CARDIFF
UNIVERSITY

PRIFYSGOL
CAERDYDD

Student Work Tracker Website

CM3203: FINAL YEAR PROJECT

Author: Ruthwik Dhaipulle
Student Number: 21009615

Supervisor: Dr. Matthew Morgan

Table of Contents

Abstract

This report delves into the use of work tracking tools and its benefits. The aim of the report is to justify the need for an all-in-one work tracking web application. The report conducts a literature review on the solutions that currently exist, identifies their limitations, and proposes a solution that aims to address some of those limitations. It argues how an all-in-one work tracker can help improve efficiency, usability, convenience, and time-management for students to track work. It describes the requirements, design, and the implementation of the project. Lastly, it showcases the testing done for the project and the findings of those tests.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Matthew Morgan for their help and support during each and every stage of the project. Dr. Morgan's guidance has been especially valuable in figuring out the aims of the project and the needs of the user.

1. Introduction

1.1 Description

This report seeks to address the need for an all-in-one student work tracker for students. It will explore the solutions that currently exist and identify its limitations. It will compare and evaluate the features and functionalities of those solutions based on how well they meet the needs of students. Furthermore, it will justify how an all-in-one tracker can improve efficiency and convenience for students and help them manage their work effectively.

1.2 Report Structure

This report has nine sections, to explore the digital work tracking solutions that exist and to justify the need for an all-in-one work tracking web application.

Section 1 briefly introduces the problem at hand and gives an understanding of what this paper is about.

Section 2 describes the digital work tracking systems that exist, their limitations, and the benefits of digital work tracking.

Section 3 provides a literature review of current approaches that exist in digital work tracking and justifies the need for my approach.

Section 4 describes the methodology used for the project.

Section 5 defines the requirements of the project, its design, and its implementation.

Section 6 presents the results of testing the web application.

Section 7 concludes the project by exploring what the purpose of this project was and what I hope to achieve with it.

Section 8 notes down how the web application could further be developed and what I could have implemented in the future.

Section 9 reflects on what I have learned overall during this project, and the challenges I faced.

2. Background

2.1 Digital Work Tracking and the Limitations of Physical Work Tracking

87 percent of students believe that improving their time management and organisation skills would help them get better grades. There are several ways to do this. One such way, is by using a popular tool called a work tracker. A work tracker is a tool that can help students keep track of their assignments, lectures, assessments, and tasks for the day. Normally students would use a physical planner to write down their work. In fact, 48 percent of students manage their assignments and deadlines by using a written calendar (Doodle, 2021). While this works, there are a few limitations that can be a hindrance. One such limitation/problem, is that physically writing by hand is time-consuming. Some people prefer to track their work by handwriting it, as it helps them remember better. In fact, behavioural studies show that handwriting helps students learn and retain information better than typing (Ihara et al., 2021). Despite this, it can be impractical when workload increases and people get busier. Especially now, as most work and learning is done online, it just makes more sense to digitally track work. Data shows that after the pandemic, the switch to remote working has increased by threefold (92 million in 2021) (Wood, 2022).

2.2 Benefits of Digital Work Tracking

There are several benefits to digital work tracking.

Firstly, using a digital work tracker is convenient. Rather than buying multiple physical items such as planners, you can easily track your work digitally. This makes the process of work tracking simpler and hassle-free. An added convenience is that you can access your work from various devices, such as laptops, phones, and tablets. This means that you can keep on top of your work no matter where you are and what device you have.

Secondly, it is far more efficient compared to a physical planner or work tracker. Having to hand-write tasks and their descriptions in a physical planner/work tracker can be time-consuming, making it inefficient in the long run. A digital planner on the other hand, is far more efficient, as it allows you to type, track, schedule, and organise work with a touch of a button (Deve, 2023). This eliminates the need to manually write, ensuring that you can use time more effectively and efficiently. A key reason why digital planners are efficient is due to the number of features they have. Unlike with physical work trackers, digital trackers allow you to create and organise tasks, set deadlines, and order them in terms of priority. It also lets you to view your tasks, assessments, and lectures in a digital calendar, making it easier to visualize your work and when you have to do it. Furthermore, digital work trackers allow for customisation. With a digital work tracker, you can sort your work using categories, labels, etc. in a way that makes it easier for you to find and access your tasks (Deve, 2023). Additionally, digital planners allow you to integrate with other apps which further increases the efficiency of digital planners. By integrating with other apps, you can have a decent amount of work in one place. For example, Todoist allows you to integrate your calendar, emails, google drive, etc. (*Features* 2023). This partly eliminates the need to use different tools constantly. The vast amount of features available in digital work trackers, its options in customization, and its integration with other apps improves efficiency as it streamlines and simplifies the process in a way that is personalised to better suit the students' need.

3. Literature Review

3.1 Current Approaches

There are several work tracking applications available that help users stay on track digitally, such as to-do lists, Blackboard, Trello, etc. One of the most popular to-do list applications is Todoist. Todoist is a popular to-do list/work tracking application. It allows users to create tasks, organise them in terms of priority, set reminders and deadlines, delegate tasks, and has several customization and organisation features such as themes, boards, labels, and filters (*Features* 2023). Another popular work tracking tool is Trello. Trello is a tool that allows users to manage projects and track tasks, usually for collaboration and group projects. Trello has several features that allow you to create boards to organise work, create lists and cards, set due dates, label work, delegate tasks, etc (*Trello makes it easier for teams to manage projects and tasks* 2023). There are also learning management systems such as Blackboard. Blackboard is a system that allows users to see their lectures, marks, courses, assessments, and assignments. It's usually used by universities to support online learning. While it isn't exactly a work tracking app, it's features can be used by students to track their work. There are several other work tracking applications available that allow students to effectively manage their work. The issue however, is that while these applications do exist, they have a fragmented or singular focus.

3.2 Justification for My Approach

While digital work trackers and planners do exist, they're spread across several different web applications and tools. E.g. Todoist, for the most part, is a to-do list application. While it does have integration with other tools, most of the information is populated by the user. Users have to manually create tasks, set deadlines, and add assessment information. Manually adding this information is inefficient and not feasible, which means that students have to look for another work tracking app to track their lectures and assessments. Trello, on the other hand, focuses on just project management and collaboration. While it is an effective tool to manage projects, it is too bloated in its features to use for simple work tracking. Despite also allowing integration with other apps, Trello requires users to create a board and manually add most of their work information. Similar to Todoist, it does not have a specific assessments or lectures section. Blackboard, is a learning management system that specifically caters to students' needs. It includes almost everything a student needs, as it has a section for courses, learning materials, assessments, marks, and has a calendar. While it may seem like it is an all-in-one work tracking application, it does have its limitations. For example, Blackboard does not have a to-do list. Learning information in Blackboard is significantly fragmented and organised depending on each module and professor. Furthermore, it is hidden behind multiple tabs, which means that users have to spend quite a bit of time to find what they need. Having fragmented information, too many features, and multiple tabs makes it tough to navigate and inefficient as an all-in-one work tracking solution. This is further supported by a study done by Al-Mouh, AlKhalifa & Al-Khalifa, that found that Blackboard does not fully integrate necessary navigational tools, making it tough for students to navigate and access content and features (Alturki et al., 2016).

The few all-in-one solutions that do exist have poor/outdated UI, are tough to navigate, and are usually paid, which is why they aren't widely used. My approach is to create a free, all-in-one work tracking web application that addresses some of these limitations. The solution I propose, is to create a simple, streamlined, multi-purpose work tracker that includes essential work tracking features in one consolidated place. The website would feature registration functionality, login/logout functionality, a to-do list that lets users organise and order tasks in terms of priority, a section for lectures (from Google Calendar for example), and a section that pulls in assessments and gives users a suggested deadline to complete the assessment. I believe that having these features would reduce the need for using various work tracking tools, such as Todoist for a to-do list, Blackboard for assessments, and a university calendar for lectures, as all of it would be in one convenient place. Furthermore, by having a suggested deadline feature, it would offer something that Blackboard does not have. Having a suggested deadline could motivate students to be more productive as they could aim to finish their work by the suggested deadline and not do it in the last minute. By having these features in one application, I believe that my solution would help students effectively manage their work and their time, therefore increasing efficiency and productivity.

