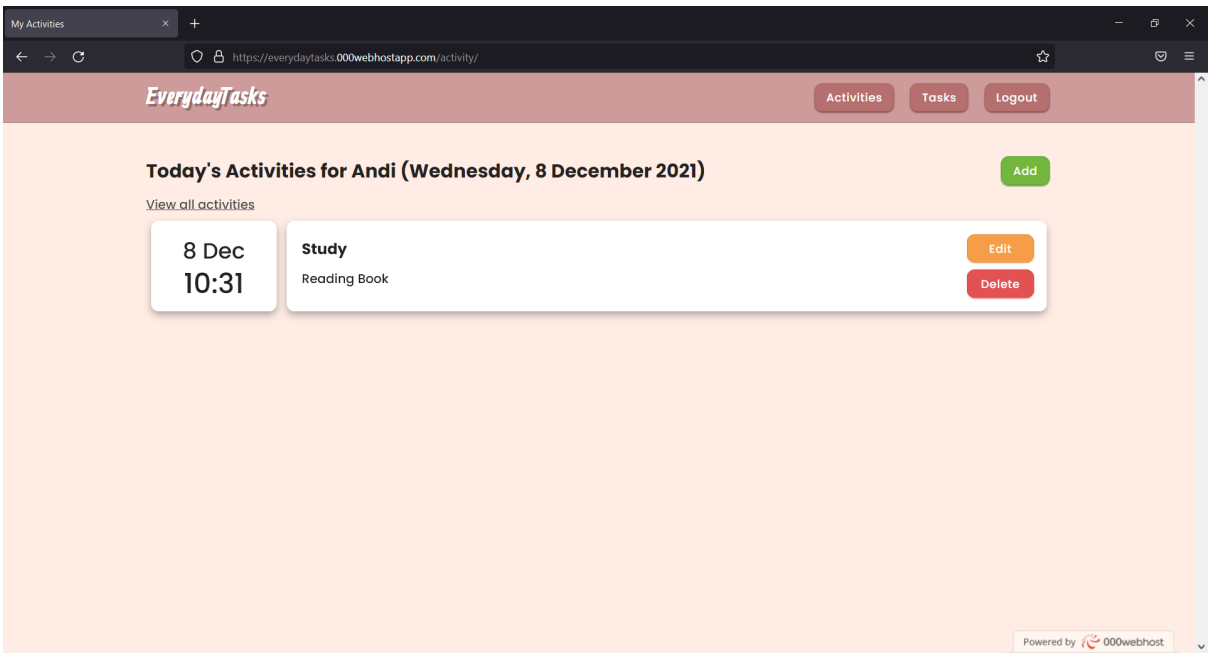
1. The user opens the application via the given link.
2. Users can press the “do so here” link to go to the “register” page and register their account.
3. If the user already has an account, the user can fill in the name and password in the text box provided and press the "Enter" button to log in.

“Register” Page



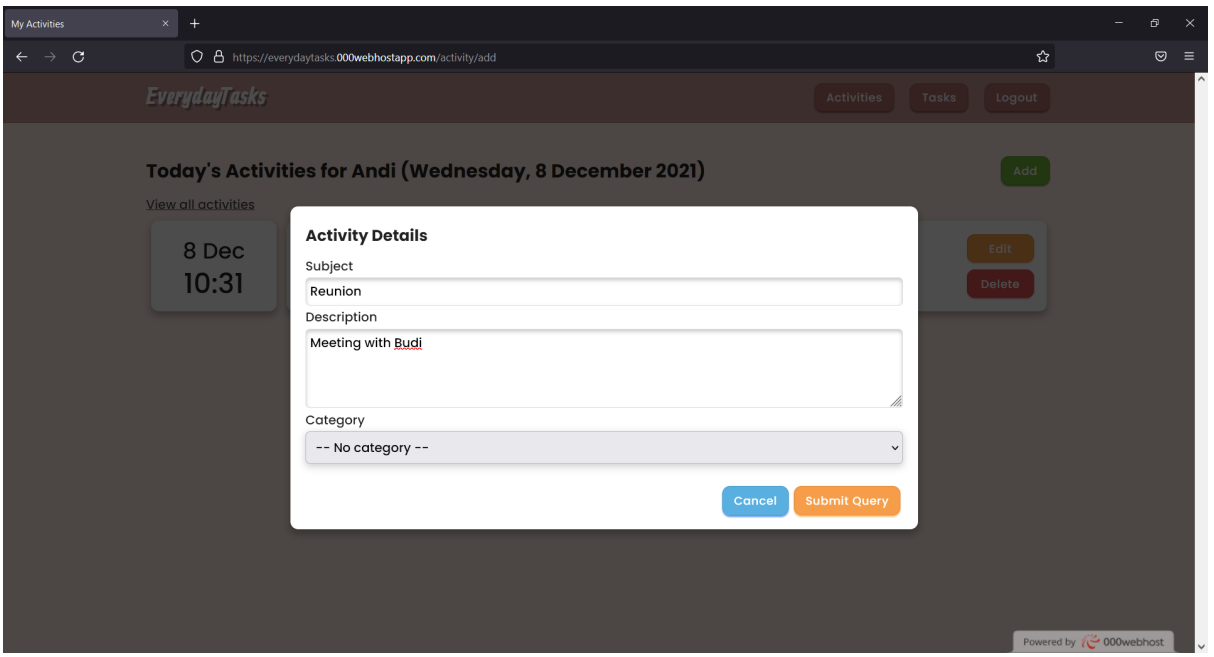
1. Users can fill in the name they want to use when logging in and operating the application in the text box provided.
2. Users can fill in the password they want to use when logging in in the text box provided.
3. The user can press the “Enter” button to register the account.

