



UNIVERSITY OF THESSALY
SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

**MULTI-PURPOSE PLATFORM FOR
COLLABORATIVE PRODUCTIVITY**

Diploma Thesis

Kyriakou Georgia

Supervisor: Hariklia Tsalapatas

July 2023



UNIVERSITY OF THESSALY
SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

**MULTI-PURPOSE PLATFORM FOR
COLLABORATIVE PRODUCTIVITY**

Diploma Thesis

Kyriakou Georgia

Supervisor: Hariklia Tsalapatas

July 2023

iii



ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

**ΠΛΑΤΦΟΡΜΑ ΠΟΛΛΑΠΛΩΝ ΧΡΗΣΕΩΝ ΓΙΑ
ΣΥΝΕΡΓΑΤΙΚΗ ΕΝΙΣΧΥΣΗ ΠΑΡΑΓΩΓΙΚΟΤΗΤΑΣ**

Διπλωματική Εργασία

Κυριακού Γεωργία

Επιβλέπων/πouσα: Χαρίκλεια Τσαλαπάτα

Ιούλιος 2023

Approved by the Examination Committee:

Supervisor **Hariklia Tsalapatas**

Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Member **George Stamoulis**

Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **George Thanos**

Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

I would like to express my sincere gratitude to my thesis advisor, Professor Hariklia Tsalapatas, for her invaluable guidance and support throughout the research process. Her expertise and encouragement has played a significant role in the successful completion of my thesis.

I am deeply grateful to my family for their support, patience and understanding. They have been my constant source of strength and inspiration while leading the way to my academic growth, and i cannot thank them enough.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Kyriakou Georgia

Diploma Thesis

MULTI-PURPOSE PLATFORM FOR COLLABORATIVE PRODUCTIVITY

Kyriakou Georgia

Abstract

The rapid adjustment to emerging technologies and the turn to remote work, has brought up the issue of focus deficiency. More and more people search for tools that may help in their psychological state towards attention span growth. The solution may again be found in technology with the development of applications focusing on productivity. The diploma thesis aims at the creation of a web application that is secure, personalized and includes a variety of tools or productivity techniques. The scope of collaboration and a gamified approach used as a motivation factors, along with private progress tracking can be a great solution to the problem, while keeping the user in a familiar environment. The final interface is the result of a MongoDB, Express, React and Node full stack application which is based on many powerful noteworthy front-end and back-end services. The ultimate goal of the thesis also incorporates research on the variety of methodologies available and hands-on experiment on these technologies that were considered useful for the process.

Keywords:

productivity, gamified, web application, MongoDB, Express, React, Node, full stack

Διπλωματική Εργασία

ΠΛΑΤΦΟΡΜΑ ΠΟΛΛΑΠΛΩΝ ΧΡΗΣΕΩΝ ΓΙΑ ΣΥΝΕΡΓΑΤΙΚΗ ΕΝΙΣΧΥΣΗ ΠΑΡΑΓΩΓΙΚΟΤΗΤΑΣ

Κυριακού Γεωργία

Περίληψη

Η ταχεία προσαρμογή στις αναδύμενες τεχνολογίες και η στροφή προς την εξ αποστάσεως εργασία, ενίσχυσαν το ζήτημα της ελλειμματικής προσοχής. Όλο και περισσότεροι άνθρωποι αναζητούν εργαλεία που μπορεί να τους βοηθήσουν ψυχολογικά όσον αφορά την προσήλωση σε κάποια διαδικασία. Η λύση μπορεί και πάλι να βρεθεί στην τεχνολογία με την ανάπτυξη εφαρμογών που εστιάζουν στην ενίσχυση της παραγωγικότητας του ατόμου. Η διπλωματική εργασία στοχεύει στη δημιουργία μιας εφαρμογής ιστού που είναι ασφαλές, εξατομικευμένο και περιλαμβάνει διάφορα εργαλεία ή τεχνικές παραγωγικότητας. Το χαρακτηριστικό της συνεργασίας και η παιχνιδοποίηση που μπορούν να λειτουργήσουν ως κινητήριος δύναμη, μαζί με την προσωπική παρακολούθηση προόδου, μπορεί να είναι μια εξαιρετική λύση στο πρόβλημα, μέσα σε ένα οικείο εικονικό περιβάλλον. Η τελική διεπαφή του χρήστη είναι το αποτέλεσμα μίας εφαρμογής πλήρους στοίβας με τη χρήση των MongoDB, Express, React και Node και των ισχυρών εργαλείων που αυτά προσφέρουν. Ο απώτερος στόχος της διατριβής περιλαμβάνει επίσης την έρευνα σε διαθέσιμες μεθοδολογίες και πρακτική εξάσκηση στις τεχνολογίες που επιλέχθηκαν για τη διαδικασία.

Λέξεις-κλειδιά:

παραγωγικότητα, παιχνιδοποίηση, εφαρμογή ιστού, MongoDB, Express, React, Node, πλήρους στοίβας

Table of contents

Acknowledgements	ix
Abstract	xii
Περίληψη	xiii
Table of contents	xv
List of figures	xvii
List of tables	xix
Abbreviations	xxi
1 Introduction	1
1.1 Project Overview	1
1.2 Purpose	2
1.2.1 Contribution	2
1.3 Thesis Structure	2
2 Current State	5
2.1 Problem to be solved	5
2.2 The Psychological Scope	6
2.3 Workspace, Education and Personal Solutions	7
3 Related Work	9
3.1 Existing Applications	9
3.2 Task Management Applications	9

3.2.1	Todoist	9
3.2.2	Trello	10
3.2.3	ClickUp	10
3.3	Time Management Applications	12
3.3.1	Pomodoro - Focus Timer	12
3.3.2	Forest: Focus for Productivity	13
3.3.3	Focusmate	13
3.4	A New Solution	14
4	Methodology	17
4.1	Agile Software Development	18
4.2	Requirements	19
4.3	Design	21
4.4	Implementation	23
4.5	Testing	23
4.6	Deployment	25
5	Application Development	27
5.1	Asynchronous Programming	28
5.2	Back-End Development	30
5.2.1	Additional Libraries	32
5.3	Front-End Development	35
5.3.1	Additional Libraries	38
5.4	Implementation	39
5.4.1	REST API Documentation	39
5.4.2	MongoDB Collections	42
5.4.3	Application Overview	44
6	Conclusion	57
6.1	Summary	57
6.2	Future Work	58
	Bibliography	61

List of figures

2.1	The Eisenhower Matrix in Theory	6
3.1	Todoist Preview	10
3.2	Trello Preview	11
3.3	Clickup Preview	11
3.4	Pomodoro - Focus Timer	12
3.5	Forest App Preview	13
4.1	Agile Methodology	18
4.2	MVC Graphically	23
5.1	Home Page	44
5.2	Register Page	45
5.3	Login Page	45
5.4	Send Email to Reset Password	46
5.5	Link to Updating Password	46
5.6	Link to Confirm Valid Email	47
5.7	Profile Page	48
5.8	Profile Edit Page	48
5.9	Pomodoro Room Page	49
5.10	Pomodoro Features Page	49
5.11	Pomodoro Chat, Different Members Left and Right	50
5.12	Pomodoro Settings Change is Broadcasted	50
5.13	Leaderboard Page	51
5.14	Private Tasks Page	52
5.15	Add New Task	52

5.16	Tasks Edit Feature	53
5.17	Tasks Search Feature	53
5.18	Priority Matrix Page	54
5.19	Calendar Page	55
5.20	Calendar Add New Task	55

List of tables

4.1	User Collection Schema	19
4.2	User Collection Schema	20
5.1	User Collection Schema	42
5.2	Leaderboard Collection Schema	43
5.3	Tasks Collection Schema	43
5.4	Rooms Collection Schema	44

Abbreviations

ADHD	Attention Deficit Hyperactivity Disorder
JS	JavaScript
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
HTTP	HyperText Transfer Protocol
DOM	Document Object Model
URL	Uniform Resource Locator
MERN	MongoDB Express.js React Node.js
REST	Representational State Transfer
API	Application Programming Interface
JSON	JavaScript Object Notation
UI	User Interface
SQL	Structured Query Language
NoSQL	Not only Structured Query Language
JWT	JSON Web Token
ODM	Object Data Modeling
JSX	Javascript Extensible Markup Language
SVG	Scalable Vector Graphics
ODM	Object Data Modeling
etc.	Et cetera

Chapter 1

Introduction

1.1 Project Overview

This specific thesis is an approach to explore the upcoming web technologies that may be an efficient solution to optimize the productivity of a person. In times of remotely learning and working, the demand for better focus and time management has emerged as opposed to several problems in evaluation of abilities. However we cannot deny the convenience that follows even after the pandemic, since many of these changes are the new normal.

Therefore, the adjustment to this new era of online work requires that people prioritize self-improvement and self-evaluation with the help of technology. A set of everyday tasks are certain to help reach the ultimate goal that anyone sets in their life and it is why many task management applications face rapid growth. It is crucial that the person identifies the value of their time and the ability to effectively utilize it by setting reachable goals.

The programming part of the research includes the development of the application 'PomoShare'. The development of this application consists of the User Interface which is built in React, the MongoDB database for all the essential information that has to be stored and the custom RESTful API that was particularly created according to the backend needs for the app. The main functionality of PomoShare is the Pomodoro Timer technique with an accountable partner that is assigned to each participant based on a match-making algorithm. The aim is to attach gamification to the app, and therefore evaluation metrics are included. The user can also privately manage their everyday tasks in a Priority Matrix or Calendar that dynamically changes based on their needs.

1.2 Purpose

As mentioned above, the time and task handling in the online world are becoming obstacles in the overall growth, either academic or professional. Along with the lack of motivation that isolation brings about, or even the procrastination due to working poorly organized from a familiar environment, it is crucial that everyone uses a system that will bring out the best version of themselves.

In this research the goal is to create software that is fun and includes gamified and collaborative features leading to solid remote work foundations. After a search on existing platforms that are commonly used for goal mapping, as well as the solutions of Psychology on the topic, a combination of both has led to an innovative idea for a social platform. In particular, a web application is developed and extensively tested on a variety of approved psychological techniques, both on a personal and on a shared level.

1.2.1 Contribution

In this section the aim is to list the decision-based process of developing the final software application. Starting from the initial motive, up to the final tests that were applied, the system that was followed is depicted in chronological order.

The contribution to the thesis is presented below:

1. Distinguish the reason why such application would be helpful.
2. Study on the psychological techniques that are being utilized.
3. Evaluation of existing software that is commonly used for the purpose.
4. Combine the systems and brainstorm on a gamified and collaborative version.
5. Choose among the available frameworks that suit best for such application.
6. Implementation and testing of platform.

1.3 Thesis Structure

As an overview of the thesis, it is worth mentioning the fundamental sections and the distinct steps of the process. The following section, Chapter 2 presents the current state of

the problem. In other words, the ways in which time and tasks are handled in the case of problematic focus or organization, specifically psychologically wise. Then, in Chapter 3, follows a brief showcase of the web products that are already launched and have been the main source of inspiration for the thesis. The methodology of software development comes after in Chapter 4 and how it is suitable for this case. Chapter 5 depicts the actual application that was developed in detail, while Chapter 6 summarizes the general results obtained, along with future upgrades of the platform.

Chapter 2

Current State

2.1 Problem to be solved

It is an undeniable fact that the quarantine era due to COVID-19 was a period of re-adjustment in workspaces and teaching environments. The obligation for isolation has brought up the need for working from home, but the expectation of equivalent productivity as when working on-site was not realistic. Human interaction and communication, the distinction between the job or school and the personal life, the mental challenges that followed for almost everyone, have made an impact on a person's growth. However, this adjustment is proven beneficial for multiple reasons and is justifiably a change that is adapted by many workspaces, even after a few years from the end of lockdowns.

Therefore, after this period of time, people find similarities with people struggling with Attention Deficit Hyperactivity Disorder on some symptoms. ADHD is a neurodevelopment disorder that is accompanied with trouble staying organized and focusing, forgetfulness and even procrastination, among other. Those that are diagnosed with ADHD do not have any less potential than everyone else, it is with the help of a system that their strengths will be enhanced and will stay on track.

In any case, the productivity of a person remains a problem in today's world. The distractions are multiple, and the information is limitless, while the time and the attention seem to be untouched. Working and learning remotely or in hybrid mode is the new normal thanks to technology and its aim to make our lives better. Consequently, the issue is to be solved with personal effort, and the help of mental health and additional guidance.

2.2 The Psychological Scope

By recognizing and resolving underlying psychological problems that may be causing poor performance, psychology is becoming an increasingly significant tool in helping people increase their productivity. People can learn the skills and methods necessary to overcome productivity obstacles and accomplish their goals by working with experienced specialists and using evidence-based techniques.

Psychologists often use the concept of Mindfulness. Being in the present without judgment is the practice of mindfulness. People who use it can lessen stress and anxiety, which can have a detrimental effect on productivity. People can improve their focus, concentration, and attention span by being more mindful, which will raise their productivity.

Additionally, to increase productivity, psychology highlights the value of creating reasonable, reachable goals. People can feel a sense of success and drive by dividing large tasks into smaller, more manageable ones, which can enhance productivity. Specifically, these to-do lists can be properly prioritized based on urgency and importance, as in the four quadrants of Priority/Eisenhower Matrix shown in fig 2.1. It also offers methods like the Pomodoro technique [1], which divides work into periods of 25 minutes and ends with a five-minute break, to assist people in better time management and distraction reduction.

