Task Management Web Platform

Submission Date: 17.11.2023

Trung Hao Tran	z5405840	Scrum Master	trung_hao.tran@student.unsw.edu.au
Tianyun Zhou	z5168660	Programmer	tianyun.zhou@student.unsw.edu.au
Qingyue Wang	z5447824	Programmer	qingyue.wang@student.unsw.edu.au
Xiaofei Wang	z5375381	Programmer	xiaofei.wang2@student.unsw.edu.au
Yongqiang Chen	z5402512	Programmer	yongqiang.chen@student.unsw.edu.au

University of New South Wales

Sydney, NSW 2052

Australia

Table of Contents

Overview

The project is dedicated to developing a Task Management Web Platform, aimed at enhancing communication between clients and service providers, more specifically, between students and tutors, ensuring that tasks or jobs are completed as per expectations and requirements. The design of this platform considers multi-device compatibility and responsiveness to cater to diverse user needs.

In terms of user functionalities, we have designed a range of modules to enhance user experience. The Profile Management module allows users to create, edit, and maintain their profiles, simplifying personal information management. The Task Management module enables clients to effortlessly post tasks, with service providers able to bid on these tasks, thus facilitating interaction between both parties. The Milestone Management module is a significant innovation of the platform, strengthening communication and managing expectations through task progression tracking. Additionally, the Communication Management module provides an effective channel for communication between clients and service providers. The User Management module covers various aspects from account creation to role assignment and access control, comprehensively managing the user experience.

Regarding the system architecture, we have adopted the MERN stack (MongoDB, Express.JS, React, and Node.js), forming a synergistic JavaScript development environment. The system architecture is divided into several layers. The Interface Layer is built with HTML, CSS, JavaScript, and React, offering a user-friendly interface. The Business Layer, comprising Node.js (backend server) and Express.js (server-side framework), handles business logic. The Database Layer uses MongoDB, ensuring effective data storage and retrieval. Moreover, the Infrastructure Layer enables software compatibility with Linux and Windows platforms, extending the system's applicability.

Furthermore, we have integrated key libraries and APIs to enhance the platform's functionality. For instance, using Google Maps for location-based services, Material UI for front-end design. These integrated tools and services not only enhance the platform's performance but also provide users with a richer and more convenient experience.

Functionalities

Map Functionalities to All Project Requirements

This part describes how each module of our project basically maps to all the requirements of the project.

I. Signup and Login

Registration and login are the basic functions of the online platform. After entering the platform, users with different needs can register and login. After logging in, users with different roles will see different interfaces and perform different operations.

Requirements:

Users with different roles and responsibilities can create their account, login and operate the system based on their roles and responsibilities.

Client users can register and create their accounts and login to the platform (following the standard registration way).

Service tasker users can create/register their account with the platform.

II. Profile Management

The Profile Management function ensures that all users have control over their personal information within the platform. For all user types (administrators, client users, and service taskers), this function allows the creation, editing, and maintenance of individual profiles. Administrators have the privilege to oversee and manage profiles across the platform, ensuring data integrity and compliance with platform standards. Client users and service taskers, upon successful registration and login, have the ability to customize their profiles, adding relevant personal information, skills, and preferences. For service taskers specifically, a detailed and comprehensive profile including their skills, experience, and availability can be managed, allowing them to effectively advertise their capabilities to potential clients.

Requirements:

Administrator users have the full privilege in the system and should be able to manage and oversee the platform.

Users can edit/maintain their profiles.

III. Task Management

Task Management connects the needs of clients with the services of taskers. Clients post detailed descriptions of tasks including title, category, location, time, and other relevant specifications. Service taskers use search and filter options to browse through available tasks. Upon finding a suitable task, taskers can place a bid with a message to the client. Clients can view all bids, evaluate them, and select a tasker to complete the work.

Requirements:

Administrator users have the full privilege in the system and should be able to manage and oversee the platform.

Clients can post details of the work that needs to be completed including title, description, categories, location, time, images, and frequency.

Clients can see dashboard highlighting the work/tasks they posted. They can choose to see details of any of these tasks.

Clients can see the details of each task and service tasker they bid on it.

Clients can choose a service tasker to complete the task.

Clients can pay the service tasker (hypothetically).

Service tasker users can edit and update their profile (e.g., personal info, skills).

Service tasker users can search and browse lists of available work/tasks with filters (e.g., location, time, description)

Service tasker users can view the details of a task and bid on it with a message.

Service tasker users can receive notification about the task allocation.

Service taskers can see the dashboard of allocated tasks and its status.

Service taskers receive the payment (hypothetically) one approved by the platform admin.

IV. Milestone Management

Task Management connects the needs of clients with the services of taskers. Clients post detailed descriptions of tasks including title, category, location, time, and other relevant specifications. Service taskers use search and filter options to browse through available tasks. Upon finding a suitable task, taskers can place a bid with a message to the client. Clients can view all bids, evaluate them, and select a tasker to complete the work.

Requirements:

Clients and service taskers can update task status (in progress, completed).

V. Communication Management

The Communication Management function is designed to offer effective communication between clients and service taskers. It allows both taskers and clients to discuss task details, clarify doubts, and stay coordinated throughout the task completion process. All users are also able to report issues faced during

Requirements:

Clients can communicate with any of the service taskers who bid on their task.

Clients can rate their experience with the service tasker.

Service taskers and clients can communicate via a messaging system once a bid is accepted.

Service taskers can rate their experience with the work/task/client.

VI. User Management

User Management oversees the entire user experience, from account creation to role assignment and access control. It guarantees that users can securely register, log in, and operate within the system based on their designated roles and responsibilities. Admin users possess extensive privileges, including the ability to manage and monitor all platform activities and user interactions, ensuring the platform's integrity and compliance with established standards. This function is integral in maintaining the security and functionality of the platform, ensuring that all users, whether clients or service taskers, have appropriate access levels and permissions for their roles.

Requirements:

Administrator users have the full privilege in the system and should be able to manage and oversee the platform.

Implementation of Functionalities

This part describes how each module of our project is implemented and how these functionalities map to project objectives and linked together into a consistent story.

I. Signup and Login

Implementation:

The signup and login functionalities are critical components of user management in any application. Our task management system uses Mongoose for MongoDB interactions, “bcrypt” for password hashing, and “jsonwebtoken”(jwt) for generating tokens that are used for session management.

The user model (userModel.js) is structured for storing usernames, emails, passwords, roles, and other relevant information. The email field is used for identifying individual users, which is marked as unique to avoid duplicate entries. We use the “bcrypt” library to hash passwords before they are stored in the database, ensuring that user credentials are securely managed.

During the signup process, we validate input data using the validator library to check for valid email formats and strong passwords. Upon successful creation, a JWT token is generated, which is used for maintaining a secure and stateless authentication mechanism.

For login, we retrieve the user record by email and use “bcrypt” to compare the hashed passwords. If the credentials are correct, we generate a JWT token, similar to the signup process. Additionally, we have implemented a method to check if a user is an admin, which is a property of the user model,, which will allow for role-based access control throughout the application. After login, the front end sends the JWT token in the header in each request, and the server will identify the user by the token and also retrieve related user information.

Mapping to Objectives:

This implementation directly maps to the project requirements, which provides a secure and efficient system for users to create accounts and login based on their roles. The system supports different user roles, such as 'user' and 'admin' (which is easy to add and expand new roles if there are new needs), with default roles set to 'user'. This role-based mechanism ensures that users have access to functionalities pertinent to their roles, which is a fundamental requirement for the platform.

The JWT tokens used for session management enable users to remain logged in for a specified period, which is set to expire in 3 days (which can be modified) as per the token creation settings. This enhances user experience by reducing the need for frequent logins.