3.3 Summary

In summary, the work tracking solutions that currently exist have several limitations, such as fragmented functionality, complex navigation, lack of specific features, etc. My proposed solution aims to address these limitations by creating a web application that is usable, easy to navigate, and includes essential features all in one place. By doing so, I believe that my solution could be a right step in the direction of eliminating the need for using several applications to track work.

4. Methodology

The Agile methodology is a project management methodology that is normally used for projects where the constrains are not well understood. The Waterfall methodology is used for projects that have a concrete timeline and defined deliverables (*Agile. VS Waterfall: Which methodology to use?: Wrike* 2023). Since my project is a final year project, it had a concrete timeline and a strict deadline (May 12th originally, but May 19th due to extenuating circumstances). Due to this, it also had self-defined deliverables, as I needed to complete tasks in phases to ensure that it is doable in the given time frame. To do this, I created a weekly work plan in my Initial report. The constrains were also well understood as I knew the time I have and the overall scope of the project. Since my project had a concrete timeline, well understood constrains, and defined deliverables, I decided to choose the **Waterfall** methodology. Another reason why I chose the Waterfall methodology, is because it is an approach that has a linear sequence from start to end (*Waterfall methodology: Project management | Adobe Workfront*), which I felt best applies for my project.

This meant that I followed the five phases of the Waterfall methodology, which are requirements, design, implementation, and verification (team, *The 5 phases of Waterfall Project Management* 2023). The requirements phase of my project involved the initial planning for the project. For this phase, I created a weekly project plan that defined what needs to be done in each stage of the project from start to finish. For the design phase, I had to figure out information such as how the web application will be built, the languages that will be used, etc. For my project, I had initially chosen to use Flask and Jinja templates. Later, however, I had to change this to use Django instead. The main reason why I did this, was because Flask was a lightweight framework and did not come with several built-in functionalities that were far simpler and easier to implement in Django. For the design phase, I also created simple wireframes such as sketches to visualise (as seen in Figure 1). how the project needs to look. During the implementation phase, I built the web application as defined during the requirements and the design phase. Lastly, for the verification phase, I decided to use unit testing to make sure some of the code works as intended and tested it by going through each use case manually on the web application.

5. Requirements, Design, and Implementation

5.1 Requirements

Since the Waterfall methodology was used, one of the initial steps of the project was to define the requirements. This meant that an initial plan had to be created, by figuring out the goal of the project, its scope, and the features that must be used. The goal of the project was to create a simple, streamlined, all-in-one work tracker that includes essential work tracking features in one place. The scope of the project was outlined in the form of a weekly plan that described the tasks that must be accomplished and the timeline that it must be accomplished under. The features of the project were determined by researching the digital work tracking solutions that exist and its limitations. Through research it was discovered that there are several digital work tracking web applications that exist, such as Todoist, Trello, Blackboard, and several more. These web applications had a different and separate focus from the other. For example, Todoist mainly focused on manually tracking and organising tasks, and setting deadlines and reminders for those tasks. Trello focused on project management and collaboration. Blackboard, focused on university, work, lectures, assessments, marking, and learning materials. These web applications came with several limitations. For instance, while Todoist was a great tool to create, manage, and organise, tasks, it did not have a section for assessments and deadlines, which meant that users had to manually populate the to-do list to keep track of them or they had to use a different tool such as Blackboard.

The following features were picked to address some of the limitations: a to-do list that lets users organise and order tasks in terms of priority, a section for lectures, and a section that pulls in assessments and gives users a suggested deadline to complete the assessment. These features were chosen based on the essential features of several other work tracking applications. A to-do list that lets users organise and order tasks in terms of priority was chosen as it is a key feature in to-do list applications such as Todoist, that is not present in other work tracking applications such as Blackboard. The lectures section was chosen as it can be inconvenient to always go back to the university calendar to keep track of lectures. The assessment section was chosen to reduce the need for students to use systems like Blackboard every time they need to track their assessments and their deadlines. Furthermore, a novel feature of having a suggested deadline was chosen. It was also decided to design these features in a simpler, convenient, and usable way.

5.2 Design

Framework

During the initial stages of the design phase, the Flask framework was chosen for the web application. As this is the framework that we had been taught during the Web Applications module in year 1. This framework was also used for implementation, initially. Later, however, the framework had to be changed to Django. Django was chosen as it is somewhat similar to Flask and uses Python, which made it easier to learn. The main factor in choosing Django over Flask was the benefits it offers over Flask for a project like this. The first benefit of using Django over Flask is that it came with a clear documentation, which made it easy to learn and follow. The second benefit, is that it included several built-in features and functionalities that were particularly useful to create registration functionality, login/logout functionality, authentication functionality, forms, CRUD operations, etc. An additional benefit, is that came with an inbuilt database called SQLite. Which meant that all data could be stored locally, and connecting to a remote database every time would not be necessary.

Frontend

For the front-end side of development, HTML Django Templates were chosen to display and provide the structure to the web application. A benefit of using Django templates is that it is syntactically similar to Jinja templates, which is normally used for Flask. For the design, Bootstrap 4 was used. Bootstrap was used to design the buttons for the web application, and the badges that showcased the level of priority for each task. For tasks with high priority, red badges (bg-danger) were used, as red is associated with importance (Chapman et al., 2010). For tasks with medium priority, light greenish blue badges (bg-info) were used as it's a mild, neutral colour. For tasks with low priority, green badges were used (bg-success), as green is normally associated with things that are easy or not as important. Additionally, instead of CSS, style elements in html were used to create the layout, spacing, and structure for the elements.

Functional Requirements

Functional requirements are used to describe how a system must behave and defines what it needs to do to meet users needs and expectations (What are functional requirements: Examples, definition, complete guide 2023). To define the functional requirements for the projects, use case diagrams were used. Use case diagrams were used as they are an excellent way to visually showcase the scope of the project by describing what the web application can do, what users can do with it, and how they can achieve their goals (Malan & Bredemeyer).

The following use cases show how the web application works, what it does, what users can do with it, and how they can achieve their goals:

Use Case 1	User Registration
Description	In this use case the user registers an account for the web application
Actor	Cardiff University Computer Science 3rd Year Student
Pre-Conditions	The student must have access to the internet, a computer, and a browser.
Normal Flow	1. The user goes to the web application's home page. 2. The user clicks on the Register button on the navbar 3. The user is redirected to the registration page 4. The user enters a valid username

	5. The user enters a valid password 6. The user confirms the password, by entering it again 7. The user clicks on the register button 8. The user’s account is registered, their account is automatically logged in, and is redirected back to the home page, and is now given access to the rest of the web application
Alternate Flow	The user enters invalid information and is unable to register.

Use Case 2	User Login
Description	In this use case the user logins into their valid account for the web application
Actor	Cardiff University Computer Science 3rd Year Student
Pre-Conditions	The student must have access to the internet, a computer, and a browser.
Normal Flow	1. The user goes to the web application’s home page. 2. The user clicks on the Login button on the navbar 3. The user is redirected to the Login page 4. The user enters a valid username 5. The user enters a valid password 6. The user clicks on the login button 7. The user’s account is now logged in, and is redirected back to the home page, and is now given access to the rest of the web application
Alternate Flow	The user enters invalid information and is unable to log in.

Use Case 3	Add/Edit/Delete Tasks

Description	In this use case, the user will create tasks to keep track off.
Actor	Cardiff University Computer Science 3rd Year Student
Pre-Conditions	The student must have access to the internet, a computer, and a browser.
Normal Flow	1. The user goes to the web application’s home page. 2. The user logs into the web application 3. The user clicks on the main dashboard page 4. The user clicks on the “Add a Task” button 5. The user is redirected to the “Add a Task” form/page 6. The user enters a title 7. The user enters a description 8. The user clicks on the “Add” button 9. The user is redirected back to the dashboard page 10. The task shows up on the dashboard page and is set to the default of medium priority
Alternate Flow	1. At step 7 the user does not enter a description and only enters a title and adds a task 2. At steps 6,7, the user does not enter a title or a description and adds a task that has no content in it and shows up as “None” on the dashboard 3. After step 7, the user clicks on the “Accomplished” button to add a task that they’ve already completed and is then redirected to the dashboard where the accomplished task shows up with a line across the title to mark that it is complete.