The Eisenhower decision matrix

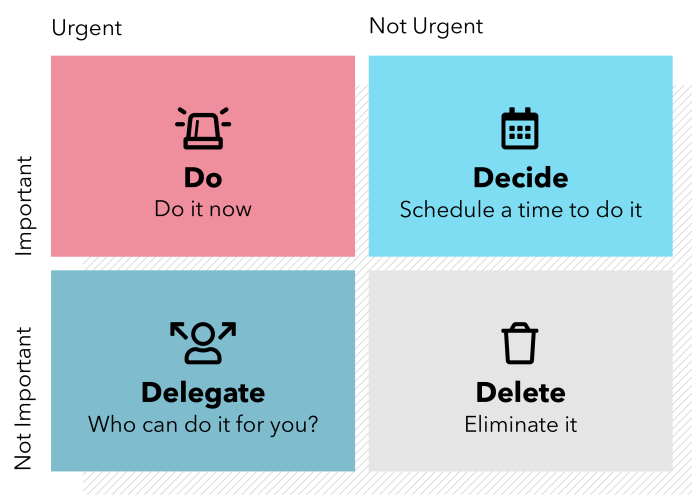


Figure 2.1: The Eisenhower Matrix in Theory

2.3 **Workspace, Education and Personal Solutions**

Psychology not only offers individuals techniques, but it also has an impact on workplace practices and regulations [2]. Employers are putting programs and policies in place that support a pleasant workplace culture because they understand how important employee mental health and wellness are. Employers can promote a productive workplace culture that is advantageous to both the firm and the employees by putting a priority on employee mental health and employing evidence-based strategies. Between a teacher and a student, similar methodologies occur.

One strategy is to include active learning methods that encourage student engagement and participation, such as group projects, in-class debates, and hands-on activities. This keeps students more engaged in the learning process and attentive. A collaborative learning environment with defined rules and expectations is another strategy that can assist prevent distractions and keep students focused. To make learning more interactive and interesting, teachers can also use technology in the classroom, such as interactive whiteboards and educational apps.

To teach students how to utilize technology more efficiently and prevent being distracted by it, many schools are introducing digital literacy training into their curricula. Teachers want to help kids form healthy digital habits and reduce distractions by teaching them how to use technology in a focused and productive way.

Individuals must be able to efficiently manage their time and remain focused in the absence of external structure or supervision, which calls for a high level of self-regulation. To combat this, people are employing a variety of techniques to stay concentrated and on task while working or learning remotely. Establishing a schedule, setting up a dedicated workplace, employing time management tools, taking breaks, using collaborative and accountable software [3], wearing noise-canceling headphones, and reducing distractions are some of these tactics. People can retain productivity and accomplish their goals in a remote work setting by using these tactics.

Chapter 3

Related Work

3.1 Existing Applications

The idea for some sort of software that enhances productivity is not an innovative case [4]. Assisitive applications are commonly used even on a daily basis since every smartphone in our days carries a compatible calendar. The calendar is the old-fashioned way of keeping everything on track, and making sure that nothing gets left behind with the help of notifications and reminders. This plain form of productivity software is almost fundamental. However, it seems as though it lacks creativity and, thus, is not motivational.

For this reason, many businesses have come up with ideas that not only intrigue the audience but also encourage them to stay consistent in the journey of attention span growth. The following two sections include some representative examples [5] of the existing applications, either web or mobile, which are divided into time and task management.

3.2 Task Management Applications

3.2.1 Todoist

Todoist 3.1 is a web, mobile and desktop application that aims to help individuals on productivity and self-improvement. Its main dashboard includes non-complex task arrangement and labeling not only for personal use, but also for teams and projects. The teams are able to assign a task to a team member, divide the work into simpler goals and discuss about it in a comments section. Todoist along with the task lists also offers a set of reminders, a calendar

for each task's due date and various filters for each task that may depict the importance or urgency of a task. One of the key advantages of the platform is the integration with commonly used productivity tools such as Dropbox, Google Calendar, etc., making it very easy for the user to view all their tasks in one dashboard.

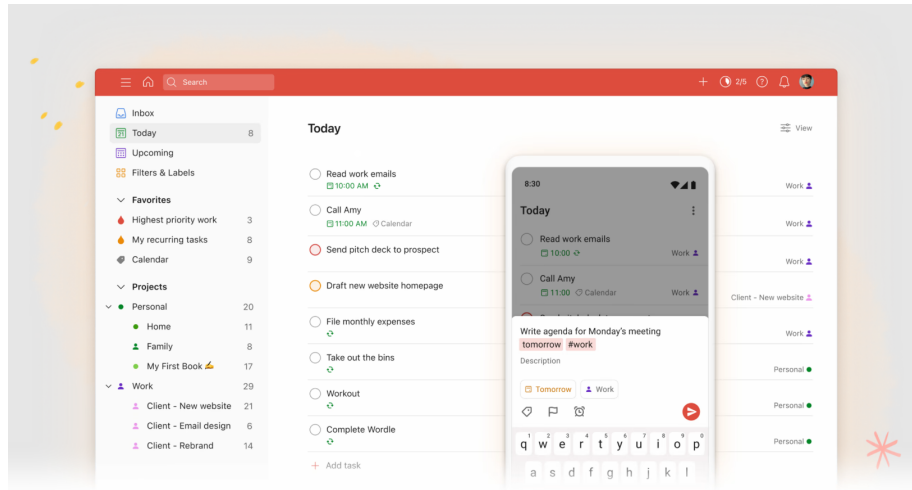


Figure 3.1: Todoist Preview

3.2.2 Trello

Trello is a useful tool for productivity enhancement in terms of a group project. The fundamental distinction compared to other platforms is the board and card system 3.2 for teams to organize and track their project sub-tasks. Each user can create a board, add a card for every goal and share it with their team. A card can be dragged and dropped to the suitable section according to the status of the task that it includes, selecting among "To Do", "In Progress" and "Completed".

Hence the team can prioritize work by visualizing the progress. Trello establishes an on-line environment of collaboration with extra features like assigning tasks to members, commenting on work progress, attaching files, and many more. Similarly to Todoist, Trello also communicates with the most popular calendars or tools available for task management.

3.2.3 ClickUp

ClickUp 3.3 is an adjustable productivity platform with functionality like Trello. The specialty of this application, which comes in any device, is workflow optimization. Depending

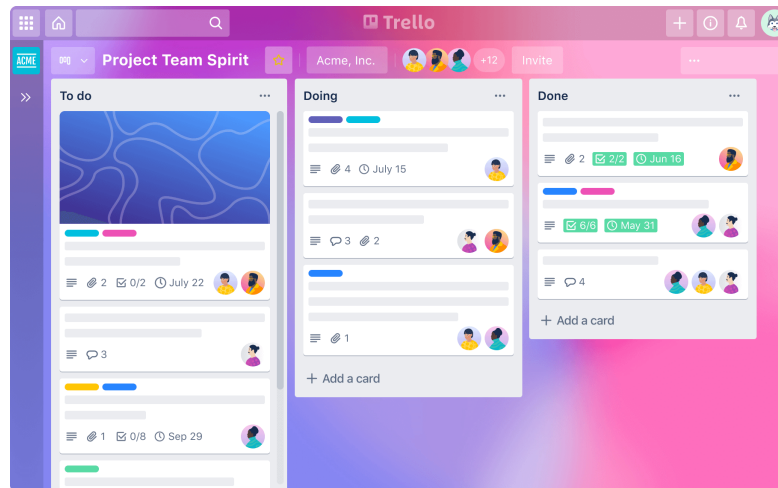


Figure 3.2: Trello Preview

on the project management needs, a set of features may be utilized, from setting due dates and distributing the various parts of the work to the members, up to timeline and progress tracking using dashboards. In this case, users can choose among certain visualisations, making it suitable for any preference on task organization. Clickup is different from other similar platforms because of the various automation features that it offers, as well as customizable templates and reporting capabilities, encouraging users to automate standard processes and perform better on their responsibilities.

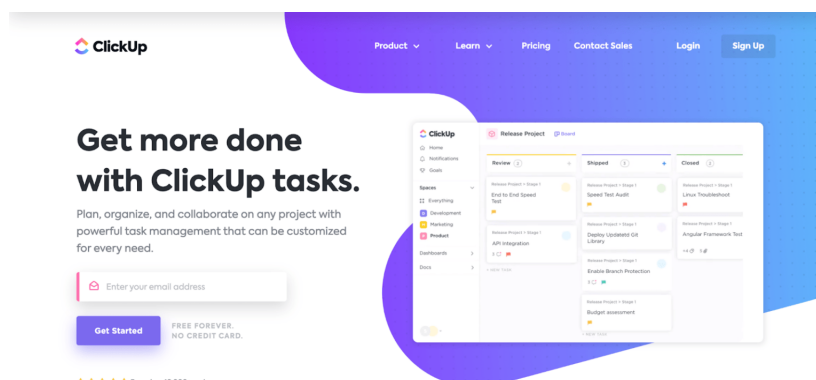


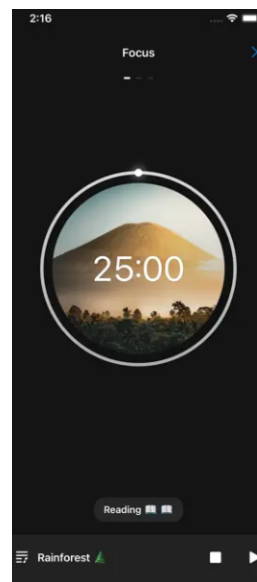
Figure 3.3: Clickup Preview

3.3 Time Management Applications

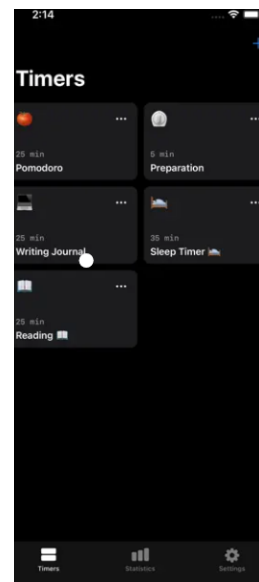
3.3.1 Pomodoro - Focus Timer

The Pomodoro - Focus Timer app 3.4 is a mobile productivity tool that follows the Pomodoro Technique. The main goal of the app is to help users increase their attention levels through interval focusing, typically 25 minutes of work and 5 minutes of break on repeat one after the other. In other words, it is a simple countdown timer but is customizable according to the user's needs. The sessions can be categorized based on the settings a user chooses for the intervals, like the duration of focusing or taking a break, as well as the topic of their current work, making it organized and motivational.

The application incorporates additional analytics like session history and progress, so that there is beneficial self-evaluation by looking back on work patterns. It is generally free, but the premium version includes ambient sounds of white noise, which is a great block distraction tool, as well. The demonstrated version is compatible with OS system devices, but there exist alike options for any other device type.



(α') Focus Session



(β') Timers

Figure 3.4: Pomodoro - Focus Timer

3.3.2 Forest: Focus for Productivity

'Forest: Focus for Productivity' 3.5 refers to a Pomodoro application that is launched in the form of mobile app and as an add-on extension for the most popular web browsers. It is a productivity app that shares common ground with the plain Pomodoro application, mentioned in the previous section, but takes the idea to the next level using gamification. In this case, the producers came up with the innovation of using a reward system for every successful work and study session during the Pomodoro technique.

The app contains a virtual forest, and the reward is basically planting trees and enriching the forest, only if the user is dedicated to their work periods without distractions. During these periods, the algorithm that runs in the background checks if the user is distracted by their smartphone or accesses any blacklisted websites like social media during work time. If that happens, the tree withers and dies and virtual coins are subtracted. This gamified approach encourages the resistance to any source of distractibility, while making an impact to the environment due to the app's partnership with tree-planting organizations and the ability to plant a real tree if an number of virtual coins is collected.

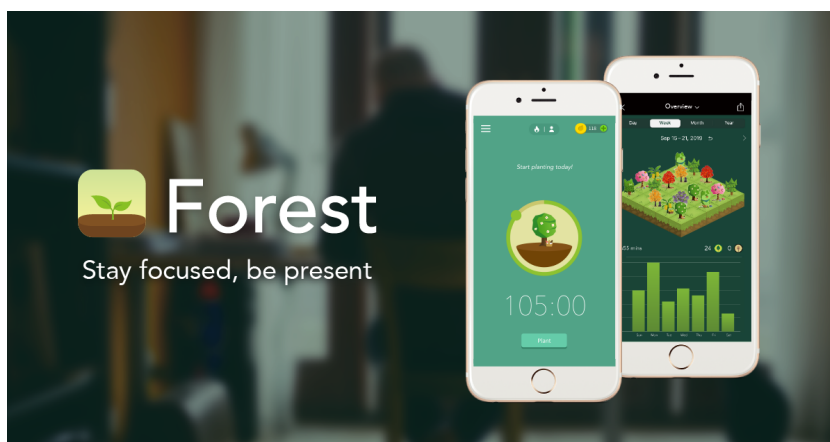


Figure 3.5: Forest App Preview

3.3.3 Focusmate

Focusmate is a time management online platform that once again benefits from the Pomodoro Technique. The work and break sessions are based on coworking which means that two virtual users schedule a meeting and can work on their own tasks with an accountability partner. To make it a more supportive and connective virtual workplace, the app producers

included video calls and chat encouraging communication of thoughts and goals between the participants.

Users are able to overcome procrastination and maintain their commitment to their tasks thanks to this accountability and social pressure, which increases productivity and decreases distractions. The platform increases productivity by taking use of the psychological benefit of working with someone else, even if they are not physically there.