Relationship with other functions:

The signup and login process is the fundamental of identifying users and storing data related to users. During the whole story, the server identifies the user by checking the token created by the signup and login process.

II. Profile Management

Implementation:

For individual users, profile management is implemented through a backend function called updateUserProfile. This function is designed to handle HTTP requests that carry user profile data to be updated. The function identifies the user by extracting the user ID from the request object, a method indicating authentication is in place to ensure that users can only update their profiles. Once the user is identified, the function executes a MongoDB operation User.findByIdAndUpdate to update the user's data with the new information provided in the request body. This operation returns the updated user object, which confirms that their profile has been successfully updated.

For administrators, they can update user profiles via a separate function named updateUser. This function retrieves the user ID from the request parameters, which allows administrators to specify which user's profile they intend to update. The function also uses the User.findByIdAndUpdate method to modify a user's profile.

Mapping to Objectives:

This implementation maps to the project's requirements by allowing users to edit and maintain their profiles and giving administrators the ability to manage and oversee all user profiles. The differentiation in functionality between the updateUserProfile and updateUser functions directly corresponds to the different privileges assigned to regular users and administrators.

Relation to the whole story:

The profile management functions enable users to customize their account, and also provide useful information for other users to make judgment during bidding and accepting bidding.

III. Task Management

Implementation:

The task management system is centered around a data model (taskModel.js) that includes fields for title, description, categories, timing, images, pricing, and user associations, among others. This model is versatile, capturing a wide range of task attributes, including an array of milestones for progress tracking.

Task-related operations are facilitated through taskController.js, which includes functions to create, update, delete, and retrieve tasks. The creation of a task (createTask) involves validation to ensure all required fields are populated before saving to the database. The getTasks function is responsible for retrieving all tasks, with an option to filter tasks based on various criteria provided by the user.

Management functions include assigning service taskers to tasks (assignTasker) and managing the lifecycle of a task with operations like deletion (deleteTask) and updates (updateTask). Moreover, the system includes functionality to retrieve all taskers bidding on a given task (getTaskers).

In addition, our task management system includes a payment function (payment) to handle financial transactions within the platform. This function is designed to manage the virtual credit system that allows clients to pay service taskers for completed tasks.

Mapping to Objectives:

This implementation fulfills several key project requirements. Clients can post detailed tasks, view a dashboard of their tasks, interact with service taskers who bid on their tasks, and pay service taskers (hypothetically, in this virtual credit system). The ability to update task status addresses the need for clients and taskers to track the progress of tasks. The system's design ensures that administrators have full privileges over the task management process.

Relation to the whole story:

As a task management system, task management is the core of the application. All other parts of the application are implemented for supporting this part. By releasing tasks, bidding and accepting bidding, students could solve their issues and tutors could earn money, which is one of the key values the application brings to users.

IV. Milestone Management

Implementation:

Milestone management in our task management system allows for tracking the progress of tasks through specific, definable benchmarks. Implemented through the addMilestoneToTask function, users can add an array of milestones to a task, each with its own title, description, due date, priority, and status. This addition is made via an update to the task document in the database using the findOneAndUpdate method, ensuring that the milestones are appended correctly.

The updateMilestoneStatus function enables the user to update the status of a specific milestone within a task. This function leverages the Mongoose method findById to locate the task and then directly manipulates the milestone's status before saving the updated task document.

Moreover, the getMilestonesForTask function allows for the retrieval of all milestones associated with a task, presenting a clear picture of task progress to the user.

Mapping to Objectives:

This detailed milestone management functionality directly supports the project requirement that clients and service taskers can update task statuses. It allows both clients and service taskers to have a clear understanding of the task's progression towards completion.

Relation to the whole story:

Milestone Management is implemented to support the task management. With milestone management, students and tutors can share the status of the task and thus improve the efficiency of communication.

V. Communication Management

Implementation:

The communication management feature is built around a message schema (messageModel.js) designed to facilitate interactions between users within the task management system. This schema includes references to both the sender and receiver, ensuring that each message is associated with two users within the system. Additional fields such as messageText, createdAt, and isRead provide the necessary structure for messaging functionality.

Mapping to Objectives:

This implementation directly correlates with the project's requirements for effective communication management. Users can send and receive messages, ensuring they can discuss task details, clarify doubts, and coordinate effectively.

Relation to the whole story:

Communication management is to support the text communication between students and tutors during task management. By the communication system, users could represent their requirements and ideas more precisely.

VI. User Management

Implementation:

User management is a foundational aspect of our task management system, ensuring that users can be created, updated, and removed as necessary. getAllUsers retrieves all user accounts, while getUser fetches details for a single user, which is vital for both administrative oversight and for users to access account details. updateUser enables modifications to user profiles, which is critical for users to maintain up-to-date profiles and for admins to manage user accounts. deleteUser allows for the removal of user accounts, which is an essential feature for maintaining the integrity of the user base and for users who wish to terminate their accounts. Additional functionality includes refresh, which can refresh the authentication token for a user, ensuring continuous secure access to the system.

Mapping to Objectives:

The implementation of these functions directly supports the requirements for comprehensive user management. Administrators can oversee and interact with the full roster of user accounts, ensuring the system is managed effectively. Users are given the autonomy to manage their profiles and interact with the system securely and privately.

Relation to the whole story:

User Management is mainly used for maintaining the system. Administrators could utilize those functions to guarantee the whole system is stable as the time goes.

Third-party Functionalities

MongoDB

Description: MongoDB is a NoSQL database used to store the application's data.[1]

Justification: Selected for its flexibility with document schemas, scalability, and strong community support.

Licensing: MongoDB is released under the Server Side Public License (SSPL), which requires that any modifications to the software be made available to the community. This does not impact our project as we use the database as-is without modifications.

Impact: Allows for rapid development and easy adjustments to data models, improving our ability to iterate on the application quickly.

Express.js

Description: Express.js is a web application framework for Node.js, used to build our server-side API.[2]

Justification: Selected for its minimalist approach and extensive middleware support, which simplifies API development.

Licensing: Express is licensed under the MIT License, which is permissive and allows for free usage, modification, and distribution.

Impact: Enhances our API's reliability and maintainability, and accelerates backend development.

React

Description: React is a JavaScript library for building user interfaces, particularly single-page applications.[3]

Justification: Selected for its component-based architecture and strong ecosystem, which includes numerous libraries and tools.

Licensing: React is MIT licensed, granting us the flexibility to use and share the code freely.

Impact: Streamlines frontend development and provides a responsive and interactive user experience.

Bcrypt

Description: Bcrypt is a library to help hash passwords.[4]

Justification: Selected for its security features, specifically its adaptive function that withstands hardware improvements in password cracking.

Licensing: Released under the BSD License, which is permissive and requires only the preservation of copyright notices and disclaimers.

Impact: Secures user data and ensures authentication features within our application.

Validator.js

Description: Validator.js is a library that provides string validation and sanitization.[5]

Justification: Selected to validate and sanitize user input and prevent common security issues such as SQL injection.

Licensing: Validator.js is MIT licensed, so it does not impose restrictions on the use or distribution of the project.

Impact: Improves the overall security posture of the application by enforcing input validation.

Google Maps API

Description: The Google Maps API provides a set of web services for embedding Google Maps into web pages. It allows for the addition of maps, geolocation, and location-based services to the front end of our application.[6]

Justification: It enhances our application by providing users with intuitive map interfaces and location-based functionalities.

Licensing: Google Maps API is available under a pay-as-you-go pricing model. The API has a free tier with limited usage and requires an API key for access. While the API is generally free for smaller applications, its cost can become a factor as usage scales.