	4. At step 10, the user clicks on the edit button to either edit the title, description, “accomplished” field, or the priority of the task. The edited task shows up on the to-do list, after the user is redirected to the dashboard. 5. Before step 8, the user sets the priority of the task to either high or low priority to change the order of the tasks in the to-do list. Then the user adds that task. If the user sets it to high priority, the task shows up at the top. If the user sets it to low priority, the task shows up on the bottom 6. After step 10, the user decided to delete the tasks in the todo list. The user clicks on the “Delete” button, and is redirected to a delete confirmation page. The user clicks on delete again, and is redirected to the dashboard where the task is removed from the to-do list 7. After step 10, the user clicks on the “See” button, to see individual tasks in more detail and to see its description. 8. The user decided to not delete the task at the delete confirmation page, and is redirected back to the dashboard where the task is still there

Use Case 4	View Lectures
Description	In this use case the user logins into their valid account for the web application, to see the google calendar of their lectures.
Actor	Cardiff University Computer Science 3rd Year Student
Pre-Conditions	The student must have access to the internet, a computer, and a browser.

Normal Flow	1. The user goes to the web application’s home page. 2. The user clicks on the “Dashboard” button to access the web application 3. The user views the google calendar that is synced to Cardiff university’s schedule for Year 3. 4. The user looks at the lectures for each month 5. The user clicks on an event (lecture) and sees the title, description, and link to the lecture if online 6. The user clicks the “x” button to go back to seeing the overall calendar
Alternate Flow	1. At step 5, the user can either click on “more details” to see more information about the event or can click on “copy to my calendar” to copy this to their google calendar

Use Case 5	View Assessments
Description	In this use case the user logins into their valid account for the web application, to see their assessments.
Actor	Cardiff University Computer Science 3rd Year Student
Pre-Conditions	The student must have access to the internet, a computer, and a browser.
Normal Flow	1. The user goes to the web application’s home page. 2. The user clicks on the “Dashboard” button to access the web application 3. The user scrolls down and clicks on the assessments button.

	4. The user looks at the assessments overall for year 3 Cardiff university students, the hand-out date, the hand in date, and the suggested deadlines.
Alternate Flow	At step 4, the user can click the back button on chrome to go back to the dashboard.

Non-functional Requirements

The non-functional requirements essentially describe how a system should do what it does and, in a way, how it should implement the functional requirements (Rome, *What are non functional requirements - with examples*). For example, usability. The web application should be easy to use and navigate. As usability was one of the key focuses of the project, this was addressed by making sure the web application is clear, convenient, and simple. One way this was done is by designing the web application to have all its features in one tab for the most part in a dashboard style, rather than having multiple tabs. This was done, because having to navigate through several tabs is time consuming and can be frustrating. By having all the features in one tab, users can track all their work in one page, in a clear way. The security of the web application was ensured by only allowing users with a valid username and password to access it. Anyone that is not logged in, would not be able to access the work tracking features. Another way security was addressed, was by user restricting the tasks. Therefore, each user would only be able to see the tasks that they've added, and not others. Compatibility was addressed by making sure that anyone with the source code (and Django installed) could run the web application on most browsers. To take this a step further, it could be hosted online so that having the source code would be unnecessary. The web application was designed to be reliable as it would be able to run most times by anyone with the source code locally. Eventually, it would be hosted on a reliable web hosting server to make sure that it runs well and is accessible whenever required.

Sketches and Wireframes

After determining the goal, scope, and requirements of the project, sketches and wireframes were created to visualise the layout and design of the project. Furthermore, it was used to decide how the project, it's features, and it's requirements should be met and implemented when coding. The basic sketch in figure 1, was used to visualise how each individual feature should look. The sketch/wireframe in figure 2, was used to determine how those features would be laid out on the webpage. In this sketch, it was determined that each feature would be laid out on one tab for the most part. As a user opens the web application, they see the home page. In the homepage, they must register, or login with valid credentials to access the rest of the web application. Once they do, they get access to the Dashboard, which includes the all-in-one work tracker and all its features. At the top, the user sees the tasks they have for the day. Here the user can add their tasks. Once they click on the “Add a Task” button, they are redirected to the form to add the task. Here they can fill out the task information such as the title, description, level of priority, and whether they have accomplished it. Once they add the information they need, they can click on the add a task button which redirects them back to the dashboard, where the task is now seen. It includes alternate flows, such as editing a task and deleting a task. Editing the task, redirects them back to the “Add a Task” form with their task information filled out. Here they can make any changes necessary, such as changing the title or description, or marking it “accomplished” if they have completed it. After this, they can click on the “add” button again, which takes them back to the dashboard and shows the changes made. They can also click on the “delete button, which takes them to a confirmation page. Once they confirm to delete, it redirects them back to the dashboard, where the task is now deleted. They can also click on the “see” button, which takes them to a separate page that shows them the title of the task, when it was created, and its description. In all these stages, there is a back button that redirects them to the dashboard in case they change their mind. As the user scrolls down, they see a calendar imported with their lectures in it. The lectures were retrieved using my university calendar onto Google calendar as a proof of concept. Here, they can click on the individual events/lectures, and they can add it to their own calendar, if they choose to do so. As they scroll further down, they see a button that takes them to the assessment page. The assessment page displays the modules, the title, the hand in date, the hand-out date, and the suggested deadline. After seeing their assessment information, the user can go back to the dashboard.

5.3 Implementation

The project was implemented by using the Django Framework, Django Templates and Bootstrap 4.

The register functionality was implemented using the Form View class which is used for user registration. This class has a built-in form, that is normally used for registration in Django web applications. The login functionality was implemented using the Login View class, which is also built-in to Django. These classes help users create an account, log in, and authenticates them (as seen in Figure 9 and Figure 10).

The To-Do list was implemented using class-based views. The List View was used to render and pass the list of tasks from the database onto the template. Then the tasks were ordered in terms of priority by creating a dictionary and mapping priorities to values that set the order based on priority (as shown in Figure 4). The task information was displayed using Detail View, which is a class that essentially shows the details of the task object such as the title, description, when it was created, etc. The functionality to add tasks was implemented using the Create View class. This class allowed users to create tasks by adding the title, description, priority, and accomplished field onto the form. The form_valid method was used to assign the user that is logged in to the 'user' field in the form. The functionality to edit tasks was implemented using the Update View Class. This class displayed the existing information for each created task in a form and allowed users to edit the information and re-submit the task, which then updated the information for the task in the database. To delete a task, the Delete View class was used. This class allowed users to delete a task or an "object" from the database.

The lectures section was added by connecting the university timetable/schedule to Google calendar. Then, an embed was created from Google Calendar to add to the html page (as seen in Figure 5).

Since the assessment information for Year 3 Cardiff University students was given as a pdf, the pdftables API was used. This was done by uploading the assessment information pdf to the pdftables.com website. Then an api key was generated. This was then used to create a view that converted the pdf to csv using the pdftables API, read the csv into a pandas dataframe, and then saved it to the Assessment model. Then a template was created that displayed the information in columns (seen in Figure 6). To implement the suggested deadline feature, a simple algorithm was created (shown in Figure 11). This algorithm first checked that the hand-in date is not empty, and then extracted the hand-in date. Then it converted the date by using the datetime library and the strptime function. Then it used the timedelta function. This function was used to create differences of two weeks and one week. Using this, the algorithm then checked each assessment and calculated a suggested deadline based on contribution. Contribution describes how much an assessment is worth for the overall module grade. If a module's contribution was above 50% of the module grade, it suggested that the user should complete the assessment two weeks before the deadline. If the contribution was below 50% of the module grade, it suggested that the user should complete the assessment one week before the deadline. The logic here was that if an assessment is worth more than 50%, it is quite important to the module grade and therefore must be completed well in advance. If an assessment is worth less than 50%, finishing a week in advance would be more than enough. The purpose of the suggested deadline is to help students manage their time a bit more wisely. If they aim to finish their work by using the suggested deadlines, they can stay on top of their work and won't have to work last minute. Furthermore, by completing work on the suggested deadlines, students can then use the remaining time to improve and make changes to their assessments.

6. Results

The web application was tested by using two different methods. One method was by using a test case built-in and imported from Django. This test case used the UrlTest class to test the URLs for each view in the web application. It used assertEqual to compare the generated URL and the expected URL and made sure that they match and are correct. This was done to make sure that the URLs work and function properly and as they are expected to. As seen in Figure 12, all the tests passed, which means that all the URLs work as they're supposed to.