“Activity” Page - General



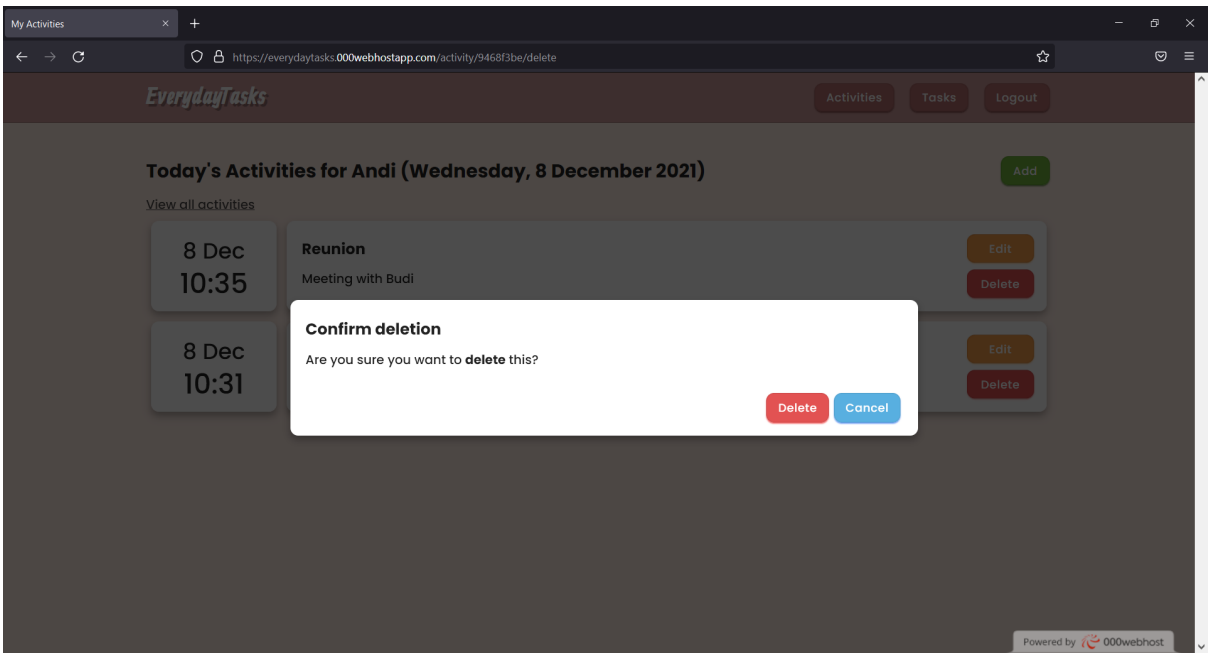
1. Users can go to the activity page in 2 ways, either automatically after logging in, or manually by pressing the “Activities” button.
2. Users can see the correct activity page, namely the name that matches the one entered on the “Login” page, the date that matches the date of that day, and the activity for that day.
3. Users can press the “Logout” button to return to the “Login” page.