Impact: Integrating the Google Maps API significantly enriches the user experience by providing interactive maps and location services.

Implementation challenges

During the implementation, the team experienced various challenges, ranging from front-end issues, back-end issues and integration issues. Those issues are caused by many reasons. The first one is the inefficient real-world software programming experience. Most of team members did not have experience of programming a large web software before, and the technology stack (i.e. MERN stack) was never used by all members. The second one is the inefficient communication between backend and frontend members. Those factors resulted in many issues during implementation, some of which were solved successfully, while the others affected the overall software quality to some degree.

Backend Challenges

1. The first challenge for backend members was database consistency. Although the code could be managed by github, a traditional database deployed locally is difficult for different members to code and test, while multiple databases would result in data inconsistency. The team solved this issue by using MongoDB atlas, which is the cloud version of MongoDB database, enabling all members to code with a consistent database. Also, the cloud database is easy to set up and use, which relieved the difficulties during coding caused by inefficient web application coding experience.

2. During backend implementation, the encryption during user authentication implementation was the second challenge faced by backend members. To guarantee data safety, the team decided to solve authenticated passwords rather than raw passwords in the database. This function increased coding difficulty, and made database management harder because we can not directly retrieve real passwords from the database. The team solved this issue by watching online tutorials teaching the encryption with MERN stack, and recording test data with a separate text file.

Frontend Challenges

1. One of the most challenging tasks in the front-end is to understand and utilize Redux and React as a beginner front-end developer. As Redux is a great way and framework which has a great application in real world web application, therefore, we could not use work around but rather try to learn and get used to it. After number of tutorials and example, Redux is implemented and used along all function in the front-end
2. Moreover, building a dynamic page such as a Task detail page where there are multiple functions (User stories) that change based on user access is quite challenging. This is due to designing a page that is a common path to other user stories. Therefore the page itself is loaded with tons of functions and code that are buggy to debug and navigate around. This problem is not solved as it can be solved due to experienced developers and how they can break down the component.
3. Other minor challenges are there, however, one problem that occurred the most, that slows down the front-end development is the consistency of the back-end logic. Where different types of response are returned and different bugs are not catched during processing. One of our solutions is to communicate this with the back-end team to help reduce the bugs. The solution worked but in emergency case, the front end would directly modified the server and later notify the back-end

Other Challenges

1. The communication between frontend and backend was one of the largest problems. The lack of communication resulted in inconsistency of frontend requirement and backend implementation, and plenty of work was redone to make the APIs consistent. The team overcame this challenge by utilizing Jira, and maintaining an API document shared by all members, which accelerated the sprint 3 coding speed.

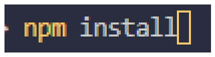
Documents

How to Build, Setup and Configure

First, before running the system, you need to ensure that nodejs v16 has been downloaded. Please uninstall the current nodejs and install nodejs v16 if you don't have nodejs v16. You first need to download the dependencies, then run the back-end code in the back-end folder, and finally run the front-end code in the front-end-modified folder.

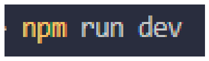
Back-end

You can use the following command to get all the dependencies for back-end:



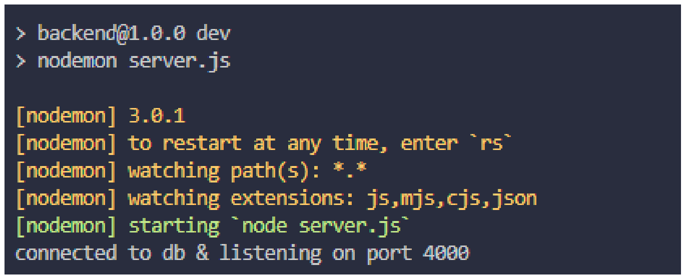
```
npm install
```

Then you need to run the back-end code by using the following command:



```
npm run dev
```

After that, you should see the following output which means that the back-end ran successfully:



```
> backend@1.0.0 dev
> nodemon server.js

[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
connected to db & listening on port 4000
```

Front-end

The front-end requires Node v16.16 to run.

Therefore, installing Node v16.16 is necessary.

Once installed, you can navigated to ./front-end-modified and use the following command to get all the dependencies for front-end:

```
npm install
```

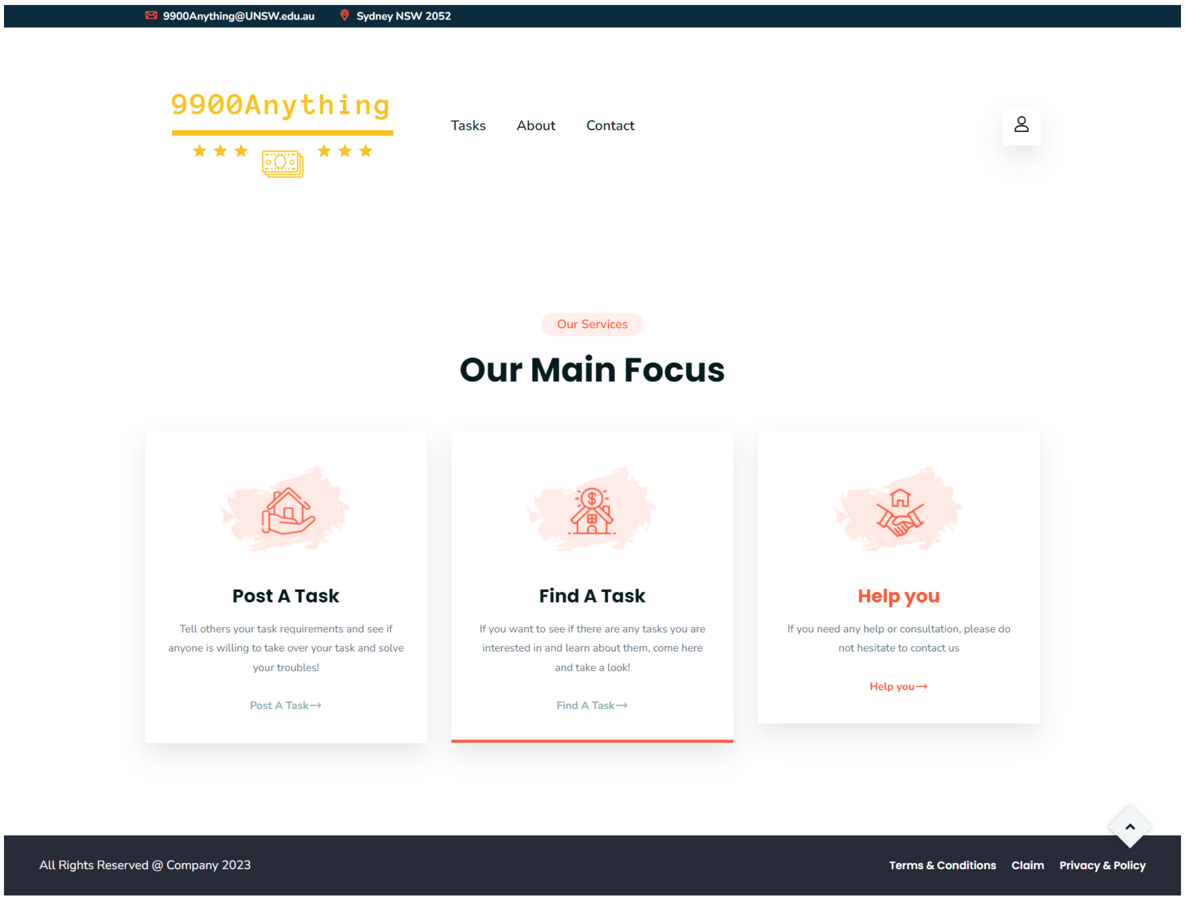
Then you need to run the front-end code by using following command:

```
npm start
```

After that, you will see the following output:

```
> quarter@0.1.0 start
> react-scripts start
```

When the current code runs normally, it will automatically open your browser and pop up the index page of the web page created by our team:



Tell others your task requirements and see if anyone is willing to take over your task and solve your troubles!