The other method used for testing was to manually test each use case and user flow. This method was used to ensure that the web application, its functionalities, its templates, and its buttons work as expected. The use cases created during the design phase and in the design section were used for the most part to test this.

To test each use case, I decided to play the role of the actor, which in this case is a student. To start off, I tested Use Case 1, which was on the registration functionality. To do this, I went to the home page of the web application. I then clicked on the register button on the navbar, which redirected me to the registration page. This page showed a form where I could add a valid username and password and could register a new user. When I entered invalid information, I was unable to create a user. Once I entered a valid username, password, and confirmed the password, I was able to create a user and it automatically logged me into the web application as that user. To test the login functionality from Use Case 2, I went to the home page of the web application and clicked on the login button. This redirected me to the login page, where I could log in if I entered a valid username and password that already exists. Once I did, it logged me in and gave me access to the rest of the web application. I tested the To-Do list functionality by following Use Case 3. I started off by going to the home page of the web application, logging in, and clicking on the main dashboard page. Then I clicked on the "Add a Task" button which redirected me to the "Add a Task" form. Here I entered a title, description, and clicked on the "Add" button. Once I did so, it redirected me to the dashboard, and I could see my task in my dashboard/in my to-do list. I explored the alternate flows by attempting to only enter a title, only enter a description, or entering nothing and creating the task, which were all successful. I went through other alternate flows by checking whether accomplishing a task puts a line through the task name, and whether setting priority orders the tasks in terms of it. These tests were also successful. I tested the lectures functionality by following Use Case 4. This use case was fairly simple, as I just had to go to the web application, register or log in, and scroll down to view the embedded Google Calendar with the lectures information. I went through alternate flows by clicking on more details to see more information about the event and by clicking on "copy to my calendar" to copy it to a separate Google Calendar. This use case test was also successful. Lastly, to test the assessments feature, I followed Use Case 5. To test the assessments functionality, I went to my web application, logged in, clicked on the dashboard button and scrolled down to the assessments button. Then, I clicked on the assessment button which redirected me to the assessment page and showed me the assessments overall, their titles, their hand out date, their hand in date, and the suggested deadline for each assessment. The alternate flow was tested by clicking on the back button, which redirected me back to the dashboard.

By following the use cases, I was able to test that the functionalities of the web application work as expected.

7. Conclusion

In summary, the aim of this project was to create a simple and streamlined all-in-one work tracking web application. To do this, research was done on the digital work tracking solutions that exist, the features they offer, and the limitations they have. Based on this research, the features of the web application were determined to address some of these limitations. These limitations were addressed by offering features such as a to-do list, a lectures section, an assessment section, etc. all in one consolidated place. After researching the solutions that exist and determining the features for the web application, the requirements of the web application were determined by creating a weekly work plan to complete the project in the given time frame. The design, structure, layout and user interface were determined through the use of wireframe sketches. The functional requirements were identified by creating use cases for each possible scenario. The non-functional requirements were addressed by ensuring that the web application is safe, secure, compatible, and reliable. The project was implemented by following the functional and non-functional requirements by using the Django framework. After the project was implemented, it was tested through the use of unit tests (for the urls), and manual use case testing for the rest of the functionality.

8. Future Work

This web application offers a simple, streamlined, convenient, and usable solution by having essential work tracking features in one page as a dashboard. There are several other features and functionalities that can further improve the website to promote efficiency and productivity. One such feature, that I was unable to implement is the feature to display stack overflow posts that relate to each assessment topic. This involved working with the Stack Overflow API, and conducting a search based on the assessment title. I tried to implement this feature, but could not get it working in the end, and therefore did not include it in the final product. The lectures section could be improved by creating a proper environment that pulls in lectures information from various tools and calendars, rather than a Google Calendar embed that is connected to the lectures schedule. Another way the web application could be improved is by adding more features to the to-do list that let the user organise, categorise, and color-coded the tasks. Lastly, the most useful feature that could be implemented to the web application is collaboration. I was unable to implement collaboration features, as it involved knowledge of technologies such as web sockets, which was beyond my current capabilities. I believe that making these changes, and adding these features would greatly improve my project and make it something worth bringing to the market, as it would offer an incredible solution to track work in one place and under one dashboard.

9. Reflection on Learning

Through the Final Year Project, I was able to learn various skills that I had never learned before. Through the final year project, I was able to learn a new web framework called Django. The web development knowledge I had before this project only involved lightweight web programming through the use of Flask. This project let me build a fully-fledged web application with CRUD operations and taught me to work with API's which I had never done before. It helped me develop my research skills as I had to compare, contrast, and analyse several solutions that exist to come up with a solution that addressed these limitations. I learned about different testing methods such as unit testing and manual use case testing, which I had never done before. However the most important thing I learned from this project is to create my own project from start to finish. Before this, I had only worked on group projects and small coursework assignments. Due to the large nature of this project, I was able to vastly improve my programming skills, as I learned new tools, technologies, libraries, and languages.

Lastly, I learned the importance of time management and proper planning. Due to several extenuating circumstances, I was unable to follow my weekly work plan exactly. I believe that this is because, while I had a clear workplan, I did not define milestones clearly. Also, I did not factor in the fact that things can over-run. Which is why when certain issues came up, I struggled to follow the weekly work plan.