“Activity” Page - Add/Edit Activity



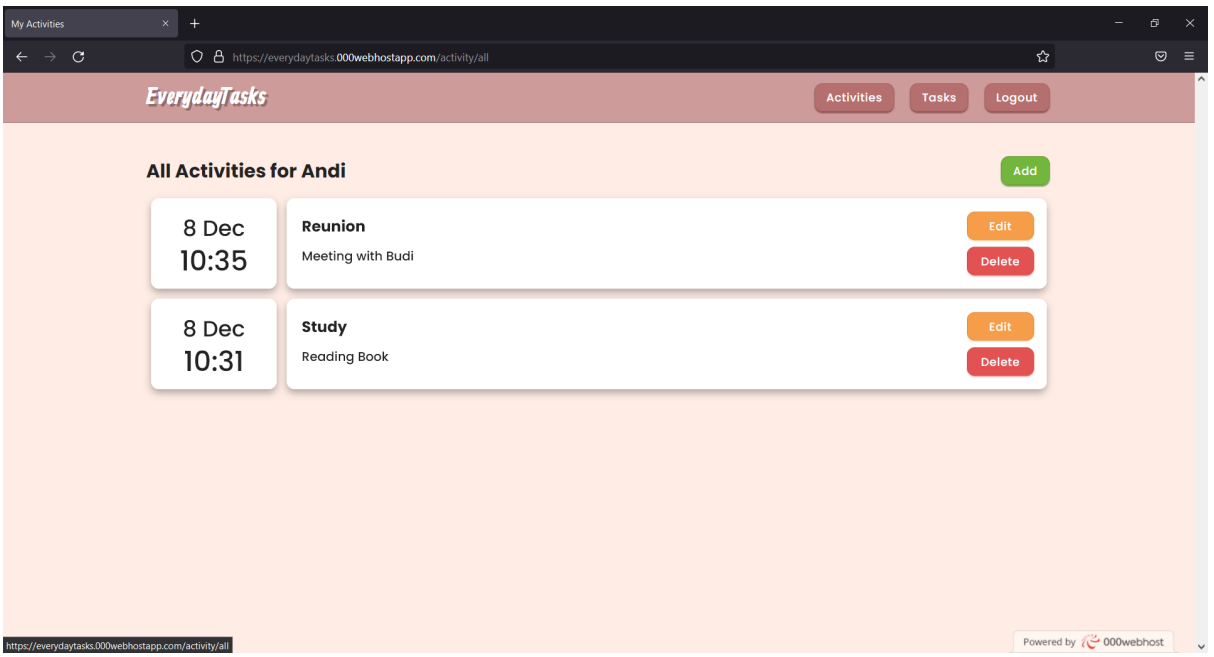
1. Users can press the “Add” button to add an activity for the day, or the “Edit” button to change an existing activity.
2. Users can fill in the title of the activity in the text box provided.
3. Users can fill in a description of the activity in the text box provided, or leave it blank.
4. The user can select a category from the activity in the drop-down box below, or leave it blank. There are various categories to choose from, namely "-- No category --", "Friends", "Work", "College", and "Play".
5. Users can press the "Cancel" button to cancel the addition/change of activities, or the "Submit Query" button to save the addition/change of activities.
6. Users can see the activities that have been added or have been changed according to the user's wishes. The time is listed next to their activity.

“Activity” Page - Delete



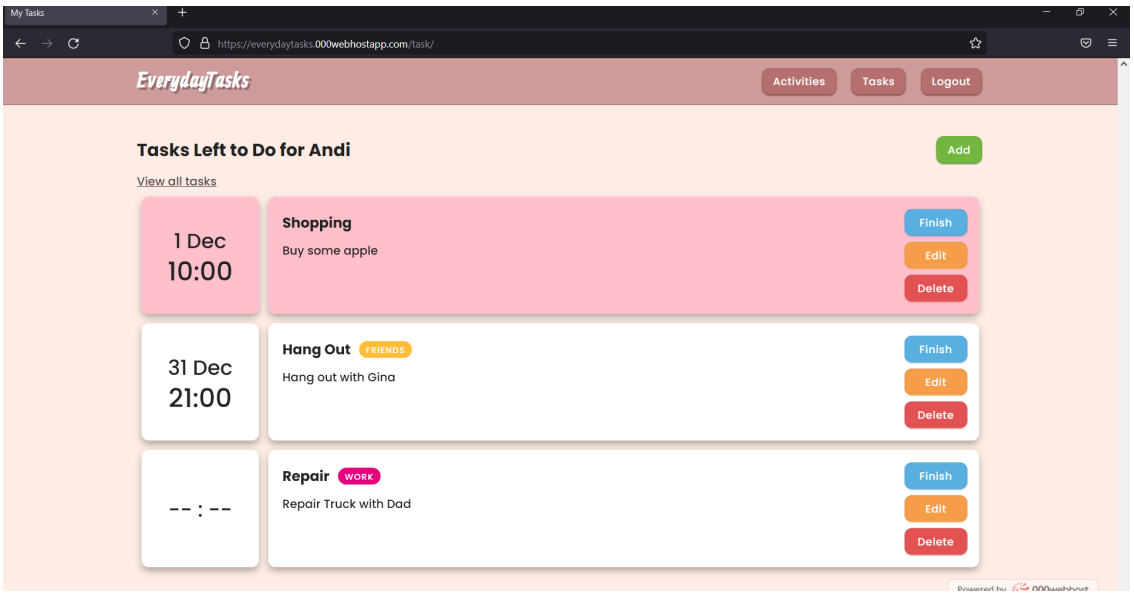
1. Users can press the “Delete” button to delete an activity for the day.
2. Users can press the "Cancel" button to cancel the deletion of the activity, or the "Delete" button to confirm the deletion of the activity.