If you want to see if there are any tasks you are interested in and learn about them, come here and take a look!

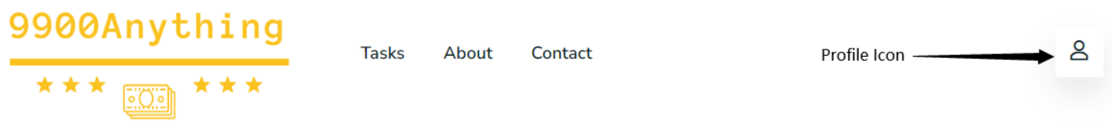
If you need any help or consultation, please do not hesitate to contact us

Use system and functionalities

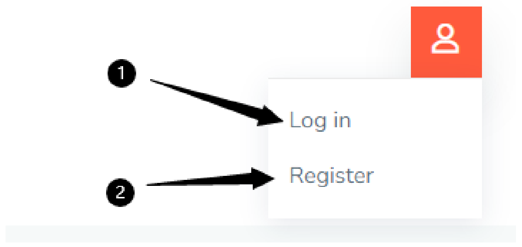
In order to make it easier for you to master the use of the system and its functions, the following is a detailed usage guide. To improve readability, we have divided the guide into the following main sections:

I. Signup and Login

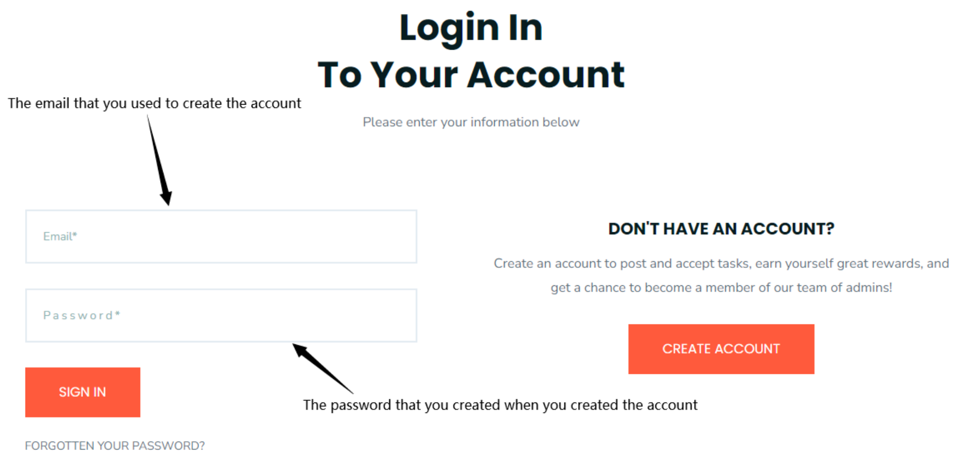
Before using this system, you first need to login as a user. If you don't have an existing account, you also can register for a user account. At the top of every page of this system, you can see the following picture.



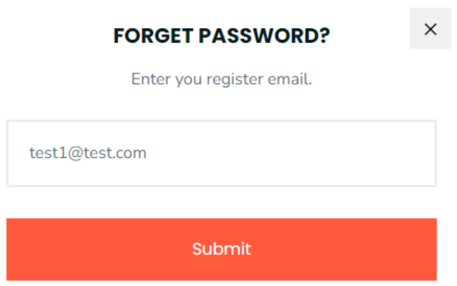
When you move your mouse to the Profile icon in the upper right corner, a small box will pop up, which includes login and register.



1) For logging in the existing account, you just need to click the "Log in" button. After that, the page will be automatically redirected to the following login page.

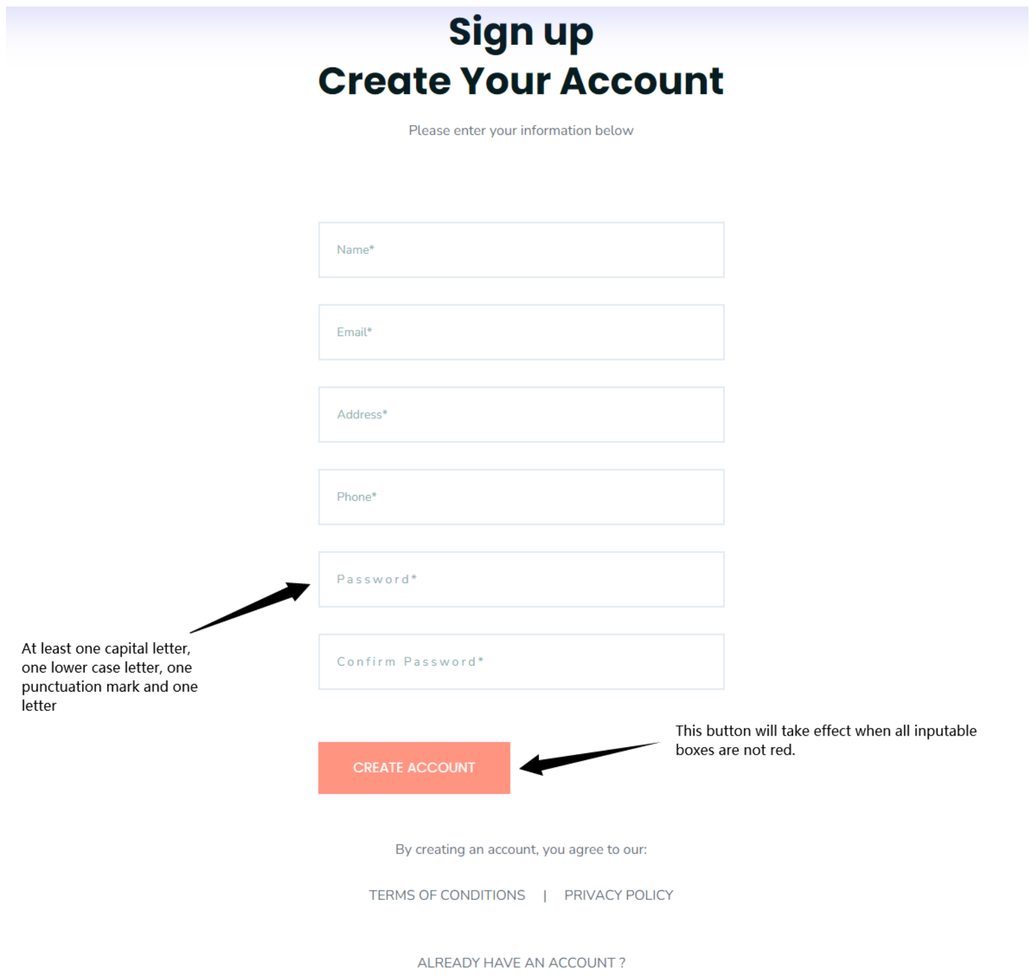

Login In
To Your Account
The email that you used to create the account
Please enter your information below
Email*
Password*
SIGN IN
FORGOTTEN YOUR PASSWORD?
The password that you created when you created the account
DON'T HAVE AN ACCOUNT?
Create an account to post and accept tasks, earn yourself great rewards, and get a chance to become a member of our team of admins!
CREATE ACCOUNT

If you enter an incorrect password, "Incorrect password" will be displayed. You can choose to click on “FORGOTTEN YOUR PASSWORD?” at the bottom left and the page will automatically pop up with a window like the one below.