3.4 A New Solution

The solution of 'PomoShare', the web platform that was developed throughout this research, could be considered an inclusive version of all the applications that were mentioned above. It is designed to enhance productivity and time management since it provides multiple features to help users. Task list management along with collaboration, are the main topics the application focuses on. Here is a description of the key pages in the web application:

- Pomodoro Room with Chat and Scoring:

This feature provides a dedicated space for users to practice the Pomodoro Technique, a psychologically approved time management method. Once a user selects the option of entering a room, an algorithm runs in the background that should match them to another available user of the same profession. This algorithm is further enhanced with finding the closest distance by using the Mean Squared Error for all available users, based on their score, their classified level of experience, their day's streak and their average focus time. In case of no user available, the system automatically creates a new room for the user which becomes available for other players.

The members of a room can set a timer for a specific work interval, typically 25 minutes, followed by a short break. After some number of sessions, members benefit from a longer break. The room includes a chat feature that allows users to communicate and encourage each other, for those who are in the same Pomodoro. Additionally, users can earn experience points based on their productivity, which serves as a motivation factor.

- Leaderboard:

The leaderboard feature displays the ratings of users based on their performance in the Pomodoro sessions. Each user is assigned a days streak factor, their average focus time and the score so that users can compare their productivity with others and track their progress over time. It is also able to sort the users based on one of these characteristics. It adds a competitive element to motivate users to improve their efficiency.

- Private Profile:

The private profile section provides users with a personalized space to view their progress and productivity metrics. It includes statistics such as their experience points, average session length, a plot for their weekly focus time progress, the percentage of tasks they have completed and the consecutive days they used the tool. Users can use this section to monitor their growth and identify areas for improvement.

Todo List:

The platform also offers a private to-do list feature which allows users to create and manage their tasks effectively. They can add, edit, and delete tasks as needed, set due dates and priorities, and mark tasks as completed. This feature ensures that users have a clear overview of their responsibilities and helps them stay organized.

Priority Matrix:

The priority matrix is a tool that helps users prioritize tasks based on their urgency and importance and is offered visually in the app. It categorizes tasks into four quadrants: Urgent and Important, Important but not Urgent, Urgent but not Important, and Neither Urgent nor Important. This visual representation allows users to set prioritize and assign their time and effort accordingly, ensuring that important tasks are not overlooked and urgent tasks are completed on time.

Calendar:

Another commonly used feature in the productivity applications is an online version of a calendar. For this reason, it was included so that this visual representation make's the user experience easier. Users can add events and view their daily, weekly, or monthly schedules. This helps users plan their time effectively, avoid conflicts, and stay on top of their commitments.

Overall, PomoShare combines elements of time management, productivity tracking, col-

laboration, and task organization. It can be considered an all-in-one platform to help enhance efficiency and achieve goals.

Chapter 4

Methodology

In this section, we should focus on the software processes that were followed for development, along with the fundamental principles in the topic [6]. First, the meaning behind software production process includes the set of actions that are necessary for this software system. It can be visualized by specified models or methods that represent arbitrarily these processes. The fundamental parts of any mentioned model can be summarized as:

1. Requirements Specification: planning the resources needed beforehand.
2. Software Design: designing the overview based on requirements.
3. Software Development: implementation of an executable system
4. Testing: checking if requirements and user's needs are met.
5. Deployment: making public.
6. Updating and Maintaining the Software: adapting the existing application to changes.

In practice, the general models of processes indicate the organization of the software development processes. Some examples of these general models are the Waterfall model, the Agile method, and the Component-Based software. It is also important to clarify that iterative models are processes that are based on stages life cycle.

4.1 Agile Software Development

Since the Agile approach [7] was chosen for the purposes of this web application, it should be investigated in depth. Agile refers to the set of principles that construct an iterative and collaborative approach to software development, which breaks down the process into small sprints of delivering software. It aims at collaboration and benefits from constant feedback making the system extremely adaptable and flexible with continuous improvements. Holding the target audience accountable to persistently improve the software, makes Agile a top choice for developers. In this way it is clearly embraced to adjust to evolving requirements and market conditions while ensuring that teams collaborate on effective development.

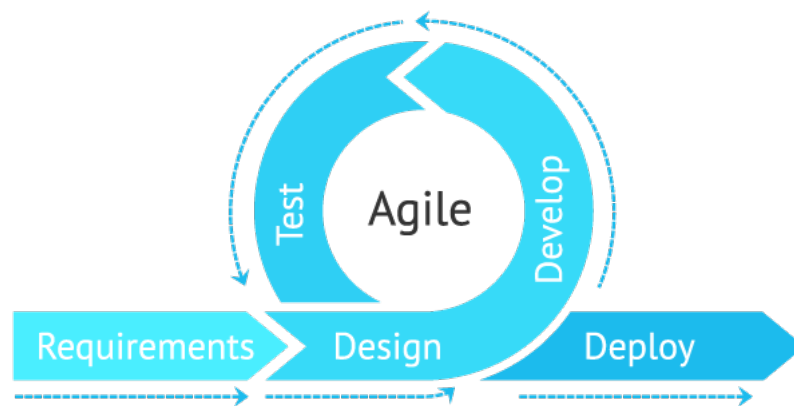


Figure 4.1: Agile Methodology

Compared to other common software process models, Agile seems to be the best option. Starting with the Waterfall model, which requires that each step is completed first, before the next one begins and therefore takes longer development cycles. In addition, customers have less accountability, and it happens after the initial requirements stage. Similar to Waterfall since it is a variation, the V-model has less flexibility and can have delayed feedback. This is due to the charging of a testing activity to each stage of the model. Another remarkable iterative method, this time with customer involvement and frequent feedback, is the Spiral Model. It can be considered a combination of the Waterfall model with prototyping and includes risk analysis and diminution. However, many times this task of risk evaluation may not be an easy task.

4.2 Requirements

The first step of the cycle concentrates on the requirements of the system, also known as the high-level description of the functionalities and the limitations of the system.

As a result of requirements specification, there is a clear distinction between the functional and the non-functional requirements. The former, refer to the user-specific features that the software must possess and what response it should produce in the event of an input. While, the latter highlights the quality limitations, such as the time complexity, the error handling, the scalability and maintainability, and other system characteristics. It is with the collaboration of the two types of requirements, that the needs of the users and stakeholders are fulfilled by the software's quality and performance.

For this reason, the system requirements for the PomoShare application are specified below in table form 4.2, 4.1:

Table 4.1: User Collection Schema

Non-Functional Requirements	
No.	Description
NFR01	The navigation should be easy
NFR02	All pages should have Startup style personality
NFR03	User interface should be user-friendly
NFR04	Components should run conservatively
NFR057	REST API must abide by architectural rules
NFR08	The server should communicate with the client
NFR09	The server should respond with correct status
NFR10	Errors should be handled
NFR11	The system should be scalable
NFR12	The interfaces should adjust to device screen
NFR013	The system should be able to handle many users

Table 4.2: User Collection Schema

Functional Requirements	
No.	Description
FR01	User should be able to register by inserting email, username, password, level of experience, profession
FR02	User should be able to Login with email and password
FR03	Authorized user should be able to logout
FR04	The system should show an error message in case of login/register fault
FR05	User must be able to see their profile card and analytics
FR06	User must be able to edit Level of Experience and Profession in the profile card
FR07	User must be able to access their private task list
FR08	User must be able to access their private Priority Matrix
FR09	User must be able to access their private Calendar
FR10	User must be able to add a new task
FR11	User must be able to label the new task by Urgence and Importance
FR12	User must be able to edit existing task
FR13	User must be able to delete a task and verify their action
FR14	User must be able to automatically create or be matched to an existing Pomodoro Room
FR15	Users in a room should share the same state of the room
FR16	Users in a room should be able to use the chat
FR17	Users in a room should be able to change the settings of the Pomodoro timer
FR18	Users in a room should be notified for any score addition or subtraction
FR19	Users in a room should be informed in the chat for any setting change another has submitted

4.3 Design

Agile encourages building a design based on the users' needs and preferences. Since the productivity application we are developing mainly aims at human growth and psychological support, the process of designing it must be concentrated on empathy. Constant adaptation is a significant principle of Agile design and feedback is a top priority through sprint reviews and demos. As users of the platform, we would appreciate a clear and minimal design with the essential additions that only contribute to a useful application without complex decision-making.

Designing the user interface requires following some fundamental UI/UX principles [8], after deciding on a global style. The style that is chosen for this platform is "Start-Up/Upbeat", which in summary is a minimalistic approach with rounded components and smooth content to make it less formal and user-friendly. The process of design thinking includes describing the problem and finding out the characteristics that should build the interface. The list that follows shows some of the principles that were embedded during the design process:

- Consistency
- Simplicity
- Feedback and Responsiveness
- Visual Hierarchy
- Accessibility
- Usability Testing
- Emotional design

Architectures

The process of designing software development can be simplified by selecting the architecture that will be followed. In this web application, the Client-Server architecture and the Model-View-Controller architecture coexist. In particular, the overall system is based on the client-server scope, while the data management and communication in the server aligns with the MVC pattern.

On the one hand, client-server divides the design process in two main components: the client and the server. It actually allows for a distinct separation of logic. The goal of this pattern is to create a server side where multiple clients may connect and servers that can handle concurrent requests from various clients. The way it works includes a client component that is responsible for user interaction and the logic behind the presentation. It is basically the part that users access through their web browser, mobile app or desktop. This side communicates with the server to request data and demonstrate it accordingly. The server part of the architecture handles the background logic and the database handling or storing. The server listens for requests and after some set of operations, sends back the results as response.

The MVC architecture 4.2 is more complex and requires deeper insight[9]. Once again, appears the concept of separating the concerns and dividing the logic into smaller parts. The parts in this case are the model, the view, and the controller:

1. Model: It refers to the data and query logic of the software. Data structures, database access, validation rules, and other operations related to the data are included.
2. View: The view includes the information that is returned to the user in various formats like JSON or HTML. It is responsible to communicate with the controller for actions related to user interactions.
3. Controller: This is the main control logic of the application and the intermediary between models and views. It handles user input and the response outputs.

This traditional pattern is commonly followed by a service layer and should be adapted in our methodology. It is considered a tool for better code organization and maintainability since it separates the complex operations from the controller and the model components. The operations do not necessarily relate to data manipulation since these services can be used by the controller for higher-level functions. Some examples that are implemented in the service layer for developing the application of this project are user authentication and data validation.

For this research, the MVC is utilized in the back-end logic, although it also can apply on the user interface. Each of the three components can be tested independently, making testing easier.

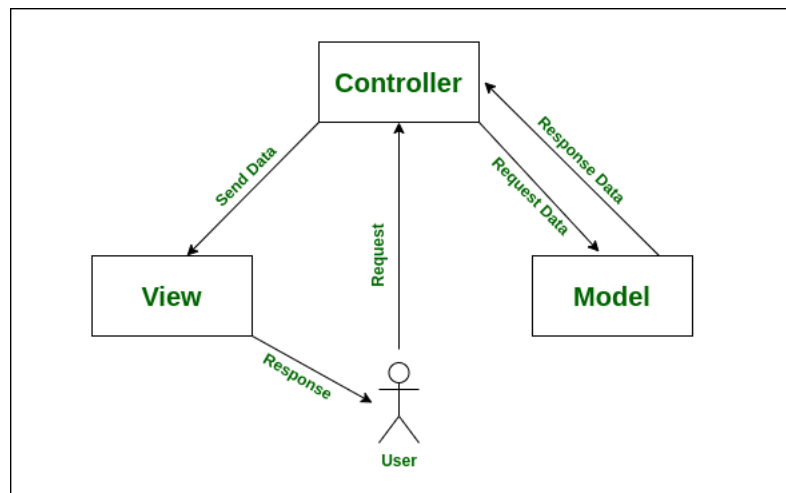


Figure 4.2: MVC Graphically

4.4 Implementation

The implementation part of the software process is extensively described in Chapter 5

4.5 Testing

In the testing phase of the Agile development there are certain challenges that should be considered. First of all, it focuses on quick iterations and frequent releases. This means that there can be time limitations for the testing which would extend furthermore the difficulty in testing processes. Considering that constantly updated requirements are not only accepted but actually embraced for the sake of adjustable applications, it is clear that tests may become unmanageable. It is why developers must prioritize test cases that are inclusive and make their strategies as expandable as possible to changes.

For the purpose of the application that was developed (PomoShare), a challenge emerged for balancing the useful in-depth testing and the constrained time to fulfill this task. It is also crucial to ensure test coverage, although with tight schedules in Agile, it is probably difficult to cover all the various scenarios. Again in this case, testers are responsible for good decision making on which cases to test by emphasizing on the most risky functionalities. These scenarios demand a considering amount of data. In other words, testers need time to come up with suitable test data which may be especially difficult for diverse and realistic use cases. At the same time implementing and maintaining test automation frameworks can be challenging. Creating automated tests and keeping them up to date with changing requirements and

system updates are of equivalent substance.