“Activity” Page - All



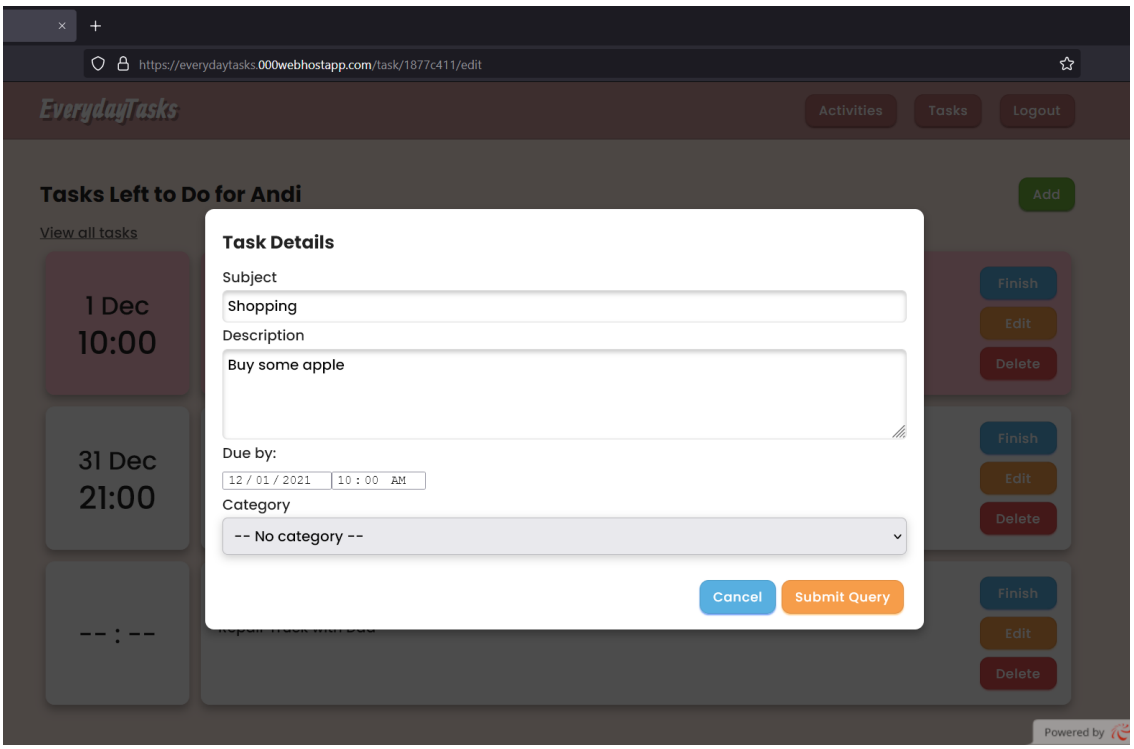
1. Users can press the "View all activities" button to display all activities that have been added both on that day and the previous day.
2. Users can use the buttons on the page according to the functionality described in the “Activity” Page - Add/Edit & “Activity Page - Delete.

“Tasks” Page - General



1. Users can go to the tasks page by pressing the “Tasks” button.
2. Users can see the correct tasks page, namely the name that matches the one entered on the “Login” page and the tasks that have not been done. Tasks that have not been completed and have passed that time will be pink, while those that have not passed that time will be white.
3. Users can press the “Logout” button to return to the “Login” page.