If you enter an incorrect or non-existent email address, "Incorrect email" will be displayed. You can also choose to register your account by clicking "CREATE ACCOUNT" on the right hand side or the Profile icon in the upper right corner as mentioned above.

2) Regardless of which of the previously mentioned registration methods you used to create your account, you will be redirected to the "Sign up" page. The image below shows what you need to fill in the "Sign up" page.



Sign up
Create Your Account

Please enter your information below

Name*

Email*

Address*

Phone*

Password*

Confirm Password*

At least one capital letter, one lower case letter, one punctuation mark and one letter

CREATE ACCOUNT

This button will take effect when all inputable boxes are not red.

By creating an account, you agree to our:

TERMS OF CONDITIONS | PRIVACY POLICY

ALREADY HAVE AN ACCOUNT ?

On this page, when your input does not apply to this input box rule, the input box border will be displayed in red. In this case:

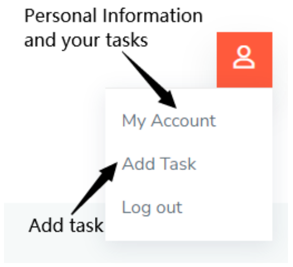
- The email address must be in the format of "Part of domain@domain", otherwise it will be considered invalid.
- The password section is set to a certain level of strength, requiring at least one uppercase letter, one lowercase letter, one punctuation mark, and one number.
- The content of the "Confirm Password" must be identical to the content of the "Password".

The "CREATE ACCOUNT" button will take effect when no input box is red. Click this button to complete the registration procedure.

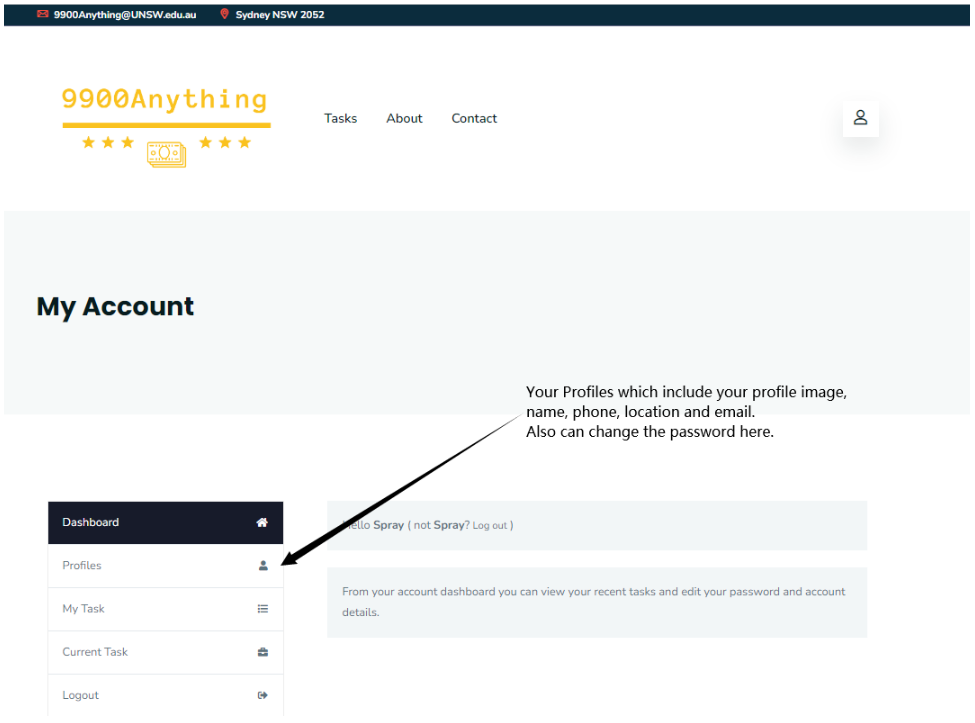
Once registration is complete, the page will jump to the login page again, repeat the first step of the login procedure to complete the account login.

II. Profile Management

After logging in, when you place the mouse cursor on the upper right corner button again, it will be the same as the picture below:



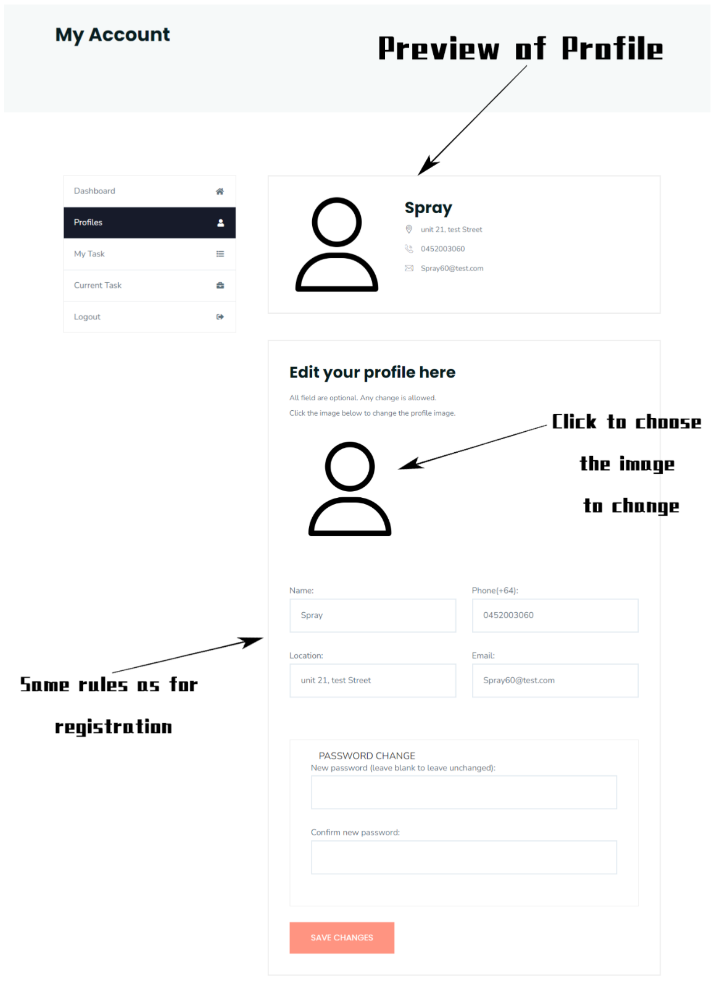
When you click on "My Account", the page will be redirected to the following page:



Hello Spray (not Spray? Log out)

From your account dashboard you can view your recent tasks and edit your password and account details.

Click on the “Profiles” on the left-hand bar, you will be redirected to “My Account”, which is the profile management page.



Spray
unit 21, test Street
0452003060
Spray60@test.com

Edit your profile here

All field are optional. Any change is allowed.
Click the image below to change the profile image.