Particularly for testing the API and services of this project, a common tool for Agile testing was utilized, known as Postman [10]. Among other, this tool simplifies each step in the lifecycle of an API and emphasizes on collaboration thanks to the streamline of activities. During the Agile testing, several benefits emerge from using Postman[11]:

1. Sending queries, creating test scenarios, and validating API responses are all made possible with Postman. We can test various API endpoints and functions because it supports a variety of HTTP methods, request headers, and parameters. Every endpoint from the list in Chapter 5 was put to test using Postman and reviewed by the custom response in JSON form.
2. By defining test scripts, building collections, and executing them in a continuous integration environment, Postman may automate testing. Even though it was difficult to implement, it was occasionally used to automate project testing.
3. Collaboration between testers, developers, and other stakeholders is facilitated via Postman. Partners can share test collections, test results, and documentation in a collaborative development process—which is not the case here—making it simpler to discuss and agree on API behavior and expectations.
4. You may handle many testing settings using Postman, including development, staging, and production. The seamless configuration and switching of variables between environments enable effective testing with various setups. For example, some valid tokens that were generated once a login request was fulfilled, were used as global environmental variables to make the authentication process quicker and test the protected routes.
5. We can check the status of the execution, spot errors, and keep tabs on the performance of our APIs with the aid of the thorough test results and insights provided by Postman. The request status is displayed in a specific box, and the response data or errors are displayed clearly, making the debugging process even easier.

4.6 Deployment

For the purposes of the thesis and the final demonstration, the deployment of both the frontend and the backend is based on Render. This online tool is considered the best alternative to Heroku, the most used cloud platform which no longer provides a free version. Render offers a free individual plan for experimental projects on static sites, web services, etc. The REST API is deployed through a web service while the React app uses a static site service, both on Render.com [12]. The only steps that the process requires is to add the code to Github [13], frontend [14] and backend [15], and pass it to the cloud platform.

Chapter 5

Application Development

At this point in the thesis, the process of the application development should be analyzed in detail. In order to construct a web application, it is crucial that we examine the various architectures and technologies that we can choose from and select the most suitable for the case. Scalability and robustness are the most demanded factors in a full-stack application. In other words, handling complex data structures, building a user-friendly UI (user interface), along with a stable and secure communication between the two sides of the application, the client and the server, should compose the optimal selection.

This project is developed with the help one of the most popular stacks and frameworks in web development, the MERN Stack [16]. It consists of MongoDB, Express.js, React and Node.js. The main advantages of following this approach include the utilization of only one programming language, JavaScript, which is essential for creating dynamic web content. Therefore, the developing process is faster and the code becomes reusable across parts of the application. The NoSQL database (MongoDB) provides flexibility and a structure that aligns well with the frontend data format (JSON). As a result, many developers support it by contributing to the documentation and libraries and make it the possibly the best choice for web development.

By looking deeper into JavaScript, it is clear that web development is the primary concept where it can be applied. To be more specific, thanks to JavaScript, the HTML Document Object Model manipulation becomes an easy task within the user's web browser. However, its benefits expand into the server and other forms of application development making it a versatile option for a dynamic web page that do not require a refresh. In any browser, JavaScript uses triggering events, such as a button click, so that content is then handled by any fron-

tend library (eg. React) or framework (eg. Angular). In case of server-client communication, it supports asynchronous programming, which prevents delay in the browser for operations that take a long time to finish.

5.1 Asynchronous Programming

Asynchronous functions [17] should be understood before moving on to server-client development because they might get complicated in both frontend and backend services. Modern programming languages have `asynn` (short for asynchronous) methods, which enable concurrent task execution. Asynchrony and the concepts of concurrency and non-blocking operations are closely related.

Since many tasks, like sending HTTP requests or reading and writing files, take a long time to complete, the remaining actions would have to wait to begin, making the code extremely inefficient. When waiting for a certain process would result in unneeded delays, inefficient resource consumption is a frequent occurrence. When a function is called in conventional synchronous programming, it often freezes program execution until it is finished, and the program waits for the outcome before moving on to the next instruction.

Async functions are used to overcome this issue while maintaining program execution, which is where the idea of a "promise" is first presented. Once the async function is invoked, a `Promise`, a unique object that symbolizes the function's ultimate outcome, is created. Without waiting for that to complete, the execution can move on to other tasks.

When the async method has finished running simultaneously, it can resolve the promise. In other words, it gives the operation's outcome, which can subsequently be handled using a callback function or other methods. The promise can be "rejected" in the event that an error arises while the async function is being executed, enabling the software to gracefully address the situation.

To build responsive and effective apps, async functions are notably common in JavaScript, where the `'async'` and `'await'` keywords are used to implement them. Asynchronous functions are frequently combined with other features like event-driven programming or threading models, allowing programmers to construct code that does many operations concurrently.

It is also crucial to engage in the places that async code can be found in the MERN stack, since it is our main target in the project development and production. API calls, file

operations, database queries are only a few examples that needed special handling during development so that the user experience is not threatened by slow loading speeds.

In Node.js there are plenty of async activities especially when working on databases and making API calls. In fact, all Mongoose methods are async. The `async/await` syntax is a common way to handle them which is a progressed version of Promises. The Promise can use chain operations with `.then` and `.catch` to complete the same task that `async/await` fulfills more simply, almost as a synchronous function would do. For instance, the MongoDB driver in Node.js can make asynchronous queries, while leaving out the chance of a 'callback hell' that may occur in Promises.

We can also ensure non-blocking middleware functions in Express.js by using `async/await` or promises. Data validation or authentication may often be time consuming and synchronous HTTP requests would otherwise cause a problem, along with many more middleware functions.

Asynchronous functions are also covered in frontend technologies, as in React. It is obvious that any data that needs to be fetched, updated or deleted, also takes a long time to complete efficiently, via Axios and other APIs. However, there is hidden async functionality in the React state and side effects. In this case, React offers hooks like `useState` or `useEffect`. This is also the reason why in React, updating the state will not showcase the result immediately, but rather needs to be wrapped in a controller. Again, the `async/await` syntax is preferred.

As mentioned previously, the NoSQL database, MongoDB, offers asynchronous techniques for database interaction. In this project, I have included an Object Data Modelling (ODM) library for Node.js called Mongoose and embedded the `async/await` syntax for the queries.

5.2 Back-End Development

As mentioned above, the MERN stack application that was used for this project, uses JavaScript for the backend technology. Node.js [18] is the runtime environment that MERN stack uses for server-side logic execution. It is a free tool that is compatible with various platforms and therefore delivers flexible hosting. Due to its non-blocking Input/Output model and single-threaded architecture, Nodejs can handle many concurrent operations while maintaining high performance.

As the first letter of the 'MERN' abbreviation suggests, MongoDB [19] is employed as the database component for this project. It is a "NoSQL" (non-relational) database system which means that the stored data are more flexible and do not follow the structure of a typical table. MongoDB supports CRUD operations in order to manage the various collections and, by extension, the document models that are stored in JSON form, that represent the entities of the application (user, room, task, leaderboard). Additionally, it is able to handle large volumes of data and make it the perfect choice, thanks to the following mechanisms:

- Horizontal Scalability: Data are distributed across more than one server/cluster, enabling the parallel and balanced actions on big datasets.
- Replication: Data are automatically copied in multiple clusters in order to ensure fault tolerance and prevent data loss
- Indexing: Data are enumerated making it easy to retrieve a specific index as well as action slike filtering, sorting, etc.
- Aggregation Framework: Complex dataset manipulation with various powerful queries used for filtering, grouping, joining and transforming data
- Flexible Schema: Allows dynamic changes in the MongoDB schema without downtime, making it ideal for varying structures or alternatively destructive changes.
- Compression and Storage Optimization: Automatic features that minimize the usage of the disk space and make it as efficient as possible

The application follows a RESTful API design pattern that is constructed with Node.js, with the help of the Express.js [20]. This framework is responsible for handling HTTP requests and responses. In particular, it can define routes for the various HTTP methods. The only methods necessary for this case are GET, POST, PUT and DELETE, since REST APIs perform a set of actions known as CRUD (Create, Read, Update, Delete). Express.js was also fundamental for creating middleware and integrating with existent middleware for ac-

tions like authenticating users or even error handling.

The term 'middleware' refers to the functions that are used during the request-response cycle to act as a communication provider between client and server. For the purposes of the thesis, considering that all actions across the platform should be protected and personalized, a user login and authentication middleware is essential. In this way, any invalid request will prohibit access. To make it even simpler throughout development and production, the error middleware is included to catch and log errors, leading to an easier debugging process.

User authentication and security across the platform are established through JSON Web Tokens [21]. This token is commonly used for authentication and authorization issues [22]. A brief overview of how it works would start with the generation of the JWT once a user is signed in, and it basically includes user's data encrypted by a private value that the server requires. It is then stored in the client's side local storage as a response body. For a user to have access to any page that needs authentication, the client is responsible for the verification of the locally saved JWT using the same private key that signed the initial token. In case of an expired token, since many carry the information of an expiration time, the client needs a refresh token that will retrieve the token and reauthenticate the user in the server.

Here is an part of code for the Authentication middleware, to verify the JWT token using try-catch for error handling:

```
const protect = asyncHandler(async (req, res, next) => {
  let token;

  try {
    // Get token from headers
    token = authorization.split(" ")[1];

    // verify token
    const decoded = jwt.verify(token,
      process.env.ACCESS_TOKEN_SECRET);

    // get user from model
    req.user = await User.findById(decoded.id).select("-password");

    next();
  }
```

```
    } catch (error) {  
      res.status(401);  
      throw new Error(error);  
    }  
  });
```

To work with JSON Web Tokens was a useful decision and a fully analyzed problem that required a list of pros and cons. First of all, JWTs contain the necessary information about user identity within the token itself. Therefore, there is less need for server-side storage, which would cost a lot more. Security was also a crucial factor in choosing to work with these tokens, since they establish authenticity and only authorized actions. While other authentication mechanisms use the server for configuration, JWT is decentralized and breaks down the tasks into different services that communicate. However, the security remains a fragile issue and requires considerate usage of the token (eg. where the private key is stored), which may make it vulnerable to attacks.

5.2.1 Additional Libraries

Some common libraries were extremely helpful throughout the process of backend development, by providing special functionalities, each one focusing on a targeted area. Following is a list of the libraries that were used in the case of this project, aiming at an organized workflow, setting real-time communication, database handling and many more benefits.

- cors : The cors library [23] provides CORS (Cross-Origin Resource Sharing), a mechanism that allows access to resources on different domains other than its web page. This enables access policies and handling cross-origin HTTP requests.

- mongoose: Mongoose [24] is an ODM library for MongoDB in Node.js that interacts with MongoDB and allows querying on a schema level for modeling application data. It can create a powerful API while simplifying data validation.

- jsonwebtoken: It is a library used for working with JSON Web Tokens (JWT) in Node.js applications.

- express-async-handler: Express-async-handler is a library that is responsible for error handling in the asynchronous calls to the database. It is used as a simpler alternative to the try-catch method that is commonly used in async/await functions, and returns the proper

response asynchronously. In our case, it collaborates with the error middleware that was previously mentioned.

- moment: The Moment.js [25] library is ideal for date and time manipulation. It handles various formats of dates as well as different time zones, so that functions that include dates are more reachable.

- socketio: Socket.IO [26] is a library that offers real-time, bidirectional communication between clients and servers that is usually triggered by events. It can be found in chat rooms or multi-user games and is extensively used in the project that will be presented in the next sections.

Socket.IO is based on the WebSocket communication protocol [27]. This protocol, unlike simple request-response connections, enables a persistent connection that transmits data in both directions without the need for repeated requests from the client. The communication can actually be implemented in both client and server side at any time, leaving out the delays of receiving updates and even the cost of making a new connection. Once a connection is established, the protocol uses a handshake mechanism with HTTP 101 success status and proceeds to a messaging system.

The server side is responsible for handling any event by listening using "socket.on()" for a message from "socket.emit()" of the same label. The usage of this library can be briefly demonstrated in the following parts of code from the backend development. In this case the server handles the case of a "send-message" event, gets access to the user that submitted the message and broadcasts it to the rest of the users in the room by default:

```
io.on("connection", (socket) => {  
  socket.on("send-message", (message, room) => {  
    const name = socket.handshake.query.username;  
    socket.to(room).emit("receive-message", message, name);  
    //broadcast is assumed  
  });  
})
```

The example above is used for the Chat between the users, and the case of the emitted message is handled in the frontend. In the application, several messages were essential, including the case of a user connecting or disconnecting to the Pomodoro room, the case of settings change of the room, as well as the play-pause button hit, to synchronize users.

By working with these backend development options, the result full stack application will have solid foundation. All in all, a grounded combination of API design, flexible database, middleware and security integration, can now lead the way to an equivalent user interface.

5.3 Front-End Development

The frontend part of the application is developed benefitting from the Vite.js [28] build tool. It is a recently released tool mainly used in order to take advantage of modern browser features and especially the native ES modules. Vite.js ensures fast startup thanks to the quick development server and the hot module replacement, so the server communicates by not affecting the state of the app and with instant and correct updates. Bundling, the concept of translating a high-level language to a lower-level code, is in this case a lot more efficient. Compared to the case of webpack or parcel, Vite.js requires minimal configuration setup. In other words, it provides configuration defaults for common cases, excluding the potential of a long time to get started with development. In addition, Vite.js becomes a great choice for developers because of its built-in end plug-in features, which only aim at a better workflow. Thus, it natively supports JSX and Typescript, the syntax used in React components.

The MERN stack includes React for the client side of the application. React is a JavaScript library[29] useful for building user interfaces with JavaScript with the help of components. The component-based architecture that React is based on, enables modularity for parts of a complex user interface and its logic. The sense of a Virtual DOM is presented in this library: a copy of the actual DOM that acts as a preliminary stage for only updating the essential parts of the UI. State management is the only case where the actual DOM is triggered, therefore enhancing the overall performance of the app. Additionally, the part of programming that requires HTML tags, the fundamental web page tool, is approached with the extension of JSX (JavaScript XML). In other words, the components return a defined structure of their appearance with the help of plain JavaScript. In React, data flow is unidirectional, which means that data changes follow a specific path in the application. This makes debugging doable and clearly states how changes are passed down through the components. Lastly, components can be reused across different parts of an application or even in other projects.