“Tasks” Page - Add/Edit Tasks

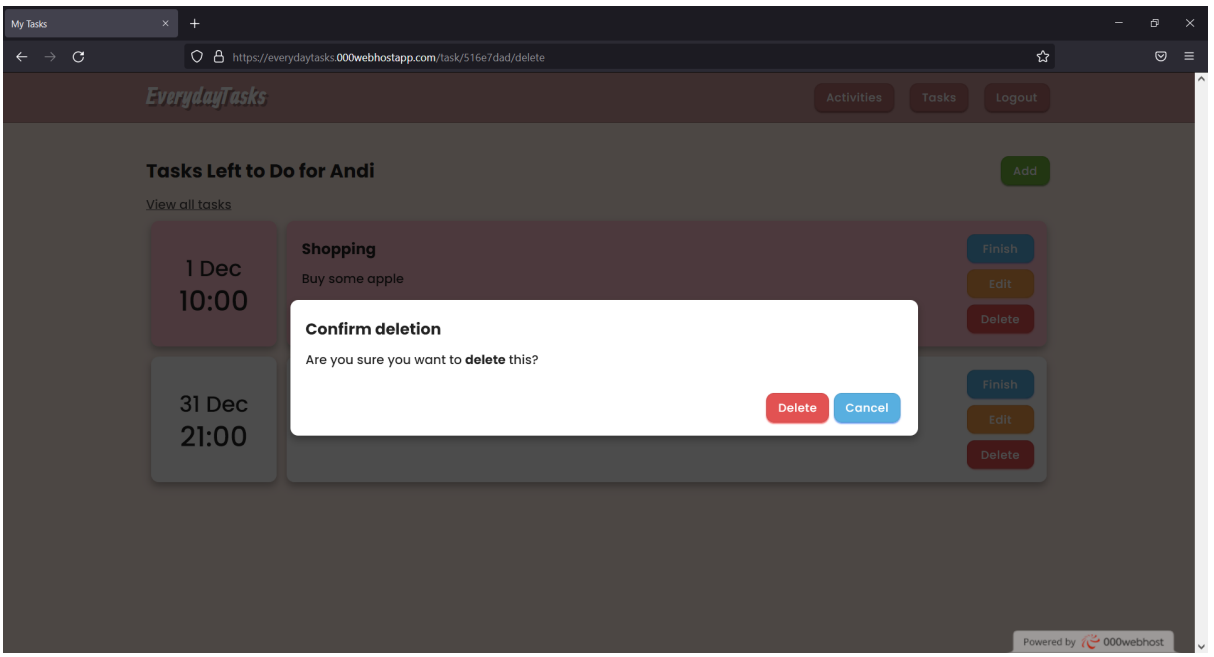


1. Users can press the “Add” button to add an unfinished task, or the “Edit” button to modify an existing task.
2. Users can fill in the title of the task in the text box below.
3. Users can fill in the description of the task in the text box below, or leave it blank.
4. Users can choose the deadline given to the task through the box provided, or leave it blank. On some devices, the time format is 12-hour so users can enter the time according to the 12-hour format (00.00 - 11.59 AM/PM).
5. The user can select a category from the task in the drop-down box below, or leave it blank. There are various categories to choose from, namely "-- No category --", "Friends", "Work", "College", and "Play"
6. Users can press the “Cancel” button to cancel the addition/change of tasks, or the “Submit Query” button to save the addition/change of tasks.
7. Users can see the tasks that have been added or that have been changed according to the user's wishes.

"Tasks" Page - Finish

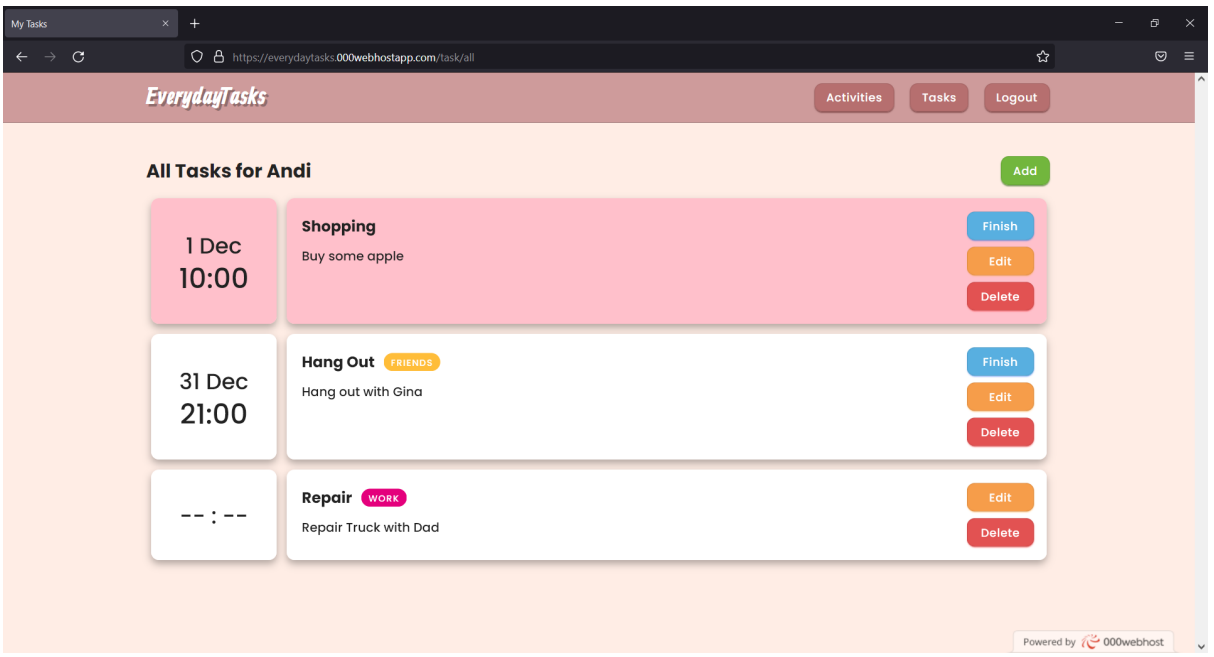
1. Users can use the "Finish" button to complete a task that has already been done. Completed tasks will be recorded as activities for the day.
2. Activities logged by tasks cannot be deleted directly on the "Activity" Page. If the user wants to delete the activity, then the user can press "Tasks" -> "View all tasks" button and delete the related activity.

“Tasks” Page - Delete



1. Users can press the “Delete” button to delete today's task.
2. Users can press the "Cancel" button to cancel the deletion of the task, or the "Delete" button to confirm the deletion of the task.