Name: Spray
Phone(+64): 0452003060
Location: unit 21, test Street
Email: Spray60@test.com

PASSWORD CHANGE
New password (leave blank to leave unchanged):
Confirm new password:

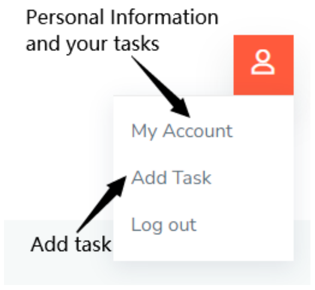
In this page,you can see the profile preview on the top. Below that you are able to modify your profile details including:

- User image (By click on the current image)
- Name
- Phone
- Location
- Email
- Password

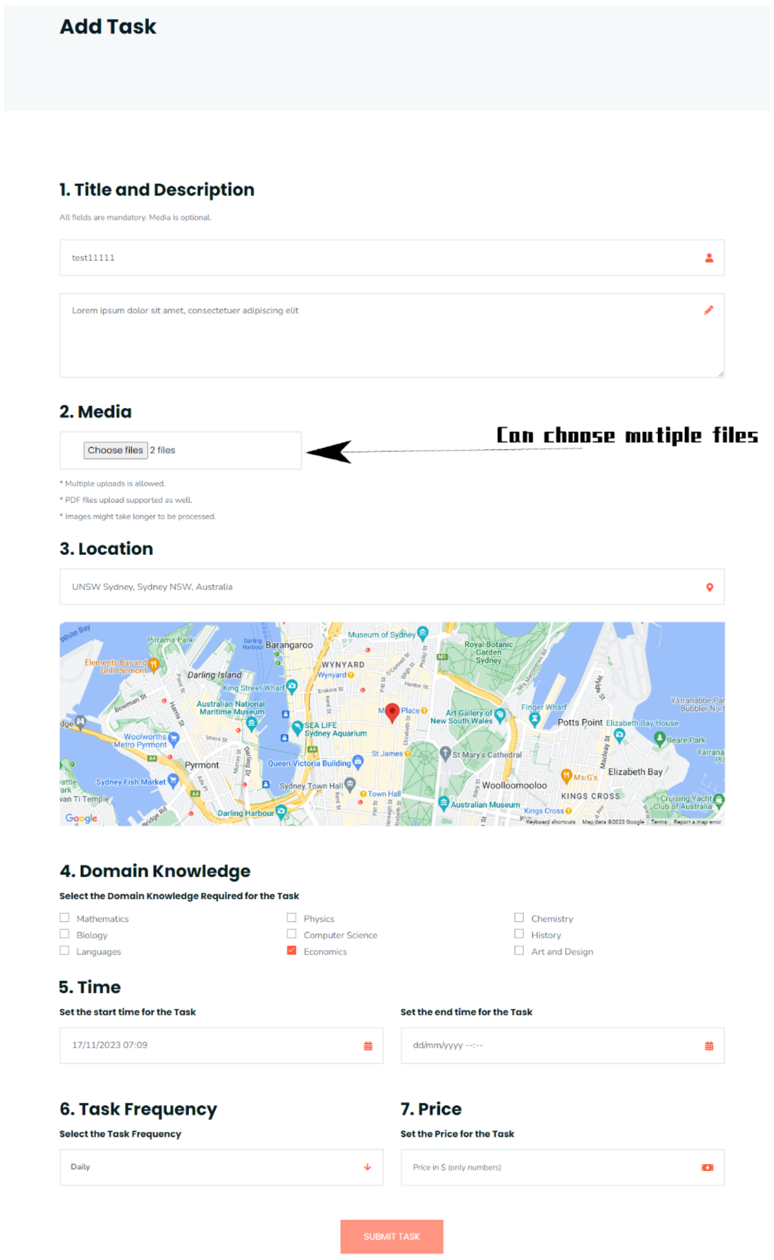
*To modify the password, make sure the new password still follows the safety requirement which is at least one capital letter, one lower case letter, one punctuation mark and one letter.

III. Task Management

When you want to create your task, you can use the button in the upper right corner. You need to click the "Add Task" button to create a new task.



After that, the page will automatically jump to the page below. You need to fill in the required fields in the following page to successfully create a task.



Add Task

1. Title and Description

All fields are mandatory. Media is optional.

test11111

Lorem ipsum dolor sit amet, consectetuer adipiscing elit

2. Media

Choose files 2 files

* Multiple uploads is allowed.
* PDF files upload supported as well.
* Images might take longer to be processed.

3. Location

UNSW Sydney, Sydney NSW, Australia

4. Domain Knowledge

Select the Domain Knowledge Required for the Task

☐ Mathematics	☐ Physics	☐ Chemistry
☐ Biology	☐ Computer Science	☐ History
☐ Languages	☑ Economics	☐ Art and Design

5. Time

Set the start time for the Task: 17/11/2023 07:09

Set the end time for the Task: dd/mm/yyyy --:--

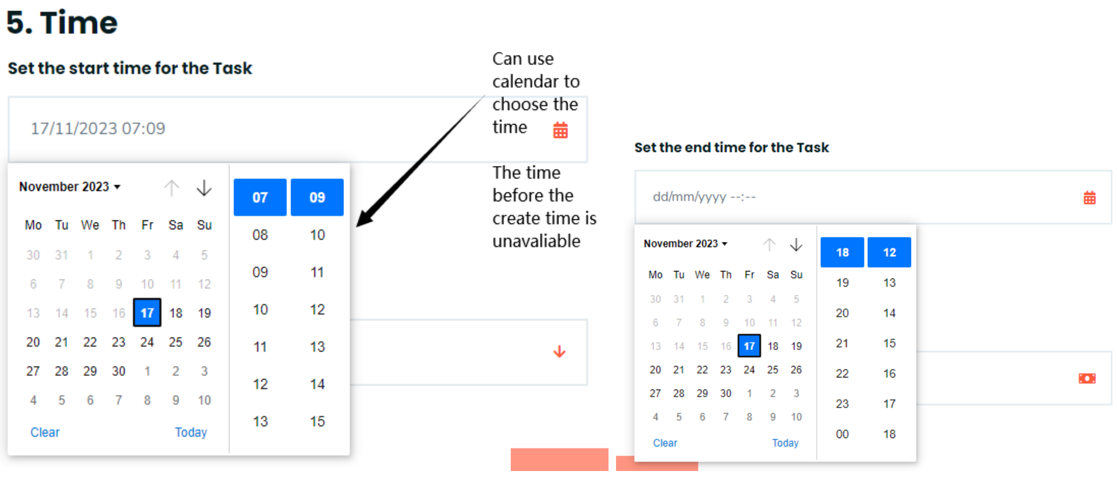
6. Task Frequency

Select the Task Frequency: Daily

7. Price

Set the Price for the Task: Price in $ (only numbers)

In the "Time" section, you can click on the input box to bring up a calendar, you can use the calendar to select your desired start time more easily and quickly, but it is important to note that the start time is not available until you create the time. The end time is the same as the start time.



5. Time

Set the start time for the Task

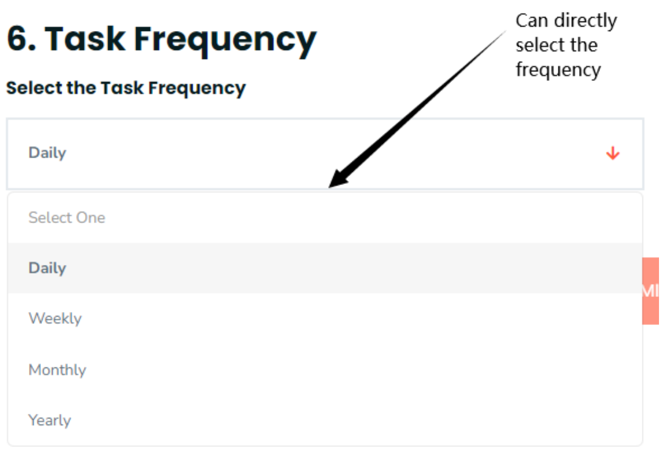
17/11/2023 07:09

Set the end time for the Task

dd/mm/yyyy --:--

Can use calendar to choose the time
The time before the create time is unavaliable
November 2023
Mo Tu We Th Fr Sa Su
Clear
Today

In the "Task Frequency" section, you can also directly select a frequency to use from the drop-down list that appears when you click on the input box.