Appendix

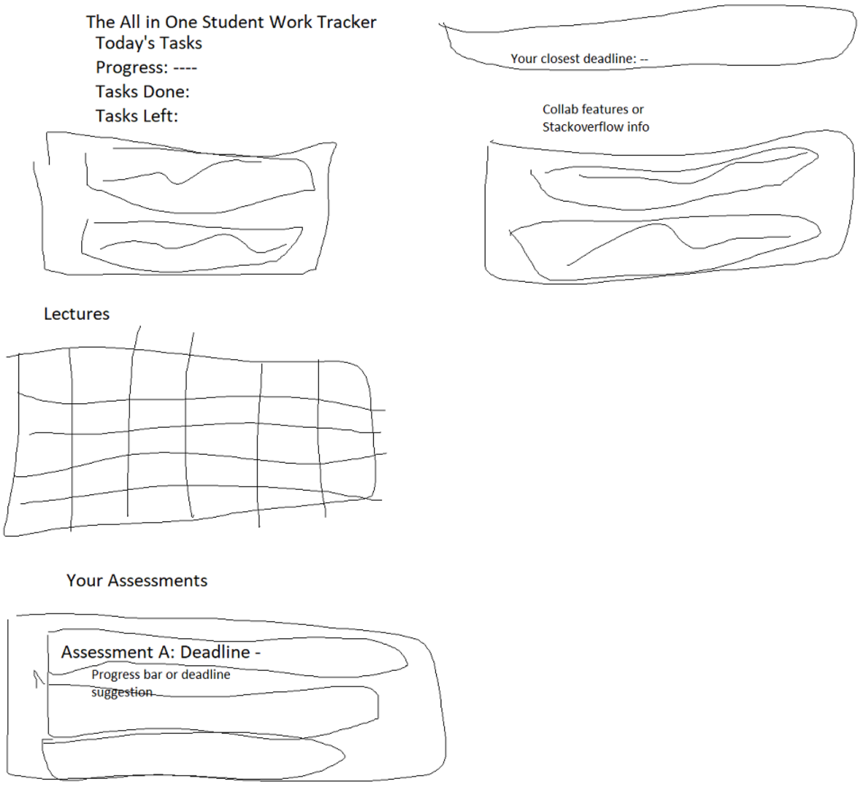


Figure 1. Sketches/Wireframes

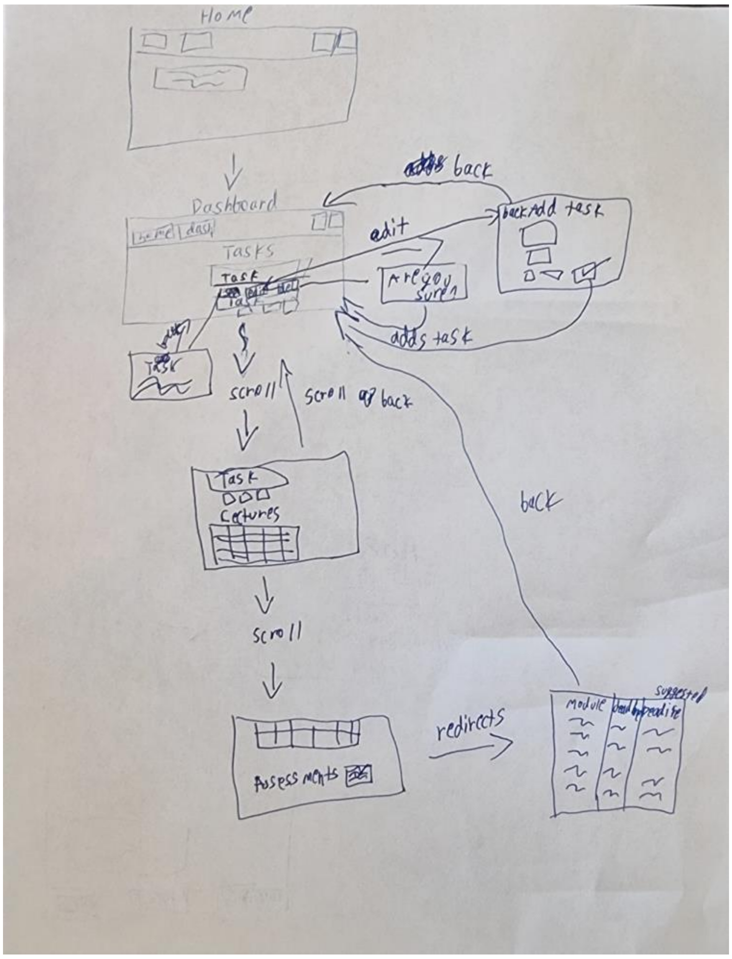
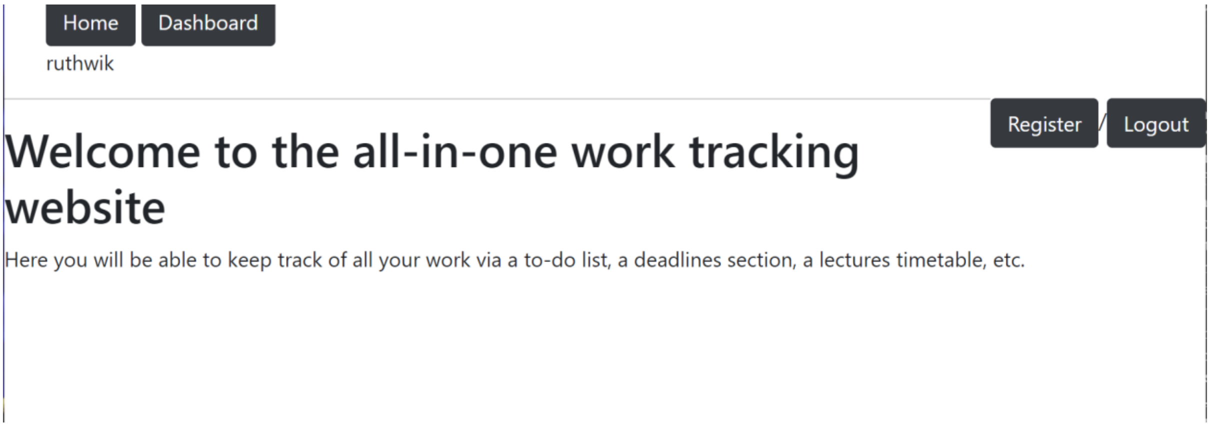


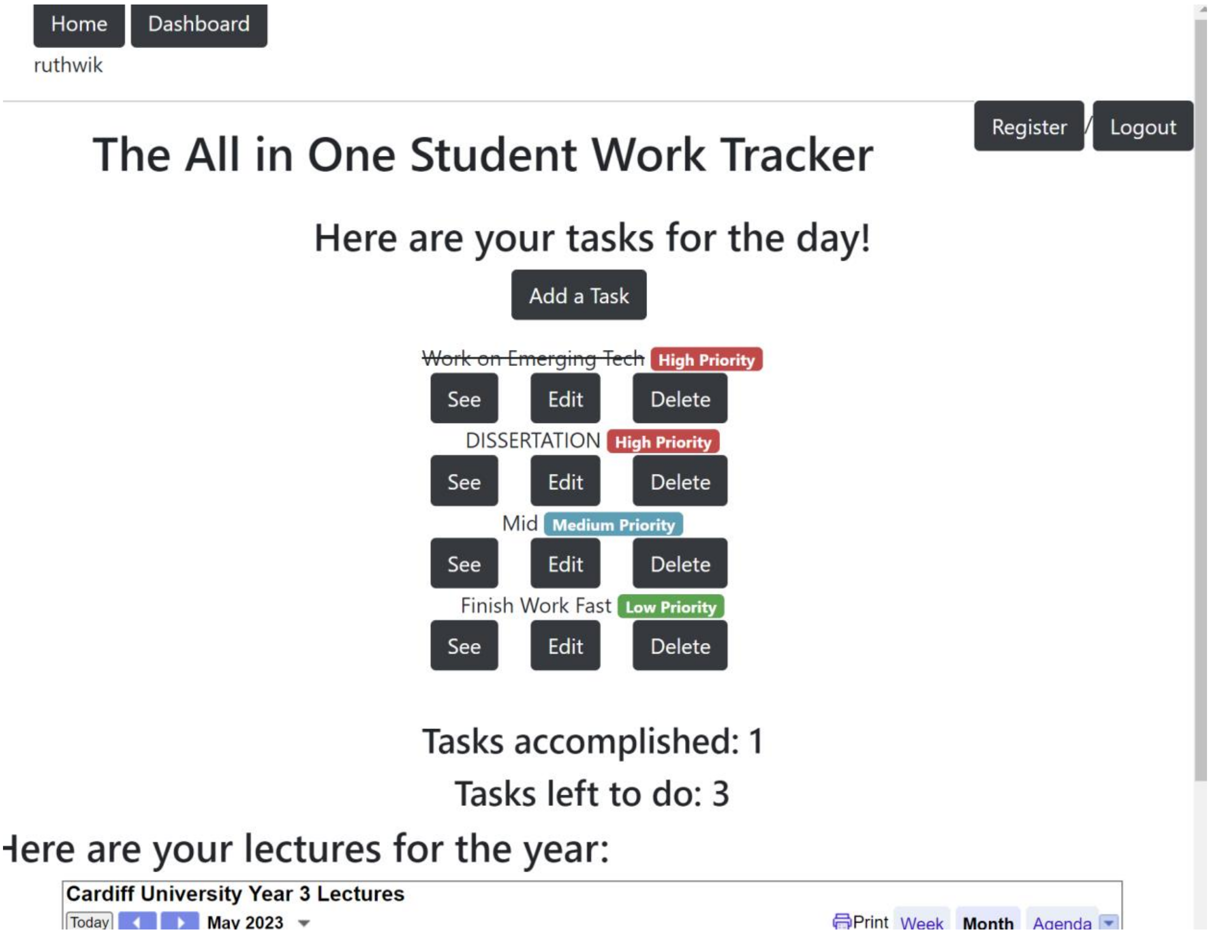
Figure 2. Sketches/Wireframes



Welcome to the all-in-one work tracking website

Here you will be able to keep track of all your work via a to-do list, a deadlines section, a lectures timetable, etc.

Figure 3. Homepage



ruthwik

The All in One Student Work Tracker

Here are your tasks for the day!