“Tasks” Page - All



1. Users can press the "View all tasks" button to display all the added tasks, both those that have been done and those that have not been done.
2. Users can use the buttons on the page according to the functionality described in the "Tasks" Page - Add/Edit, "Tasks" Page - Delete, and "Tasks" Page - Finish.

CHAPTER VII

USER SURVEY

7.1 Questions

The following is a copy of the survey questions:

7.1.1 Application Functionality

For each row, check if all of the conditions apply. Based on the result, write True/False. If you did not attempt a row, write “Not attempted”.

Pre-condition	**Condition**	**Post-condition**	**True/False/ Not attempted**
User is in the login page (/login)	User clicks on the **do so here** link.	User is led to the registration page (/register)	
User is in the login page (/login) and owns an account	User has filled out all necessary information on the page before clicking **Enter** button	User is successfully logged in	
User is in the registration page (/register)	User has filled out all necessary information on the page before clicking **Enter** button	User is successfully registered and led to the login page (/login)	
User is in the Activities page (/activity) or Tasks page (/task)	User clicks on the **Activity** button in the main menu	User sees only the activities logged for today (/activity)	
User is in the Activities page (/activity)	User clicks on the **Add** button	User is presented with a form that allows him/her to create an activity (/activity/add)	
User is in the Activities page (/activity)	User clicks on the **Edit** button on an activity.	User is presented with a form that allows him/her to edit an existing activity	

		(/activity/edit)	
User is in the Activities page and have an activity (/activity)	User clicks on the **Delete** button on an activity.	User is presented with a confirmation dialog for deleting an existing activity (/activity/delete)	
User is in the Activities page (/activity)	User clicks on the **View all activities** link	User is taken to a page listing all activities ever recorded (/activity/all)	
User is in the Activities page (/activity) or Tasks page (/task)	User clicks on the **Logout** button	The user is logged out and taken back to the login screen (/login)	
User has the add or edit activity dialog open (/activity/add, /activity/edit)	User is able to enter the correct subject and description in the proper boxes		
User has the add or edit activity dialog open (/activity/add, /activity/edit)	User is able to choose a suitable category from the drop down box		
User has the add or edit activity dialog open (/activity/add, /activity/edit)	User clicks on the **Cancel** button	The dialog closes and the user is returned to the Activities page	
User has the add or edit activity dialog open (/activity/add, /activity/edit)	User clicks on the **Submit Query** button	The dialog closes, the user is returned to the Activities page with his/her newly-created activity	
User is in the Activities page (/activity)	Activity data is correct and proper		
User is in the Activities page (/activity) or Tasks page (/task)	User clicks on the **Tasks** button	User is taken to a page listing unfinished tasks (/task)	
User is in the Tasks page (/task)	User clicks on the **Add** button	User is presented with a form that allows him/her to create a	

		task (/task/add)	
User is in the Tasks page and have an unfinished task (/task)	User clicks on the **Finish** button on a task		
User is in the Tasks page and have an unfinished task (/task)	User clicks on the **Edit** button on a task.	User is presented with a form that allows him/her to edit an existing task(/task/edit)	
User is in the Tasks page and have an unfinished task (/task)	User clicks on the **Delete** button on a task.	User is presented with a confirmation dialog for deleting an existing task (/task/delete)	
User is in the Tasks page (/task)	User clicks on the **View all tasks** link	User is taken to a page listing all tasks ever recorded (/task/all)	
User has the add or edit task dialog open (/task/add, /task/edit)	User is able to enter the correct subject and description in the proper boxes		
User has the add or edit task dialog open (/task/add, /task/edit)	User is able to choose the deadline with the clock box		
User has the add or edit task dialog open (/task/add, /task/edit)	User is able to choose a suitable category from the drop down box		
User has the add or edit activity dialog open (/task/add, /task/edit)	User clicks on the **Cancel** button	The dialog closes and the user is returned to the Activities page	
User has the add or edit task dialog open (/task/add, /task/edit)	User clicks on the **Submit Query** button	The dialog closes, the user is returned to the Tasks page with his/her newly-created task	
User is in the Tasks page (/task)	Task data is correct and proper		

7.2.2 User Satisfaction

Check the answer that applies.