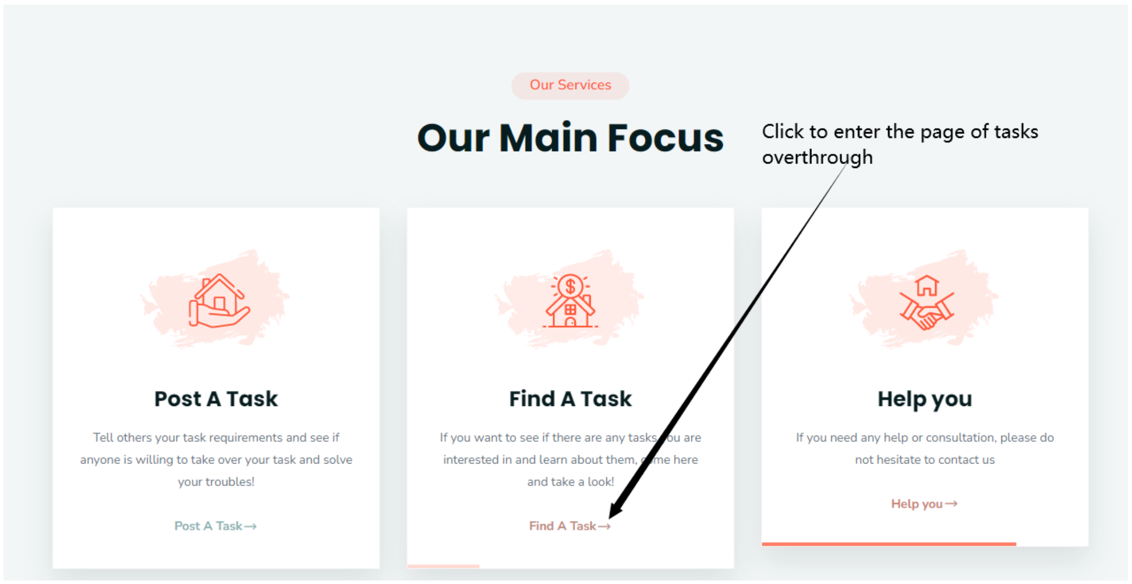
6. Task Frequency

Select the Task Frequency

Daily

After completing all the above steps and clicking on the "SUBMIT TASK" button, you will successfully create a new task.

After that the page will then automatically jump to the index page. You can then access the “Tasks” page via the "Find A Task" button on the index page, or via the "Tasks" button on the top navigation bar.



Our Services

Our Main Focus

Click to enter the page of tasks overthrough

Post A Task

Tell others your task requirements and see if anyone is willing to take over your task and solve your troubles!

Post A Task→

Find A Task

If you want to see if there are any tasks you are interested in and learn about them, come here and take a look!

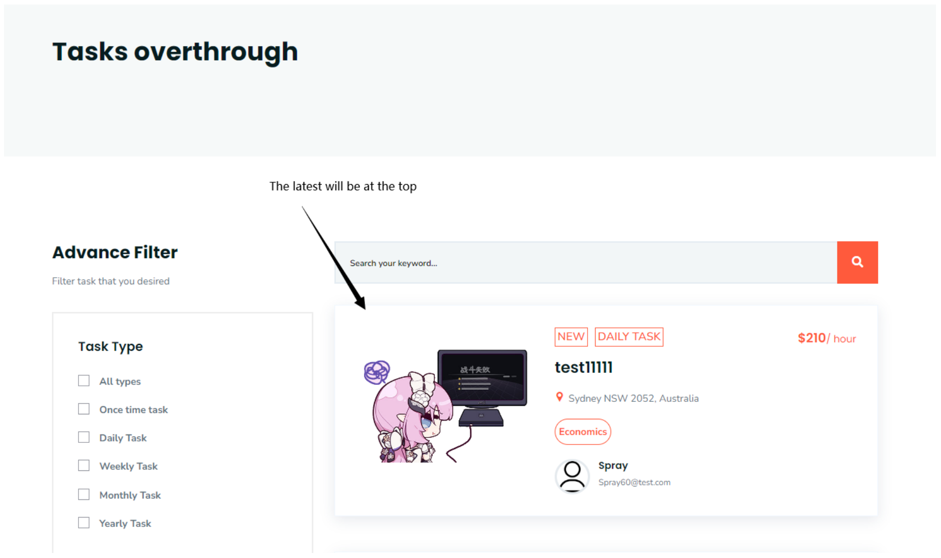
Find A Task→

Help you

If you need any help or consultation, please do not hesitate to contact us

Help you→

As you can see, the task you just created appears at the top of the task list, i.e., the task list is sorted by the time it was created. The most recently created task will appear at the top of the list.



Tasks overthrough

The latest will be at the top

Advance Filter

Filter task that you desired

Search your keyword...

Task Type

- All types
- Once time task
- Daily Task
- Weekly Task
- Monthly Task
- Yearly Task

NEW DAILY TASK $210/ hour

test11111

Sydney NSW 2052, Australia

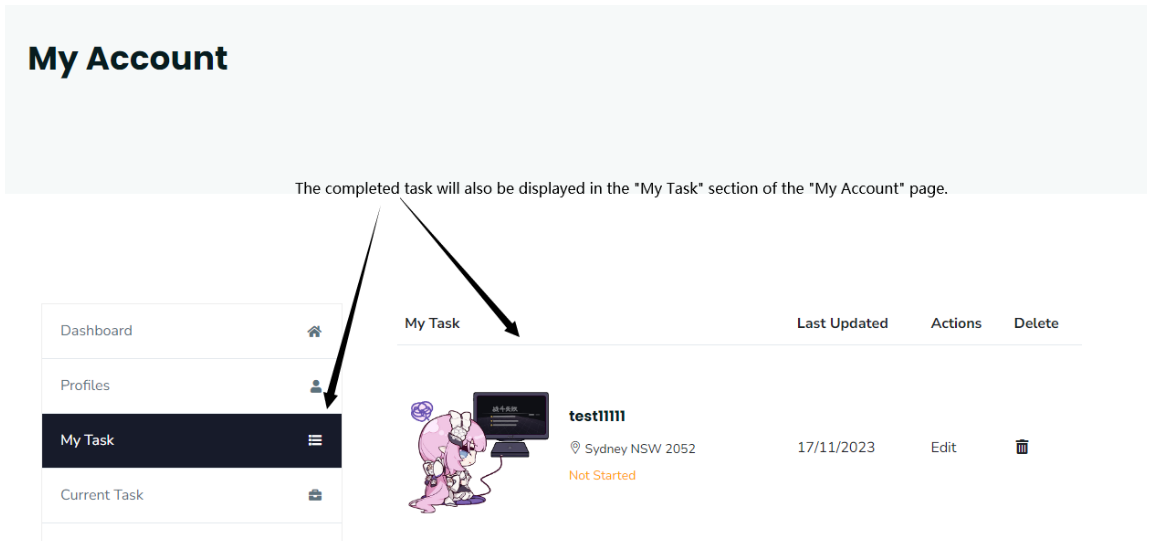
Economics

Spray

Spray60@test.com

Similarly, all of the tasks you have created will be displayed in the "My Task" section of the "My Account" page.

If you want to quickly find all the tasks you have posted yourself, you can find them in that section.



My Account

The completed task will also be displayed in the "My Task" section of the "My Account" page.