By examining the various libraries and frameworks when manipulating the Document Object Model with JavaScript, it is obvious that each of those has both strengths and weaknesses. As the results of the research[30] suggest, React is a library that contributed to low DOM manipulation overhead and efficient updates by using its Virtual DOM. On the other hand, Angular still performed accordingly, but slightly slower compared to React, due to the framework's change detection mechanism. Another commonly used framework that was put to the test is Vue.js, whose lightweight and optimized rendering method made it a challenging competitor. In summary, React stood out in terms of performance and is therefore considered a selection with great characteristics.[31]

Another very important part of React is the concept of state. The state includes the current internal data of a component that may change anytime. Managing the state is another crucial topic which can take up many pages to explain, but is sufficiently analyzed throughout this report and especially in terms of the Context API. However, the fundamental tool for managing state in the context of a functional component is the `useState()` hook. In case of side effects or component updates, the hook of `useEffect()` is extremely helpful to handle the lifecycle of a component, previously handled by outdated methods. The lifecycle of a component consists of mounting, updating and unmounting.[32]

It is a notable fact that setting the state is an asynchronous function, which means that the value may not be immediately updated. The solution to this problem is to wrap setting the state in a `useEffect()` function to make sure that it happens when the variable is updated. In addition, the `setState` function that occurs from the `useState()` hook should benefit from the function version which updates current state based on previous state and with this re-render, the variable actually updates.

The way that components communicate is quite straightforward. Props (properties) are passed down from a parent component to its child components and are carrying the information or state variables that are necessary for configuration in the children. Otherwise, React Context can be enabled for the purpose of global state management and sharing data through the component tree. This is especially useful when passing down too many props since that may perplex the code and lead to serious mistakes. It is also currently used as an alternative to the Redux toolkit, which has a similar role but many developers found it too complicated to maintain.

The application includes a countdown timer for the Pomodoro Technique, which can be

established with `setInterval()`. This React function can be used to execute a callback function at a specified interval. However, it's important to use `setInterval` with caution in React components to avoid memory leaks and unexpected behavior. By wrapping it inside a `useEffect` hook, it is rerendered every second (1000 milliseconds) with the appropriate dependencies and attributes. Ultimately, the interval is cleared when the component unmounts by returning a cleanup function from the effect in order to prevent memory leak

```
useEffect(() => {
  const intervalId = setInterval(() => {
    setCount((time) => time - 1);
  }, 1000);

  return () => {
    clearInterval(intervalId);
  };
}, []);
```

The process of React Context API integration should be properly examined, since it is widely used in the final application development of the thesis research.

1. The first step is to define a new context using React's `createContext()` function. This returns an object of two properties, one will then be used to wrap the JSX code of the components that communicate (Provider) and the other will be used to access state inside those components (Consumer).
2. The Provider takes as input or returns as output all the essential state manipulation functions and state variables as props and, as the naming suggests, provides to the child components.
3. Accessing the Context is next on the list, and is essential for components that need to consume the state. Another React hook is successfully used in this case, `useContext()`, for fetching any useful data from a context and perform operations on them.
4. Ultimately, in order to update the state in the context, an appropriate function can be inputted as a value prop in the Provider, or the top-level component should extract state update functions.

5.3.1 Additional Libraries

React offers a wide range of libraries that simplify the developer's responsibilities. The ones that were used in the project, among others, are listed below with a brief explanation:

- Axios [33] is a popular JavaScript library and the best solution for React to make HTTP requests from the browser. Fetching data, updating datasets, deleting from datasets becomes a lot easier due to a simple API for asynchronous calls to the server, and responses can be handled. In case of a bad server response, the client is able to serve the error messages, making the user experience smoother. The JSON data parsing that Axios provides makes the MERN stack more accessible when handling data during development or production.

- For the most complex JSX components, Material-UI was used [34]. It is a React UI component library that is commonly used in respect to the Material Design guidelines. Many responsive or visually appealing user interfaces are very complex to implement in plain HTML and CSS, like the SVGs, so Material-UI is a time-saving choice for built-in forms, dialogs, icons, etc.

- React Router DOM [35] was imported for the purpose of navigation through the pages of the application. It is a routing library for React applications that also handles URLs by rendering the matching current component. It additionally offers routing history and nested or conditional routing for protected URLs in collaboration with the authentication on the client side.

- As will be presented in the following sections, an analytics graph is created with Chart.js, a library specialized in creating interactive charts. There is enough space for exploration and editing in this library, from the various chart types (line, bar, pie, etc.) up to tooltips, colors, animations, which is ideal for data tracking.

- Web designers can build dynamic, editable calendars in their apps using the FullCalendar JavaScript package [36]. It includes functions like event display, event dragging and resizing, and date navigation, and it offers many calendar layouts, including month, week, day, and agenda. FullCalendar offers a wide range of customization possibilities and is compatible with several backend frameworks and data sources.

- A JavaScript package called Day.js offers tools for working with dates and times. It has a smaller footprint but a similar API to the Moment.js package. Day.js makes it possible to parse, format, manipulate, and display dates and times in various ways. It is intended to be effective and provides answers for tasks like figuring out durations, changing time zones, and

making unique date displays in JavaScript programs.

5.4 Implementation

5.4.1 REST API Documentation

In order to retrieve and analyze data for the web app, a REST API was built as mentioned above. The API can be retrieved through certain endpoints that will be listed in this subsection.

The base URL for the REST API is in the form "https://api.example.com/", locally during development it is in the localhost.

Some routes require authentication. These routes are protected by authentication middleware, and therefore require an API key in the header: 'Authorization: Bearer [API_KEY]'

Here is the list of endpoints:

- Endpoint: '/api/users/'

Allowed methods: GET, POST

GET: Returns a list of usernames for the active users only.

POST: Registers a new user.

- Endpoint: '/api/users/token'

Allowed methods: POST

POST: Uses the refresh access token to regain access.

- Endpoint: '/api/users/:id/confirm'

Allowed methods: POST

POST: Updates user 'confirm' to true once email is confirmed.

- Endpoint: '/api/users/:email'

Allowed methods: POST

POST: Sends email for resetting password.

- Endpoint: '/api/users/new_password/:id/:token'

Allowed methods: POST

POST: Resets the user password.

- Endpoint: `"/api/users/login"`

Allowed methods: POST

POST: Authenticates user, generates secret access and refresh tokens.

- Endpoint: `"/api/users/:id"`

Allowed methods: PUT

PUT: Updates user information by id.

- Endpoint: `"/api/users/:id/score"`

Allowed methods: PUT

PUT: Safely increments the score in the Leaderboard by id.

- Endpoint: `"/api/users/logout"`

Allowed methods: DELETE

DELETE: Deletes Refresh Access Token.

- Endpoint: `"/api/tasks/"`

Allowed methods: GET, POST

GET: Gets all user's tasks. Requires authentication.

POST: Create new task. Requires authentication.

- Endpoint: `"/api/tasks/:id"`

Allowed methods: PUT, DELETE

PUT: Updates task information by id. Requires authentication.

DELETE: Deletes task by id. Requires authentication.

- Endpoint: `"/api/rooms/"`

Allowed methods: GET, POST

GET: Returns room data. Requires authentication.

POST: Finds the best available room and inserts the current user. Requires authentication.

- Endpoint: `"/api/rooms/:id"`

Allowed methods: PUT, DELETE

PUT: Updates room information by id. Requires authentication.

DELETE: Deletes room by id. Requires authentication.

- Endpoint: `"/api/leaderboard/"`

Allowed methods: GET

GET: Returns the whole leaderboard list sorted by the user's score.

5.4.2 MongoDB Collections

USER

The User model 5.1 has a unique ID and must include the fundamental attributes that will be submitted once registered, the username, the email, the password, the occupation and the level of experience. The password is encrypted for security reasons, and is never returned in any HTTP request that needs the user in JSON.

Table 5.1: User Collection Schema

Users Collection		
Attributes	Type	Description
<u>id</u>	String	Unique user id
username	String	Unique Username
email	String	Unique Email address
password	String	Hashed Password
isActive	Boolean	True if user is activec
occupation	String	User's Field of Interest
experienceLevel	Number	User's Education Level
focusTime	Array	Weekly Focus Times
lastSession	Date	Previous day of Sessions

LEADERBOARD

A separate collection is essential for all the focus-based characteristics of each user, as shown in the table 5.2. The model includes a reference to the user so that rendering the according query is easier.

TASKS

The Tasks collection consists of multiple entries for the tasks of users. Each task is identified by a unique ID and the creator, referencing a User object. The description, the label ("Do Now", "Decide Later", "Delegate", or "Delete"), the due date and the state of the task (done or not), are included.

Table 5.2: Leaderboard Collection Schema

Leaderboard Collection		
Attributes	Type	Description
<u>id</u>	String	Unique Leaderboard Entry id
user	ObjectID	Reference to User Object
isActive	Boolean	User is Active Now
daysStreak	Number	Count days in a row User was active
score	Number	User XP points
avgFocus	Number	Average User's Focus Time

Table 5.3: Tasks Collection Schema

Tasks Collection		
Attributes	Type	Description
<u>id</u>	String	Unique Task id
text	String	Task Description
label	String	Type of Task
user	ObjectID	Reference to User
isDone	Boolean	Task is Finished
dueDate	Date	Due Date of Task

ROOMS

The Room schema is essential for keeping the available rooms in track, and distinguished based on IDs. It stores the information for the room like the members, the duration of each session, etc.

Table 5.4: Rooms Collection Schema

Rooms Collection		
Attributes	Type	Description
<u>id</u>	String	Unique Room id
member1	ObjectID	Reference to first user object
member2	ObjectID	Reference to second user object
isPlaying	Boolean	If Room Timer is Running
autoChange	Boolean	Auto Start next Session
focusMins	Number	Focus Mode Duration
breakMins	Number	Break Mode Duration
longBreakMins	Number	Long Break Mode Duration
longBreakInterval	Number	When Long Break Mode Appears

5.4.3 Application Overview

The first view of the application is the landing page, also known as home page. This page concentrates on giving the motive for user to sign up or log in and get started with using the various tools of the platform. By scrolling down, they get information on the idea behind these special tools for productivity, along with their usage hands-on. The scoring system is explained giving the essential information to the possible audience. Design-wise, the buttons that are brightly colored according to the styling, are known as the Call-To-Action buttons, and aim to encourage smooth navigation.

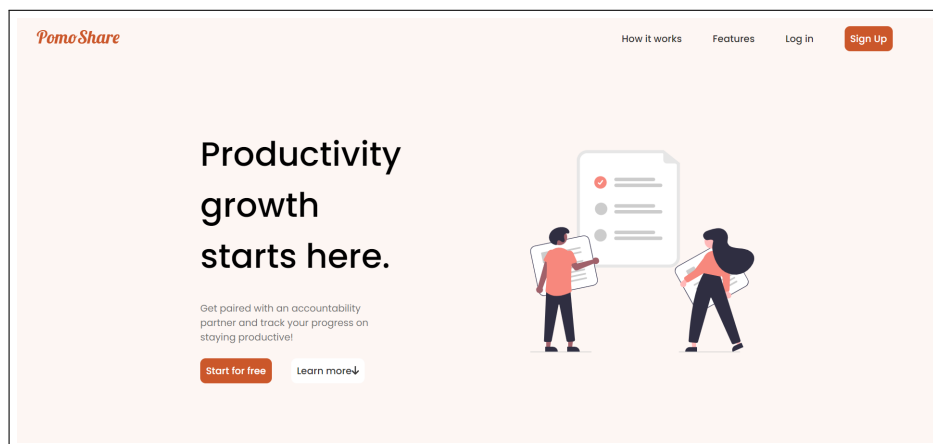


Figure 5.1: Home Page

Similarly to the login page, in the register page 5.2 all fields are necessary to complete registration and create a new user. Both in log in and sign up pages, there are navigation abilities.

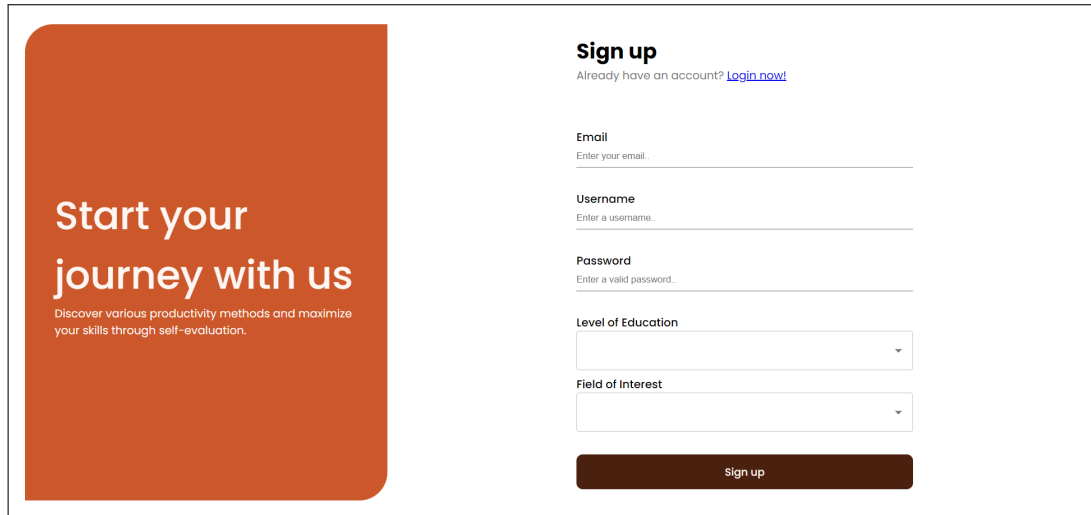


Figure 5.2: Register Page

For registered users, logging in can be easily completed by entering to the login page 5.3 from the home page or using the hard-coded URL path. Their email and password are required and in case of invalid user data, a message error appears to let them know.