Statement	Disagree	Slightly Disagree	Does not apply	Slightly agree	Agree
The user interface is functional and easy to use					
The features provided fit your needs					
The color combinations used are pleasant					
All parts of the user interface are laid out comfortably					

7.2.3 Suggestions

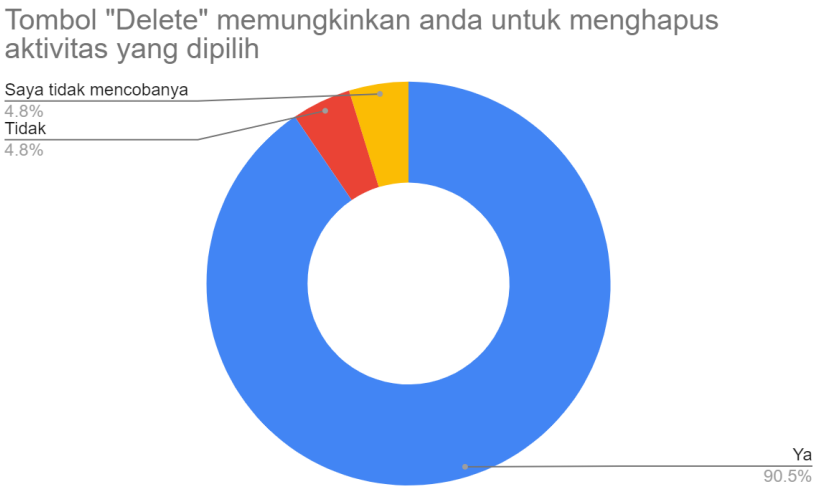
“Thank you for using our EverydayTasks App. Every feedback from you is precious for further development of our app in the future.”

7.2 Results

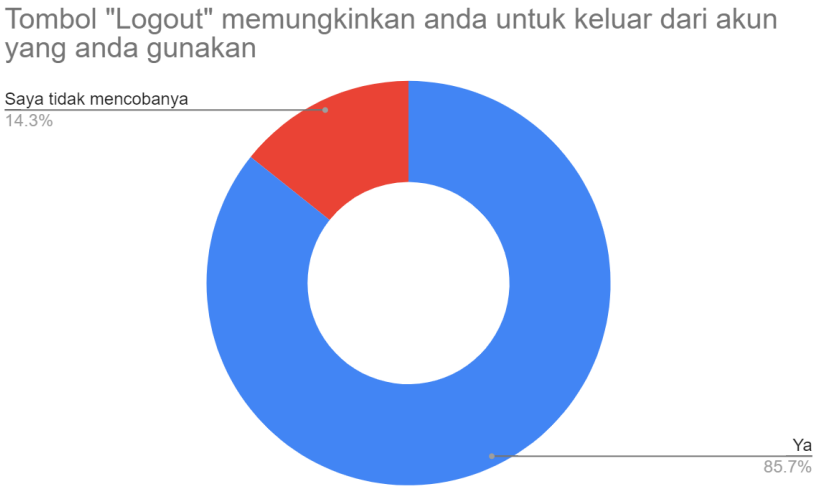
The survey was conducted through Google Forms between 15 December until 17 December 2021. The number of responses received as of 13:38, 17 December 2021 was 21. The results are shown as follows, grouped by each aspect of the app.

7.2.1 Functionality

From 21 responses, 21 respondents said that they successfully logged in with their account. This number also occurred on the register part, as well as on the part of adding activities/tasks and the proper view of the page.



From 21 responses, 19 respondents said that they can delete the activity of their choosing, while 1 respondent said that they didn't try the functionality, and another 1 respondent said that they can't delete the activity of their choosing, possibly because they choose to delete an activity that is created by tasks page and tried to delete it from activity page.



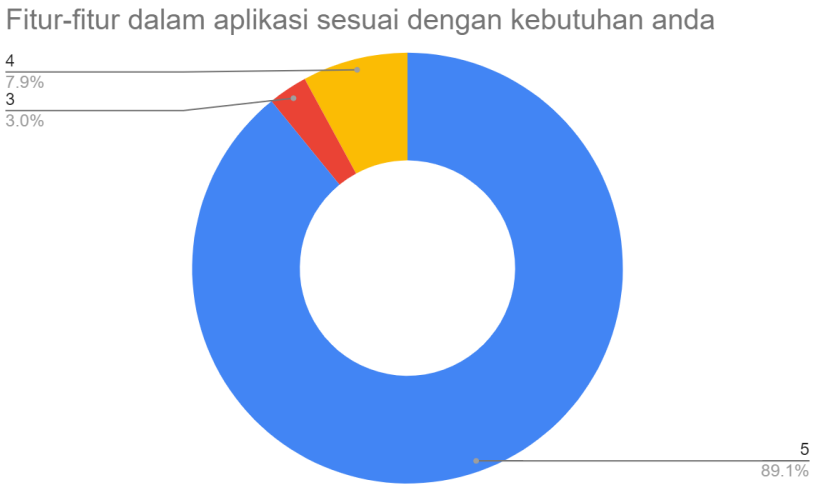
From 21 responses, 18 respondents said that they can log out using the “Logout” button, while 3 respondents said that they didn’t try the functionality. There are 2 possibilities of why this could happen

a. The user wasn’t interested on this feature, or
b. The user didn’t need to log out from their account because of their personal use, which is more relevant then the first possibility.

7.2.2 Satisfaction

From 21 responses, 21 respondents said that they are very satisfied with the layout in the app. This number also occurred on the satisfaction based by the combination of the color in the app.