~~Work on Emerging Tech~~ High Priority

DISSERTATION High Priority

Mid Medium Priority

Finish Work Fast Low Priority

Tasks accomplished: 1

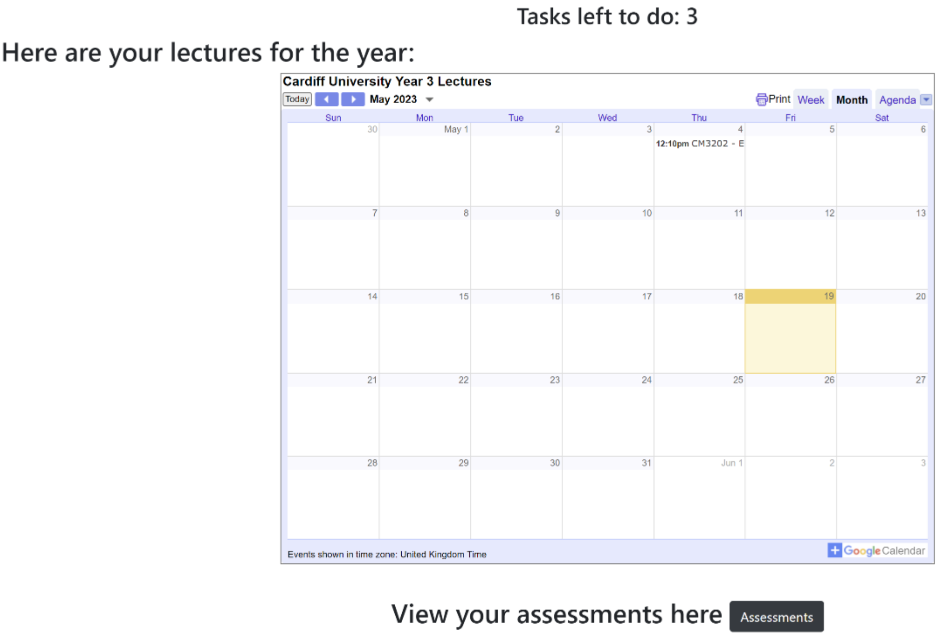
Tasks left to do: 3

Here are your lectures for the year:

Cardiff University Year 3 Lectures

Today May 2023 Print Week Month Agenda

Figure 4. Dashboard – To-Do List



Tasks left to do: 3

Here are your lectures for the year:

Cardiff University Year 3 Lectures

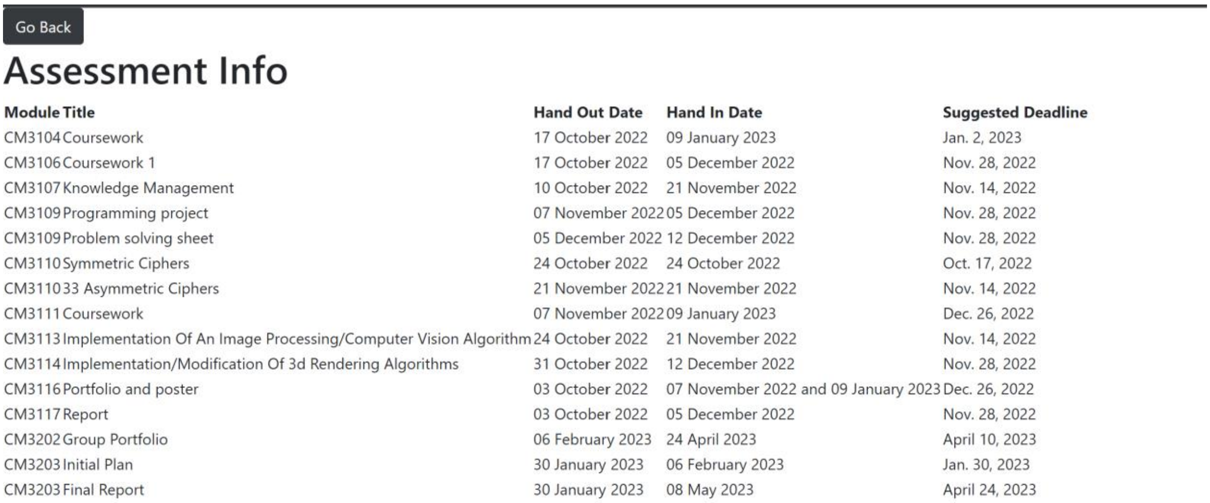
May 2023

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	May 1	2	3	4 12:10pm CM3202 - E	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	Jun 1	2	3

Events shown in time zone: United Kingdom Time

View your assessments here

Figure 5. Lectures and Assessments button

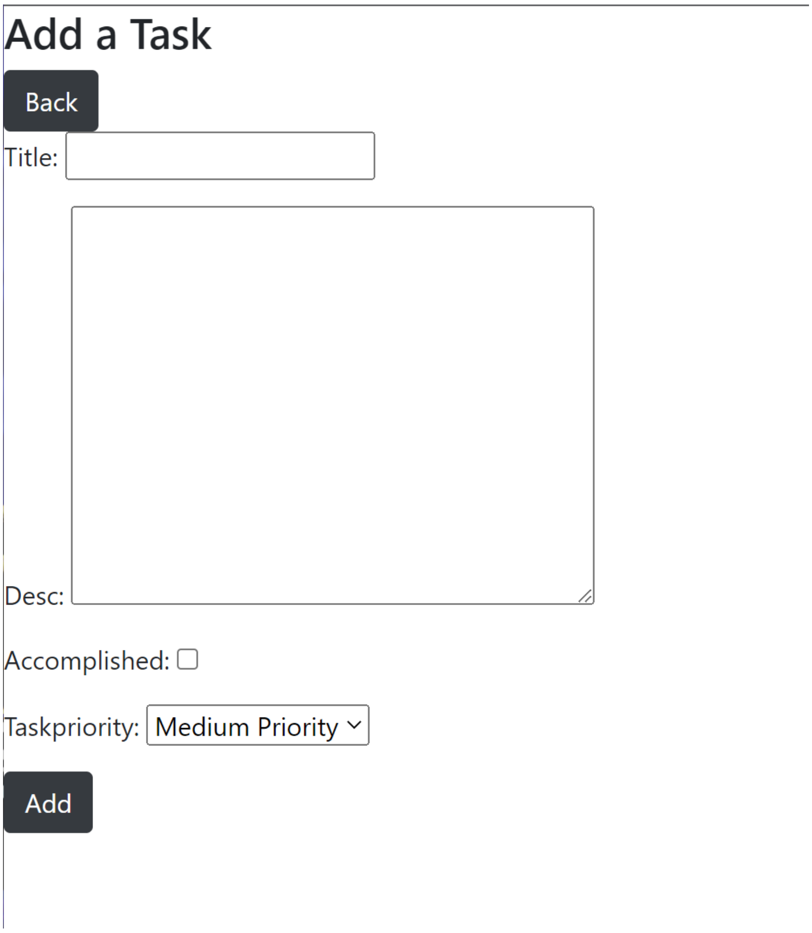


Go Back

Assessment Info

Module	Title	Hand Out Date	Hand In Date	Suggested Deadline
CM3104	Coursework	17 October 2022	09 January 2023	Jan. 2, 2023
CM3106	Coursework 1	17 October 2022	05 December 2022	Nov. 28, 2022
CM3107	Knowledge Management	10 October 2022	21 November 2022	Nov. 14, 2022
CM3109	Programming project	07 November 2022	05 December 2022	Nov. 28, 2022
CM3109	Problem solving sheet	05 December 2022	12 December 2022	Nov. 28, 2022
CM3110	Symmetric Ciphers	24 October 2022	24 October 2022	Oct. 17, 2022
CM3110	33 Asymmetric Ciphers	21 November 2022	21 November 2022	Nov. 14, 2022
CM3111	Coursework	07 November 2022	09 January 2023	Dec. 26, 2022
CM3113	Implementation Of An Image Processing/Computer Vision Algorithm	24 October 2022	21 November 2022	Nov. 14, 2022
CM3114	Implementation/Modification Of 3d Rendering Algorithms	31 October 2022	12 December 2022	Nov. 28, 2022
CM3116	Portfolio and poster	03 October 2022	07 November 2022 and 09 January 2023	Dec. 26, 2022
CM3117	Report	03 October 2022	05 December 2022	Nov. 28, 2022
CM3202	Group Portfolio	06 February 2023	24 April 2023	April 10, 2023
CM3203	Initial Plan	30 January 2023	06 February 2023	Jan. 30, 2023
CM3203	Final Report	30 January 2023	08 May 2023	April 24, 2023

Figure 6. Assessments Page



Add a Task

Back

Title:

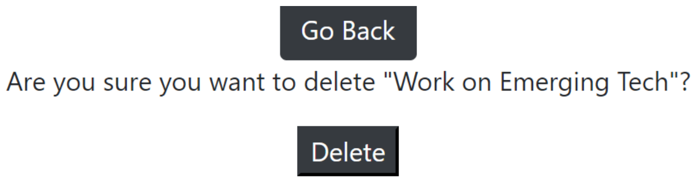
Desc:

Accomplished: ☐

Taskpriority: Medium Priority

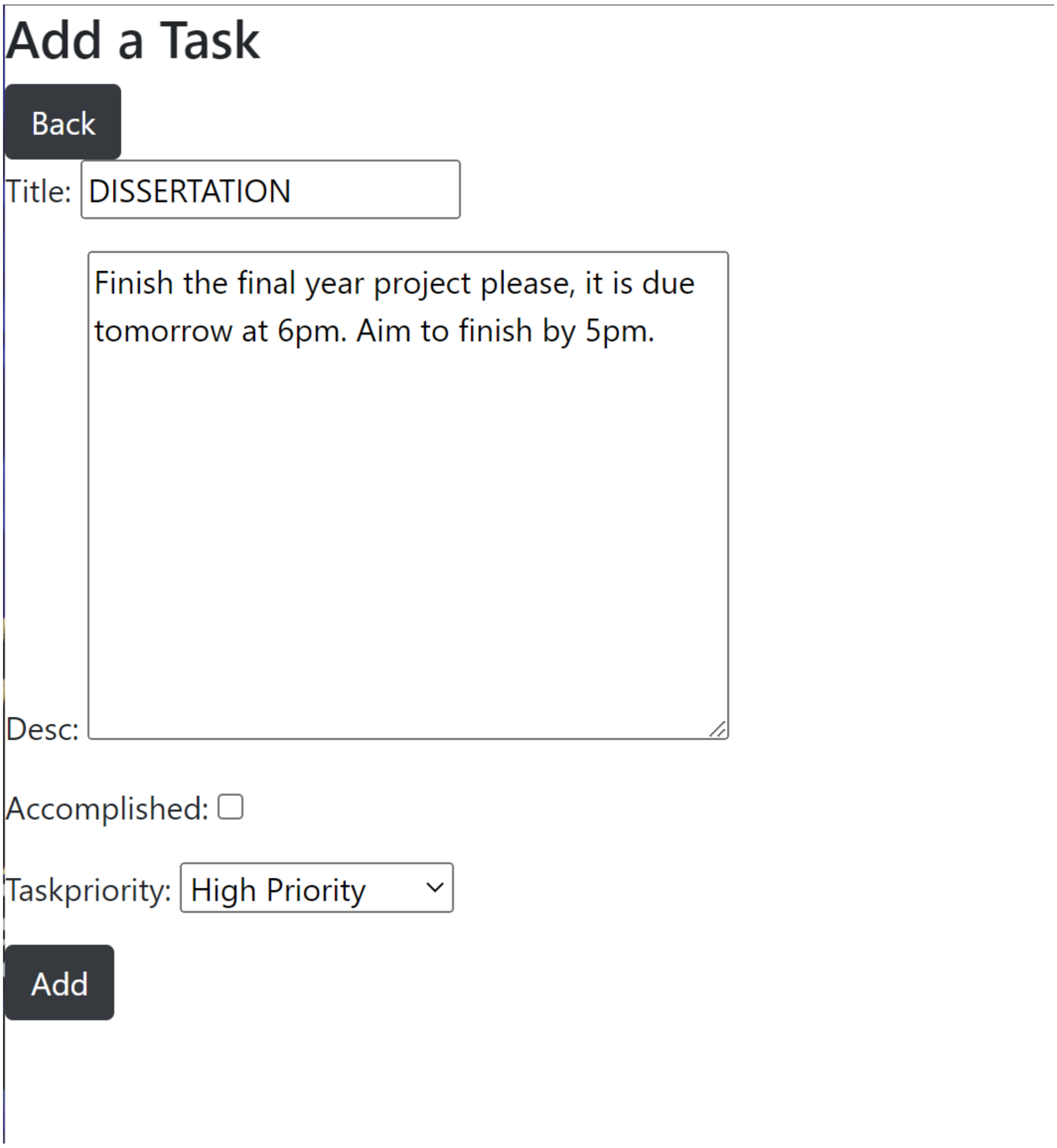
Add

Figure 7. Add a Task Form



Are you sure you want to delete "Work on Emerging Tech"?

Figure 8. Delete Task Confirmation Page



Add a Task

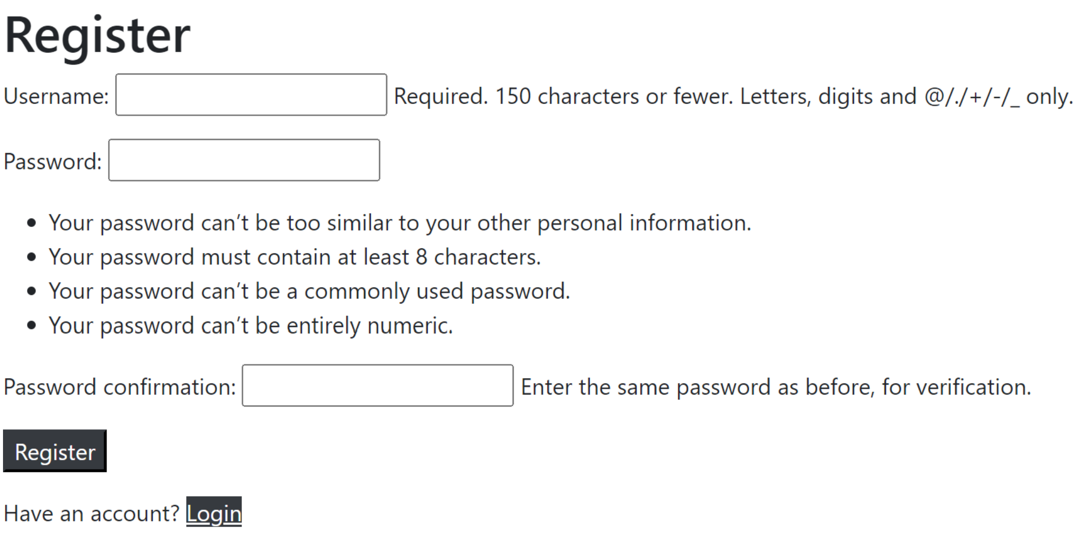
Title: DISSERTATION

Desc: Finish the final year project please, it is due tomorrow at 6pm. Aim to finish by 5pm.

Accomplished: □

Taskpriority: High Priority

Figure 9. Edit Task Form



Register

Username: Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

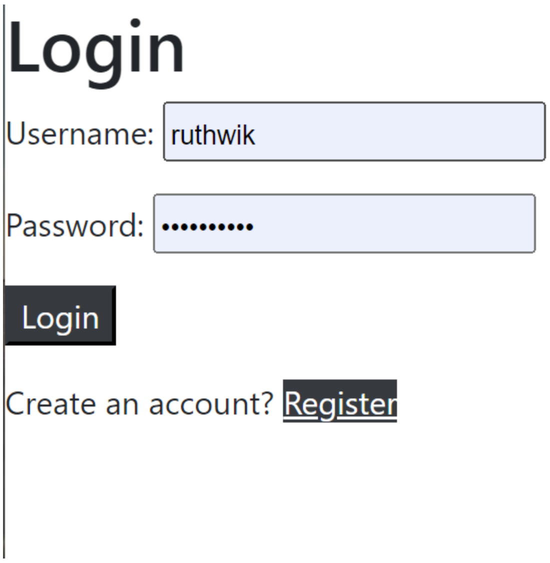
Password:

- Your password can’t be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can’t be a commonly used password.
- Your password can’t be entirely numeric.

Password confirmation: Enter the same password as before, for verification.