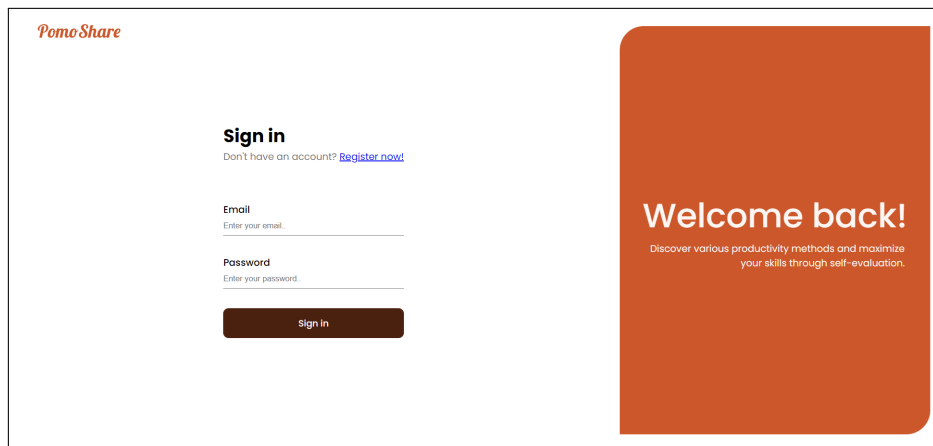


Figure 5.3: Login Page

In case a registered user forgets their password, they are able to retrieve their account via email. After submitting a valid email 5.4, a link is sent as seen in figure 5.5 with a unique token specifically generated for the user to create a new password. The token expires in 1 hour, so if the user enters the URL after that time limit, an error will pop up letting them know the issue.

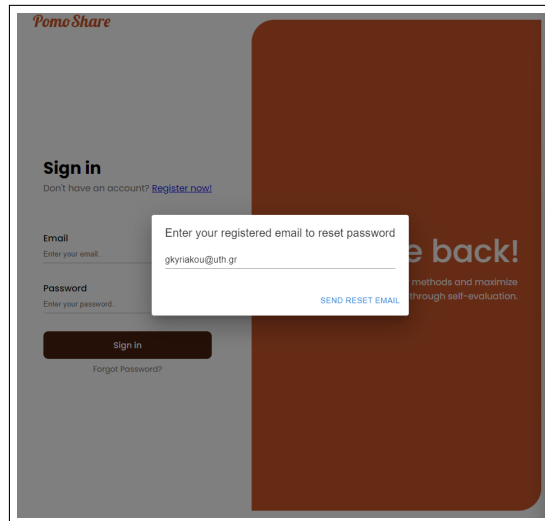


Figure 5.4: Send Email to Reset Password

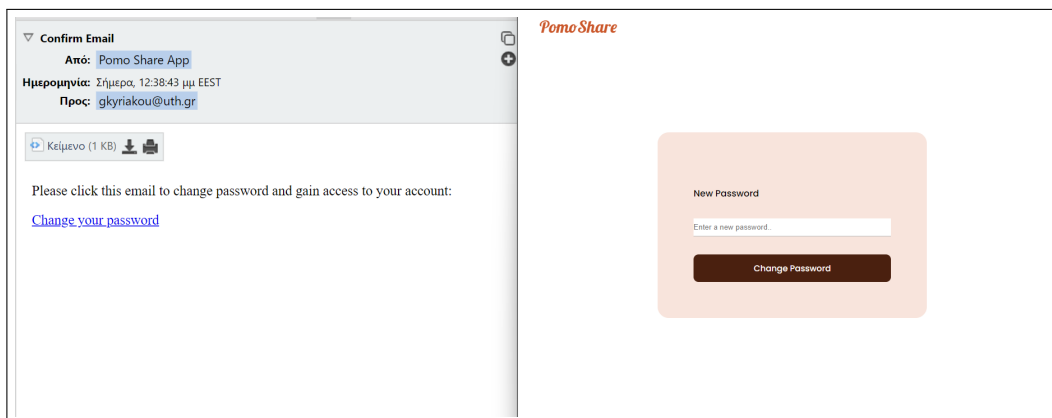


Figure 5.5: Link to Updating Password

A similar email is sent to the user once they register to the platform 5.6. In order for them to log in afterwards it is necessary that they click on the link sent to them via email. By clicking on this link the user is redirected to the login page after the user email is automatically confirmed. With this link they confirm that the email exists and therefore can ensure security for the application by permitting access to fake accounts while also making it certain for the user to retrieve their password via the confirmed email if that need occurs.

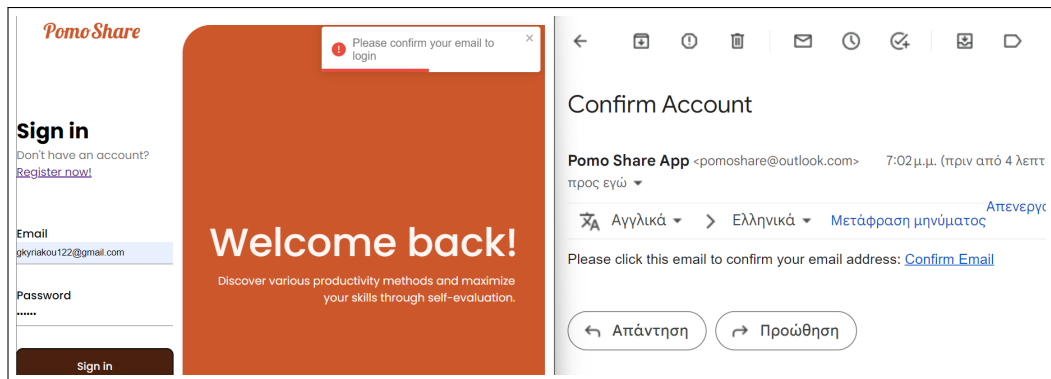


Figure 5.6: Link to Confirm Valid Email

A profile page 5.7 is the initial route for authenticated users, where the user information is displayed along with analytics, a plot for the average weekly focus time, the consecutive days, the completed tasks and the score. As seen in figure 5.8 the only editable part is the occupation and the level of experience of the user which once submitted, is immediately updated in the server side of the object.

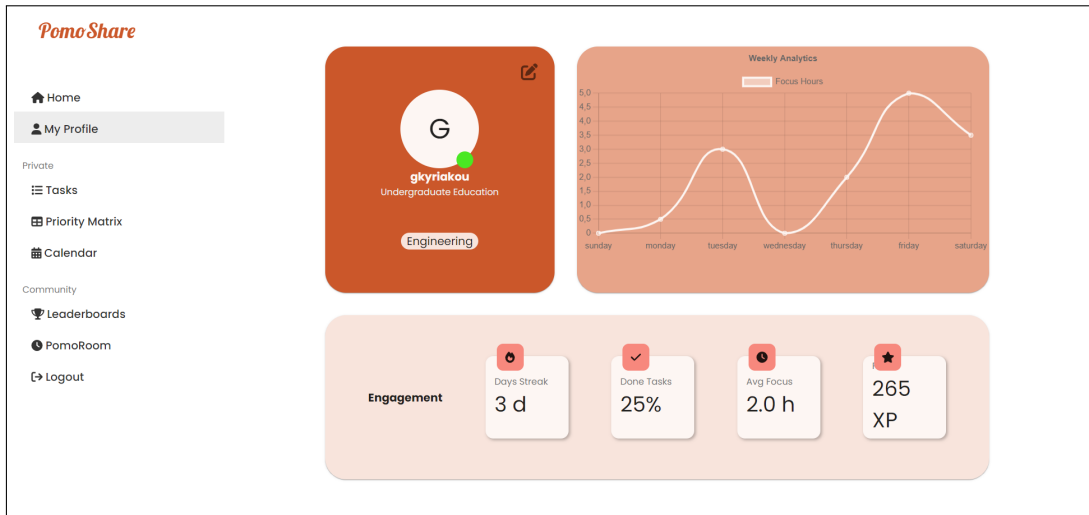


Figure 5.7: Profile Page



Figure 5.8: Profile Edit Page

In the following figures, the Pomodoro Room 5.9 is displayed. To be more specific, in the case of a user trying to enter a room, but there is no room available or no perfect match, they are given the ability to start their sessions. The first member therefore can make the changes they want on the timer settings or chat 5.10, and can wait for their accountability partner, without losing the chance of earning points. Once a focus session is completed, 10 experience points are assigned to the user, but if they pause during this time, 5 XPs are subtracted. After a set of intervals, which is also adaptable from the settings, the user automatically earns a bonus of 40 points for completing consecutive focus sessions, thus rewarding them for good performance.

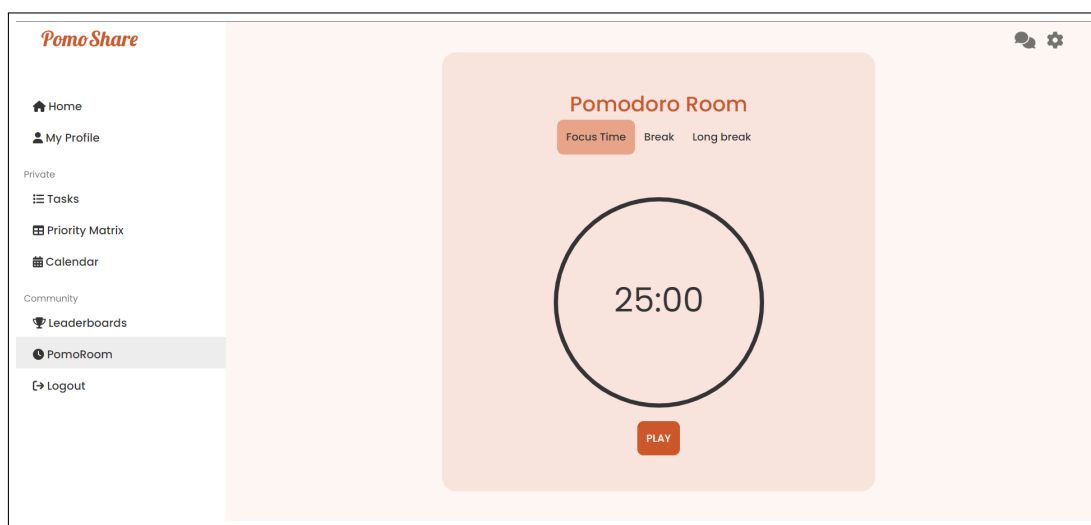


Figure 5.9: Pomodoro Room Page

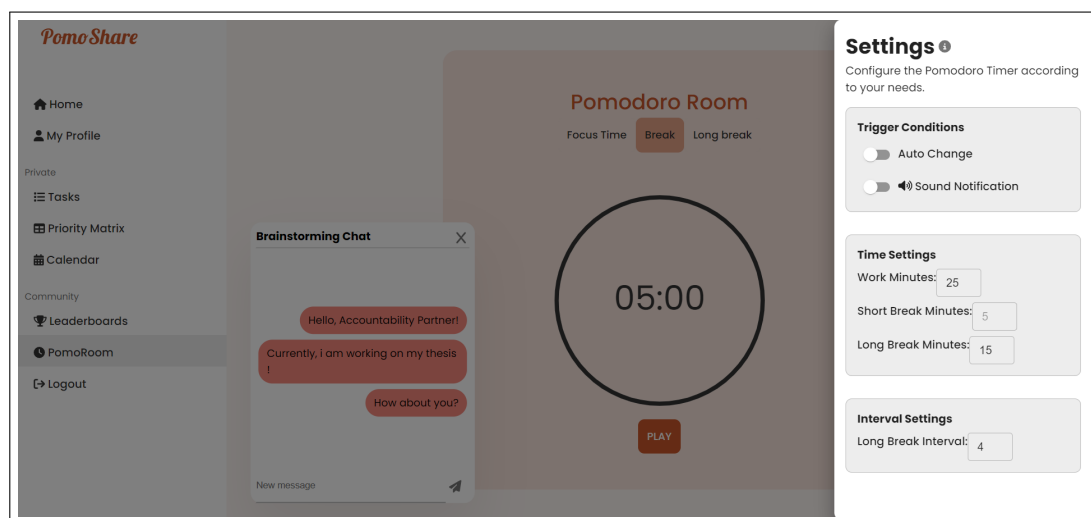


Figure 5.10: Pomodoro Features Page

In case a pair is created and the room is full, with two members, the first member and creator gets notified accordingly thanks to web sockets. The sockets are not only useful for the communication between the two members in the chat, but also in the case of a settings change. Then, the other member gets notified on the change, by broadcasting any new submission, except for the sound notification which is a personal preference. The timer and its state (paused/playing) is also synchronized for the two members, but the points are subtracted only for the user that hit the pause button 5.11.