From 21 responses, 16 respondents said that they can access the app easily, while 5 respondents said they can access the app with a small problem, possibly because this app doesn't very compatible with android



From 21 responses, 18 respondents said that the app's feature is suitable for their needs, while 3 respondents said that the app's feature is less suitable for their needs. This response is very valuable to us so that we can improve the app more in the future.

7.2.3 Suggestions

We receive some suggestions along with compliments for the app, and this suggestion is valuable to us for knowing what the user really needs and for future improvements on the app. Here we include some of the noticeable suggestions for the app:

a. Adding a calendar widgets to the app,

b. Make the app compatible with android, and
c. Adding an option to use a custom category for activity and task.

We also notice a minor “bug” mentioned by the respondent, which is that there are some conditions where the user cannot delete an activity made by task directly from the activity page, and we will discuss that matter in the future.

CHAPTER VIII

CONCLUSION

8.1 Conclusion

EverydayTasks is the first application that we created in collaboration. Our application is running properly and got a lot of good reviews, along with some suggestions for the application's further development. In our first project, the lesson we learned is to make a good schedule for our project, so that our goals are reached and the application is working as it's supposed to be. We need to know what is the most important for our project, and get some further improvement later on if we have enough time to do it.

8.2 Resources

We provided you with the link to our progress

- "Web Hosting" for the Application (ENG)
  https://everydaytasks.000webhostapp.com/login/
- "Youtube" for Video Presentation

  https://www.youtube.com/watch?v=Q55YoTAI-gg
- "Google Slide" for our Presentation (ENG)
  https://docs.google.com/presentation/d/10wrm3gajMED1a2LFfRC09JqrioBk4_Ef5CVIMnENZB8/edit?usp=sharing
- "Google Docs" for Application's User Manual (IDN)
  https://docs.google.com/document/d/1Esuq6QBxR2KO1oe6pgmKMD3NHNpcP1mBIXhEIb82D9U/edit?usp=sharing
- "Google Form" for Application's Survey (IDN)
  https://forms.gle/Lo8TYuZocTGUKR2s9
- "Google Sheet" for Survey's Result (IDN)
  https://docs.google.com/spreadsheets/d/1p9oN4H2yEU0aTmPbBnhaacqquW1WWxzFMPH5s0vTFuk/edit?usp=sharing
- "Github" for Application's Back End Progress (ENG)
  https://github.com/rbsoen/EverydayTasks

- “Figma” for Application’s Front End Progress (ENG)
  https://www.figma.com/file/15PxZqMsq2llbRSofov6oj/EverydayTasks
- “Trello” for our Project Management (ENG)
  https://trello.com/b/yOYBjY8p/group-13-everydaytasks

REFERENCES

1. M. Chappal. “Personal Productivity Guide: How to Maximize Your Time & Productivity Levels”. Friday. [Online] Available: https://friday.app/p/personal-productivity
2. D. Pham. “PHP – Strengths and Weaknesses”. (Jun. 13, 2020). Ryadel. [Online] Available: https://www.ryadel.com/en/php-programming-language-strengths-weaknesses/
3. S. Bocetta. “Comparing 3 open source databases: PostgreSQL, MariaDB, and SQLite”. (Jan. 15, 2019). Opensource.com. [Online] Available: https://opensource.com/article/19/1/open-source-databases
4. Crowdsource. “Common Nginx misconfigurations that leave your web server open to attack”. (Nov. 10, 2020). Detectify. [Online] Available: https://blog.detectify.com/2020/11/10/common-nginx-misconfigurations/
5. OWASP ZAP. “Alerts”. [Online] Available: https://www.zaproxy.org/docs/desktop/start/features/alerts/
6. A. Ivankov. “Ubuntu Operating System: Advantages and Disadvantages” (Jun. 23, 2020). Profolus. [Online] Available: https://www.profolus.com/topics/ubuntu-operating-system-advantages-and-disadvantages/
7. RJ Systems. “Windows pros and cons”. (Aug. 2, 2017). [Online] Available: http://www.rjsystems.nl/en/3200.php
8. MDN Contributors. “GET - HTTP | MDN”. (Aug. 13, 2021). Mozilla Developer Network. [Online] Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET
9. MDN Contributors. “PUT - HTTP | MDN”. (Aug. 13, 2021). Mozilla Developer Network. [Online] Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT
10. MDN Contributors. “DELETE - HTTP | MDN”. (Aug. 13, 2021). Mozilla Developer Network. [Online] Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/DELETE
11. MDN Contributors. “POST - HTTP | MDN”. (Aug. 13, 2021). Mozilla Developer Network. [Online] Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST

12. REST API Tutorial. “HATEOAS Drived REST APIs”. (Oct. 10, 2021). [Online] Available: https://restfulapi.net/hateoas/
13. D. Taylor. “Heroku vs AWS: What is the Difference?” (Oct. 6, 2021). Guru99. [Online] Available: https://www.guru99.com/heroku-vs-aws.html