Have an account? Login

Figure 9. Registration Form



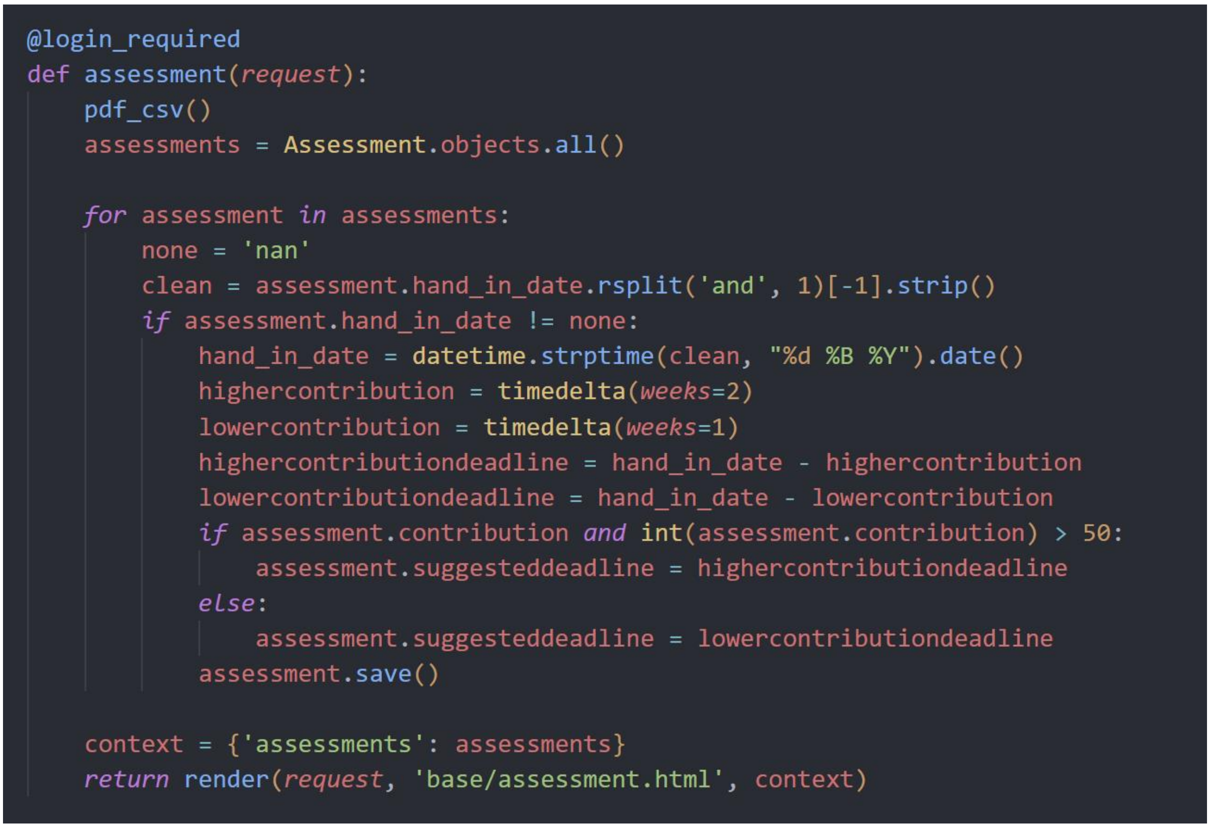
Login

Username: ruthwik

Password: ••••••••••

Create an account? Register

Figure 10. Login Form

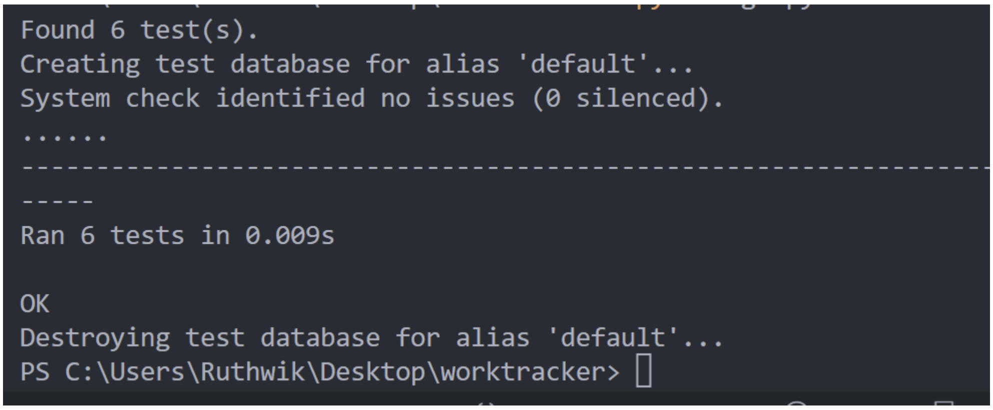


```
@login_required
def assessment(request):
    pdf_csv()
    assessments = Assessment.objects.all()

    for assessment in assessments:
        none = 'nan'
        clean = assessment.hand_in_date.rsplit('and', 1)[-1].strip()
        if assessment.hand_in_date != none:
            hand_in_date = datetime.strptime(clean, "%d %B %Y").date()
            highercontribution = timedelta(weeks=2)
            lowercontribution = timedelta(weeks=1)
            highercontributiondeadline = hand_in_date - highercontribution
            lowercontributiondeadline = hand_in_date - lowercontribution
            if assessment.contribution and int(assessment.contribution) > 50:
                assessment.suggesteddeadline = highercontributiondeadline
            else:
                assessment.suggesteddeadline = lowercontributiondeadline
            assessment.save()

    context = {'assessments': assessments}
    return render(request, 'base/assessment.html', context)
```

Figure 11. Suggested Deadline Algorithm



```
Found 6 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
......
----------------------------------------------------------------------
-----
Ran 6 tests in 0.009s

OK
Destroying test database for alias 'default'...
PS C:\Users\Ruthwik\Desktop\worktracker> 
```

Figure 12. URL's Test Case

References

Agile. VS Waterfall: Which methodology to use?: Wrike (2023) *Versatile & Robust Project Management Software*. Available at: https://www.wrike.com/project-management-guide/faq/when-to-use-agile-vs-waterfall/ (Accessed: 19 May 2023).

Alturki, U.T., Aldraiweesh, A. and Kinshuck (2016) *Evaluating The Usability And Accessibility Of LMS "Blackboard" At King Saud University*, *Contemporary Issues in Education Research – First Quarter 2016*. Available at: https://files.eric.ed.gov/fulltext/EJ1087602.pdf (Accessed: 19 May 2023).

Chapman, C., About The AuthorCameron Chapman is a professional Web and graphic designer with over 6 years of experience. She writes for a number of blogs and is the author of The Smashing …More aboutCameron ↬ and Author, A.T. (2010) *Color theory for designers, part 1: The meaning of color*, *Smashing Magazine*. Available at: https://www.smashingmagazine.com/2010/01/color-theory-for-designers-part-1-the-meaning-of-color/#:~:text=Red%20can%20be%20associated%20with,warning%20labels%20are%20often%20red). (Accessed: 19 May 2023).

Deve, M. (2023) *The ultimate benefits of digital planners*, *Times of India Blog*. Available at: https://timesofindia.indiatimes.com/readersblog/levitatinginmywords/the-ultimate-benefits-of-digital-planners-49360/ (Accessed: 18 May 2023).

Doodle (no date) 4 statistics every university should know: Doodle blog, Free online meeting scheduling tool. Available at: https://doodle.com/en/resources/blog/4-statistics-every-university-should-know/ (Accessed: February 6, 2023).

Features (2023) *Todoist*. Available at: https://todoist.com/features (Accessed: 19 May 2023).

Ihara, A.S. *et al.* (2021) *Advantage of handwriting over typing on learning words: Evidence from an N400 event-related potential index*, *Frontiers in human neuroscience*. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8222525/ (Accessed: 18 May 2023).

Malan, R. and Bredemeyer, D. (no date) *Functional requirements and use cases - bredemeyer*. Available at: https://www.bredemeyer.com/pdf_files/functreq.pdf (Accessed: 19 May 2023).

Rome, P. (no date) *What are non functional requirements - with examples*, *Perforce Software*. Available at: https://www.perforce.com/blog/alm/what-are-non-functional-requirements-examples (Accessed: 19 May 2023).

team, Profit. co (2023) *The 5 phases of Waterfall Project Management*, *Best OKR Software by Profit.co*. Available at: https://www.profit.co/blog/task-management/the-5-phases-of-waterfall-project-management/#:~:text=Phases%20of%20waterfall%20project%20management%20differ%20from%20one%20project%20to,implementation%2C%20verification%2C%20and%20maintenance. (Accessed: 19 May 2023).

Trello makes it easier for teams to manage projects and tasks (2023) *Trello*. Available at: https://trello.com/tour (Accessed: 19 May 2023).

Waterfall methodology: Project management | Adobe Workfront. Available at: https://business.adobe.com/blog/basics/waterfall (Accessed: 19 May 2023).

What are functional requirements: Examples, definition, complete guide (2023) *Visure Solutions*. Available at: https://visuresolutions.com/blog/functional-requirements/#:~:text=A%20functional%20requirement%20is%20a,features%20that%20the%20user%20detects. (Accessed: 19 May 2023).

Wood, J. (2022) These 3 charts show the global growth in online learning, World Economic Forum. Available at: https://www.weforum.org/agenda/2022/01/online-learning-courses-reskill-skills-ga p/ (Accessed: February 6, 2023).