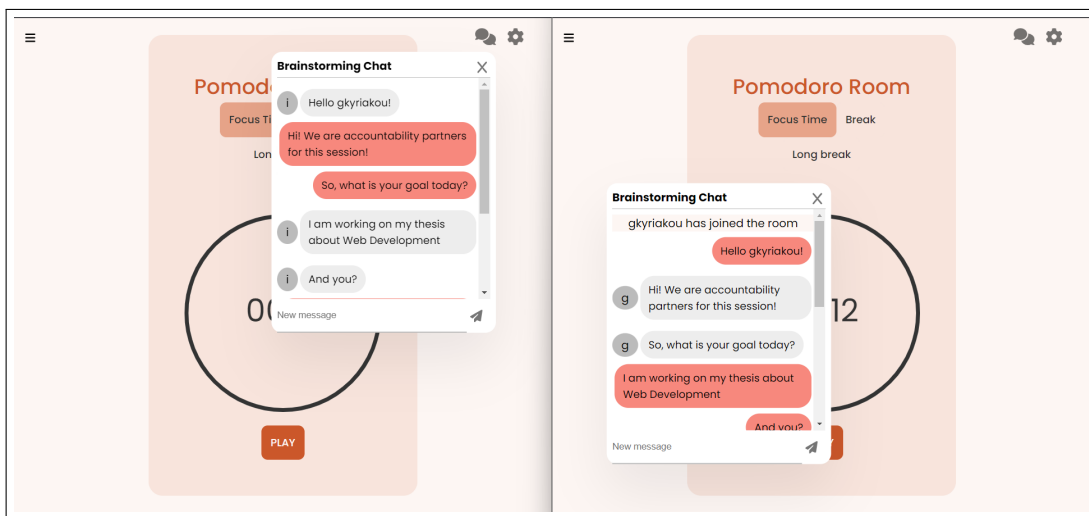


Figure 5.11: Pomodoro Chat, Different Members Left and Right

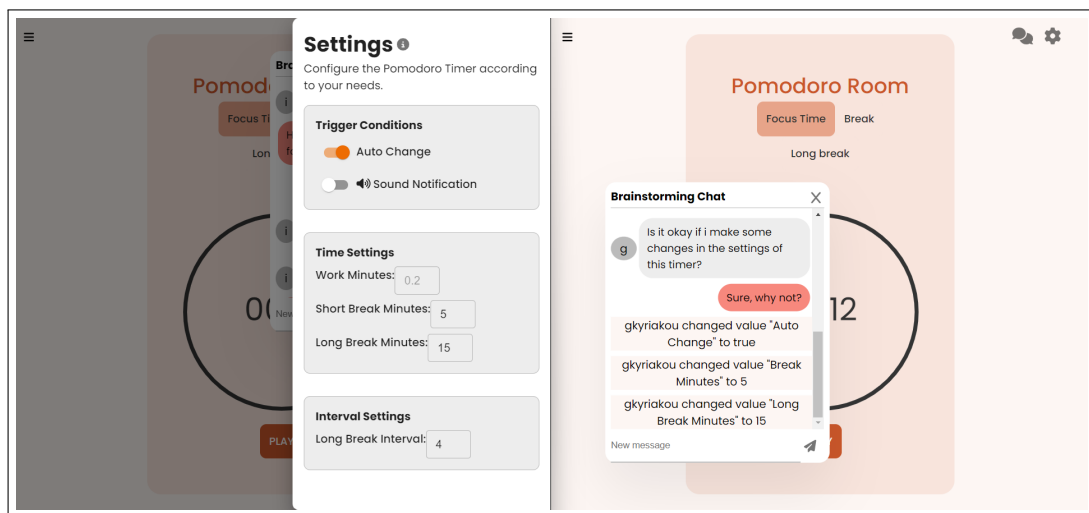
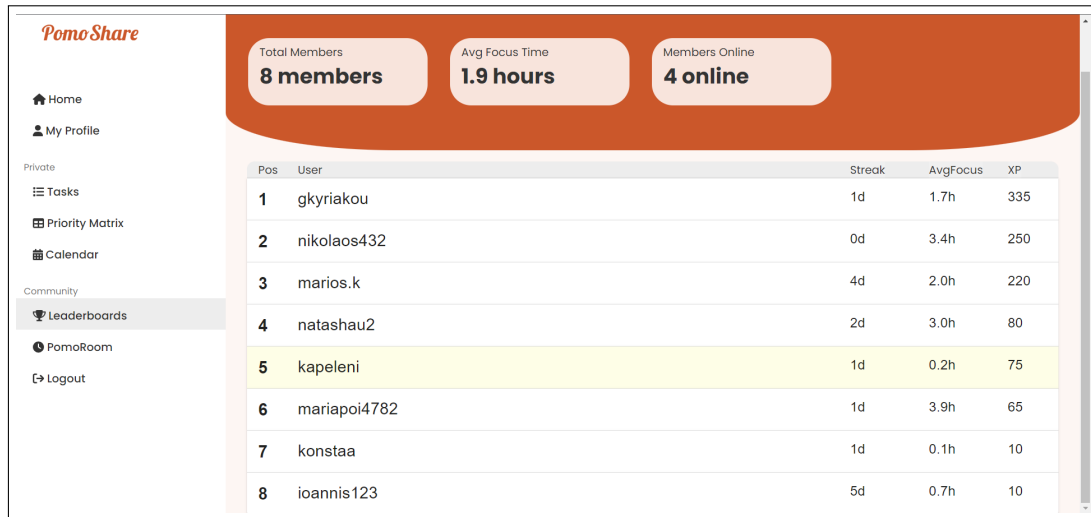


Figure 5.12: Pomodoro Settings Change is Broadcasted

In the Leaderboard page, the user can get a boost of motivation by given the ability to see the progress of all users. At the top of the page, a summary of all members appears and underneath, lies the ratings list. This list is by default sorted by the score of users and can be sorted based on other filters by clicking on a column label ("Streak", "AvgFocus" or "XP") as seen in 5.13. Also, the user can easily find their rating by looking for the yellow entry.



Pos	User	Streak	AvgFocus	XP
1	gkyriakou	1d	1.7h	335
2	nikolaos432	0d	3.4h	250
3	marios.k	4d	2.0h	220
4	natashau2	2d	3.0h	80
5	kapeleni	1d	0.2h	75
6	mariapoi4782	1d	3.9h	65
7	konstaa	1d	0.1h	10
8	ioannis123	5d	0.7h	10

Figure 5.13: Leaderboard Page

The task list page consists of the users submitted goals that are stored in the database. The layout as seen in 5.14 consists of four sections, based on the urgency and importance of the tasks. The categorization/labeling happens in the server side once a new task is submitted 5.15 where the user selects or unselects the according checkboxes. The form for the new task also contains the description of the task and the due date in a small calendar popup form. These considerations aim at a user-friendly navigation without much complexity, since managing tasks can be exhausting.

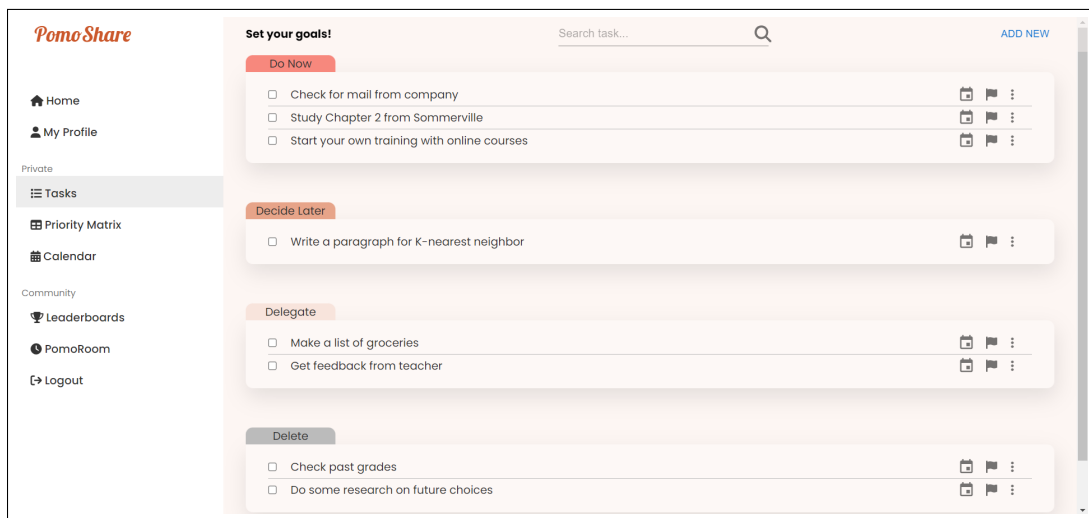


Figure 5.14: Private Tasks Page

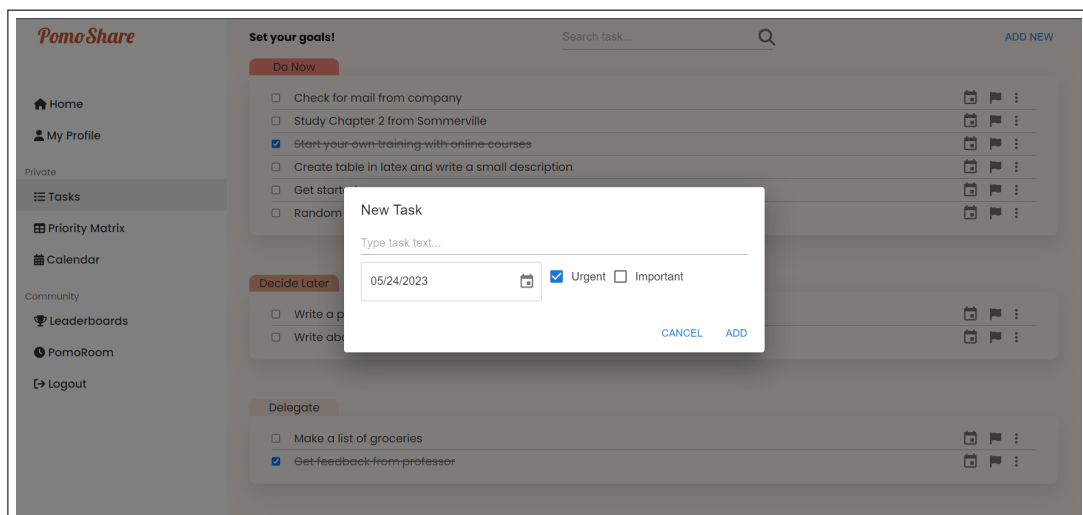


Figure 5.15: Add New Task

Each task can be edited in many ways^{5.16}. The first option is to change whether or not the task is completed by checking the box on the left of the text part. A popup calendar also offers the ability to change a due date immediately, while the three dots trigger options for editing the text or completely deleting the task. Each of these changes is automatically saved in the backend, to make the user experience even simpler.

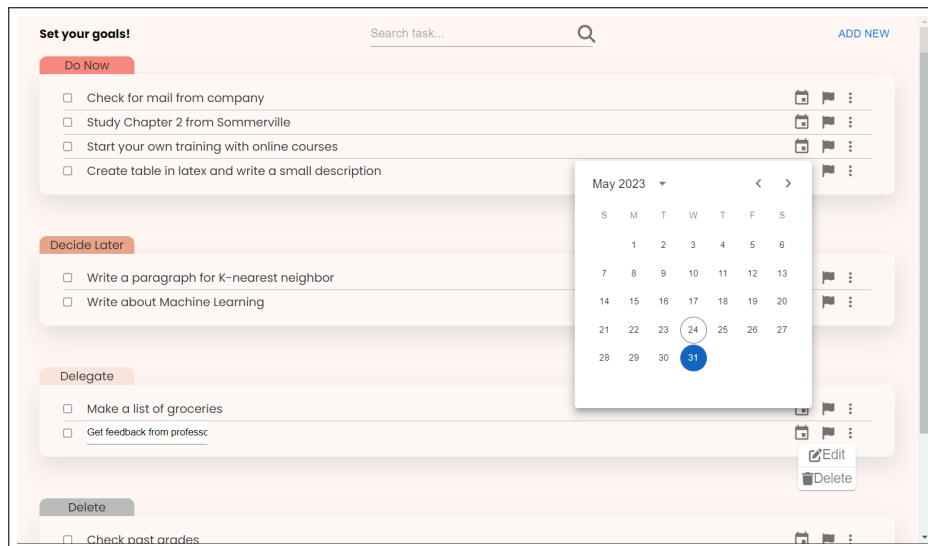


Figure 5.16: Tasks Edit Feature

To avoid confusion if the tasks are too many, the ability of searching a task is provided^{5.17}. By typing a word or letter, the task list is responsively filtered to only show the tasks that include it.

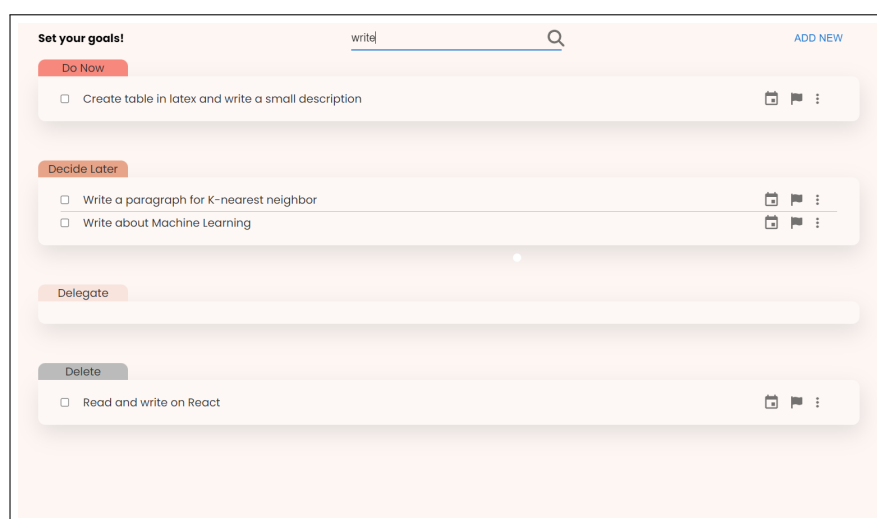


Figure 5.17: Tasks Search Feature

As for the psychological scope, the original Priority Matrix is visualized 5.18, with the four quadrants with similar features as the simple task list. A variety of visual options can improve the user interface and fit anyone's preferences.

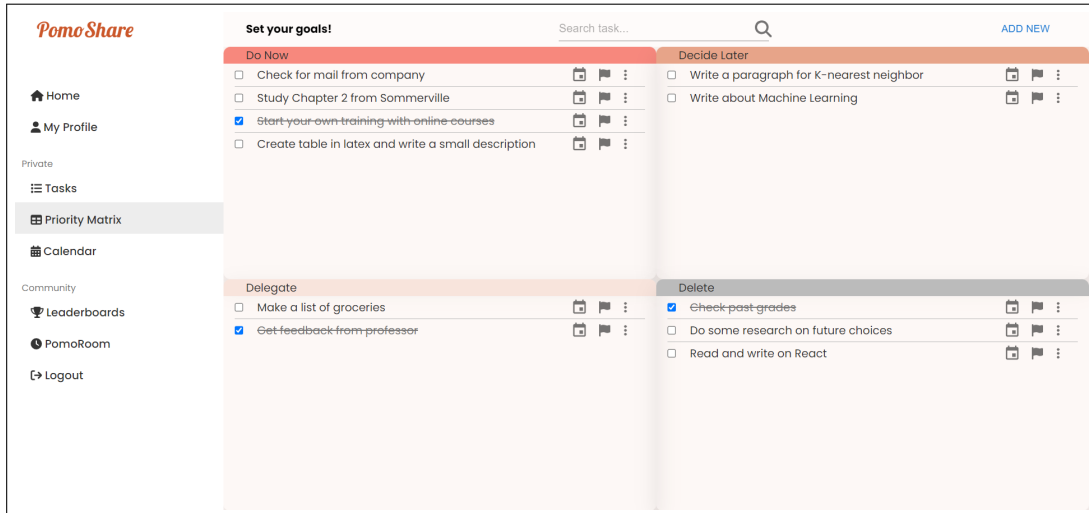


Figure 5.18: Priority Matrix Page

The last private page in the app available to the users is the Calendar page 5.19. It is synchronized to all the tasks that may be included in the other task management pages, and uses them for organizing the monthly and weekly goals. By clicking on a specific date, the form for adding a task appears in a dialog section 5.20, while by clicking on an existing task, the user can easily delete the task, after confirmation of the action.

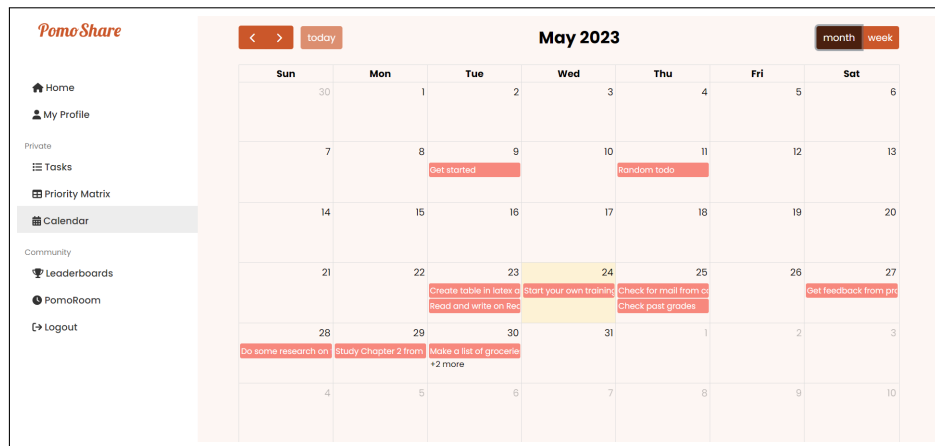


Figure 5.19: Calendar Page

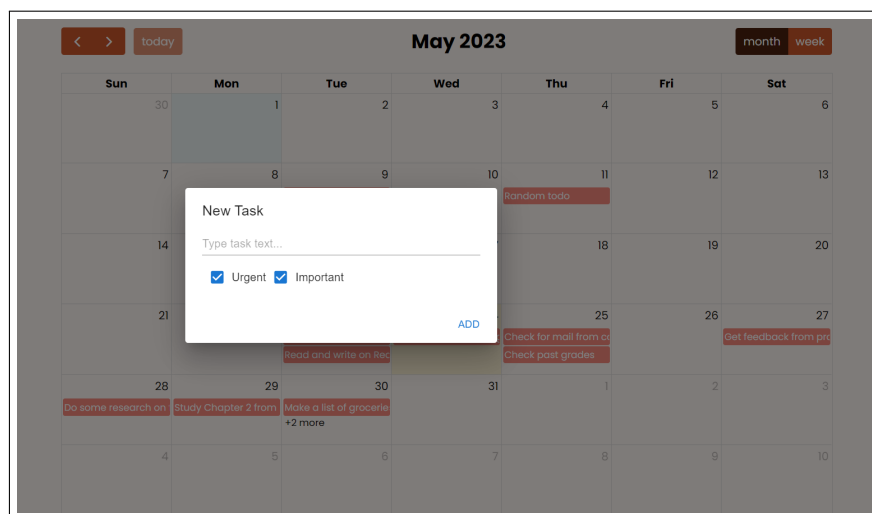


Figure 5.20: Calendar Add New Task

With the Calendar Page, the overview of the application is completed. The various paths have been demonstrated briefly, but to further explore the application, one should actually use it.

Chapter 6

Conclusion

In this chapter, the final results and conclusions that occurred from the thesis will be discussed. Additionally, future adaptations can be demonstrated, since the application is expandable and has optimization margins. With the following sections, the thesis is concluded with an overview and potential for future research.

6.1 Summary

This Thesis is an approach to develop software that is useful and is safe to utilize. The Pomodoro technique appears as the main functionality of the platform and therefore required a lot more complexity and testing. However, the availability of both private and public activities makes the application a type of social media. The gamified approach makes it fun, without focusing a lot on the competition, but rather the collaboration. It is well known that accountability is not only a personal matter but also a team effort, and PomoShare seems to contribute in both.

The software technologies that were used, seem to have great coordination, while the development part was a smooth process. JavaScript provides powerful tools and frameworks that helped throughout brainstorming, developing, implementing and testing. It is crucial to search for the suitable tools in each case, since finding a good fit can make development less time consuming. The client-server architecture is adequately established with error handling when necessary. Also, having the REST API based on the Model-View-Controller architecture has made it easy to implement since the structure is clear to use and adaptable to changes or additions.

One of the most fascinating facts that resulted from the thesis, are the issues that may occur due to the functionality of React [37], especially in the case of the Pomodoro timer using intervals and effects. The way that React works is through an event loop to handle the code execution. The timer elapses after it has been added to the queue of the event loop. Especially when pushing user interaction or functions that require re-rendering updates, the queue might be overbooked. Latency issues then may be caused, not only by the event loop, but also by the single-threaded nature of JavaScript. In other words, JavaScript executes one operation at a time

6.2 Future Work

Additional features can be added to the application to enhance the abilities of productivity tools. Although, having a minimal content with clear directions appears to be a good choice, these methods might not fit everyone's needs. One example for future update of the platform would be a tool that will help limit distractions, such as visiting common social media while focusing on tasks. This can be implemented as a browser plugin extension with the same theme as PomoShare.

It is also a very trending feature in applications to add an AI tool. In the case of PomoShare there could be an AI-powered chatbot to act as an admin in the Pomodoro chats, and even solve questions or help brainstorm the accountability partners. This AI-chat can also accompany a member while they are waiting for the next perfect match to enter the room, somehow replacing the human dependence. There are multiple available frameworks that aim to integrate a chatbot into applications, especially in our days.

A common case in similar applications for engagement growth, include the ability to integrate data from third-party applications. The most popular productivity tools, such as Google Calendar or Microsoft Outlook, can be synchronized with PomoShare in an attempt to create more approachable material. By having a variety of applications available, the problem of staying on track may become even harder and therefore an all-inclusive app can undo that problem. In this way events, meetings, appointments and whole schedules are appropriately displayed to the user.

Future work may also involve the enhancement of an application performance-wise. The application has a margin for scaling and load balancing, since it is a common issue in most

applications. The more users create accounts, the greater the risk for failure. Increased data volume demands for load balancing across multiple servers and leveraging technologies like Docker and Kubernetes to achieve scalability. This could be followed by appropriate performance testing in order to identify performance issues before deployment.

Cross-platform development is another issue that would be a beneficial addition to the project. Especially with the growing liability on mobile devices, customers seek multi-device applications. Although PomoShare is responsive for various screen sizes, accessing it through browser may often be unmanageable. Therefore we should provide applications that run on different platforms such as iOS, Android and the web. Some considerate advantages of mobile applications is the offline functionality, and push notifications for any upcoming event or task.

As a final point, software technologies are limitless and can be appropriately utilized for the benefit of humans. The combination of a web or mobile application with common productivity boosting techniques, might be the modern way of personal growth.

Bibliography

- [1] Staffan Noteberg. Pomodoro technique illustrated: The easy way to do more in less time. *Pomodoro Technique Illustrated*, pages 1–156, 2009.
- [2] Andrew J Martin. The role of positive psychology in enhancing satisfaction, motivation, and productivity in the workplace. *Journal of Organizational Behavior Management*, 24(1-2):113–133, 2005.
- [3] Mark T Morrell and Alex T Krouse. Accountability partners: Legislated collaboration for health reform. *Ind. Health L. Rev.*, 11:225, 2014.
- [4] Stuart K Card and Austin Henderson Jr. A multiple, virtual-workspace interface to support user task switching. *ACM SIGCHI Bulletin*, 17(SI):53–59, 1986.
- [5] N S Jyothi and A Parkavi. A study on task management system. In *2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS)*, pages 1–6, 2016.
- [6] I. Sommerville. *Software Engineering*. International Computer Science Series. Pearson, 2011.
- [7] Pekka Abrahamsson, Outi Salo, Jussi Ronkainen, and Juhani Warsta. Agile software development methods: Review and analysis. *arXiv preprint arXiv:1709.08439*, 2017.
- [8] Heonsik Joo. A study on understanding of ui and ux, and understanding of design according to user interface change. *International Journal of Applied Engineering Research*, 12(20):9931–9935, 2017.
- [9] John Deacon. Model-view-controller (mvc) architecture. *Online][Citado em: 10 de março de 2006.] <http://www.jdl.co.uk/briefings/MVC.pdf>*, 28, 2009.

- [10] Postman. <https://www.postman.com/>.
- [11] Dave Westerveld. *API Testing and Development with Postman: A practical guide to creating, testing, and managing APIs for automated software testing*. Packt Publishing Ltd, 2021.
- [12] Render.com. <https://render.com>.
- [13] Github. <https://github.com/>.
- [14] Github repository for pomoshare frontend. <https://github.com/geokyriakou/productivity-app-frontend>.
- [15] Github repository for pomoshare frontend. <https://github.com/geokyriakou/productivity-app-backend>.
- [16] Mohanish Bawane, Ishali Gawande, Vaishnavi Joshi, Rujuta Nikam, and Sudesh A Bachwani. A review on technologies used in mern stack. *Int. J. Res. Appl. Sci. Eng. Technol*, 10(1):479–488, 2022.
- [17] Saba Alimadadi, Ali Mesbah, and Karthik Pattabiraman. Understanding asynchronous interactions in full-stack javascript. In *Proceedings of the 38th International Conference on Software Engineering*, pages 1169–1180, 2016.
- [18] Node.js. <https://nodejs.org/en>.
- [19] MongoDB. <https://www.mongodb.com/>.
- [20] Evan Hahn. *Express in Action: Writing, building, and testing Node.js applications*. Simon and Schuster, 2016.
- [21] jwt.io. <https://jwt.io/>.
- [22] Salman Ahmed and Qamar Mahmood. An authentication based scheme for applications using json web token. In *2019 22nd international multitopic conference (INMIC)*, pages 1–6. IEEE, 2019.
- [23] cors. <https://expressjs.com/en/resources/middleware/cors.html>.

- [24] Mongoose odm. <https://mongoosejs.com/>.
- [25] Moment.js. <https://momentjs.com/>.
- [26] Socket.io. <https://socket.io/>.
- [27] Andrew Lombardi. *WebSocket: lightweight client-server communications*. ” O’Reilly Media, Inc.”, 2015.
- [28] Vite.js. <https://vitejs.dev>.
- [29] React documentation. <https://react.dev/>.
- [30] Mattias Levlin. Dom benchmark comparison of the front-end javascript frameworks react, angular, vue, and svelte. Master’s thesis, Åbo Akademi University, 2020.
- [31] Elar Saks. Javascript frameworks: Angular vs react vs vue., 2019.
- [32] Daniel Bugl. *Learn React Hooks: Build and refactor modern React. js applications using Hooks*. Packt Publishing Ltd, 2019.
- [33] Axios http requests. <https://axios-http.com/docs/intro>.
- [34] Mui. <https://mui.com/>.
- [35] React router dom. <https://reactrouter.com/en/main>.
- [36] Fullcalendar.io. <https://fullcalendar.io/docs/react>.
- [37] Adam Freeman. Understanding react. *Pro React 16*, pages 31–36, 2019.