My Task	Last Updated	Actions	Delete
test11111 Sydney NSW 2052 Not Started	17/11/2023	Edit	🗑

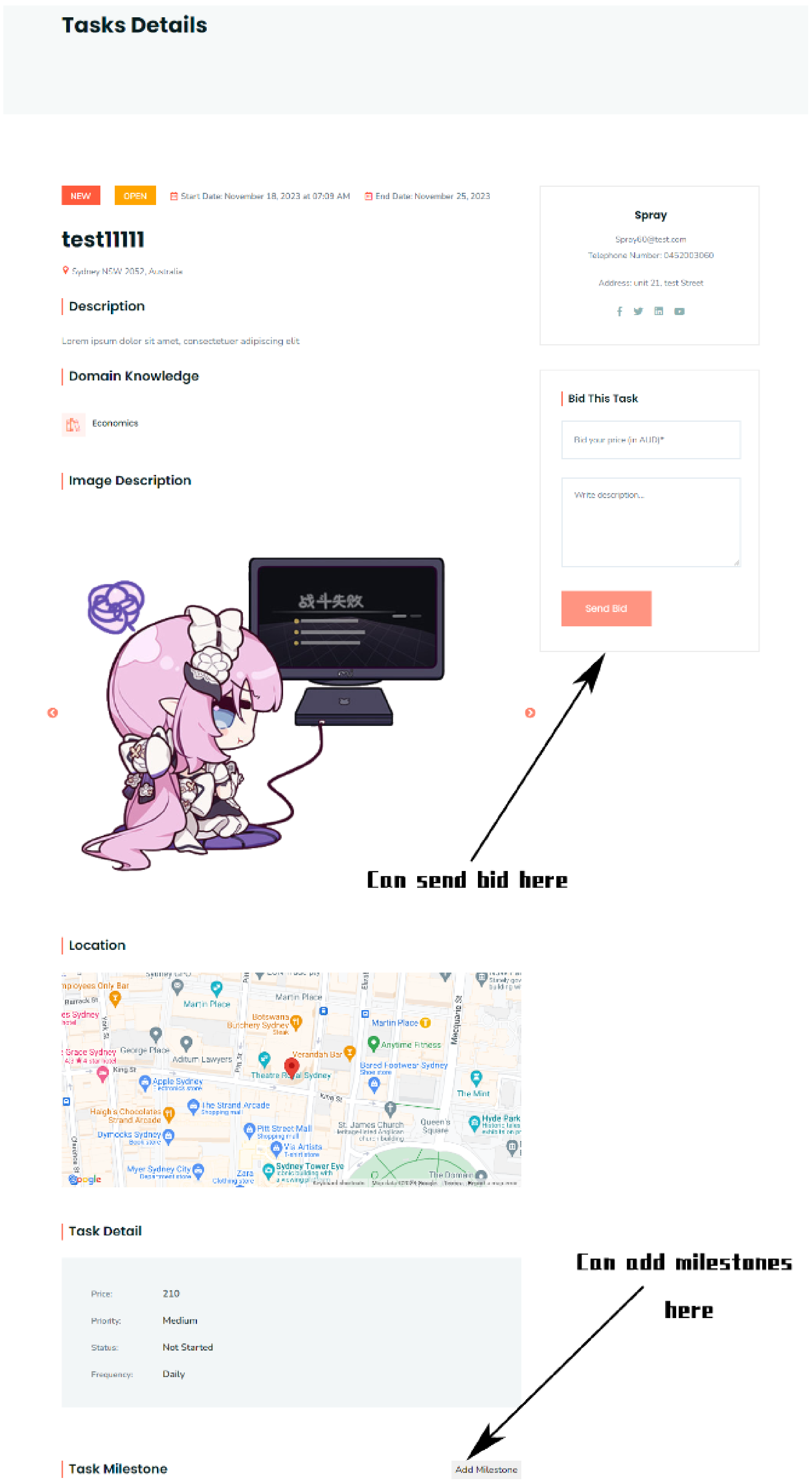
By clicking on a task on either of the two pages above, either the creator of the task or a user will be able to see all the information related to that task, including but not limited to:

- Title
- Description
- Domain Knowledge
- Image Description
- Domain Knowledge Image Description
- Task Detail
- Milestones
- Comments

The "Bid This Task" section is visible only to users who are not the creator of the task.

Although the creator does not have this section, no one can create or edit milestones except the creator and the service tasker whose bid was accepted by the creator.

The page “Tasks Details” is shown below:



Tasks Details

NEW OPEN Start Date: November 18, 2023 at 07:09 AM End Date: November 25, 2023

test11111

Sydney NSW 2052, Australia

Description

Lorem ipsum dolor sit amet, consectetuer adipiscing elit

Domain Knowledge

Economics

Image Description

Location

Task Detail

Price:	210
Priority:	Medium
Status:	Not Started
Frequency:	Daily

Task Milestone

Add Milestone

Spray

Spray60@test.com

Telephone Number: 0452003060

Address: unit 21, test Street

Bid This Task

Bid your price (in AUD)*

Write description...

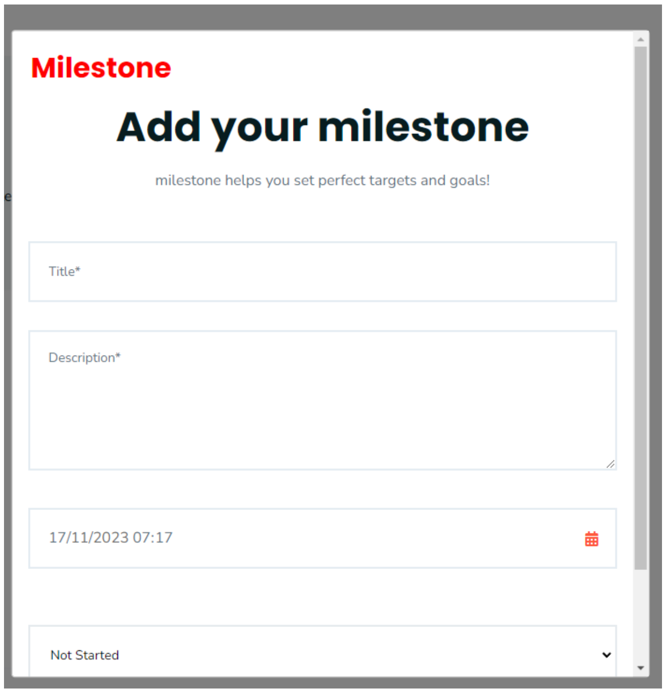
Send Bid

Can send bid here

Can add milestones here

IV. Milestone Management

By clicking on the Add milestone button on task details, you are provided with a form. Fill all fields and submit it to create a new milestone.



Milestone

Add your milestone

milestone helps you set perfect targets and goals!

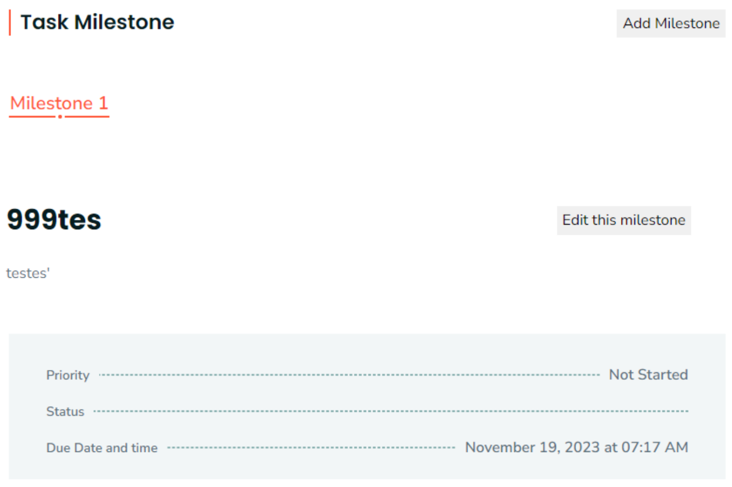
Title*

Description*

17/11/2023 07:17

Not Started

After adding, the milestone will display for you and the tutor who accepted the task.



Task Milestone

Add Milestone

Milestone 1

999tes

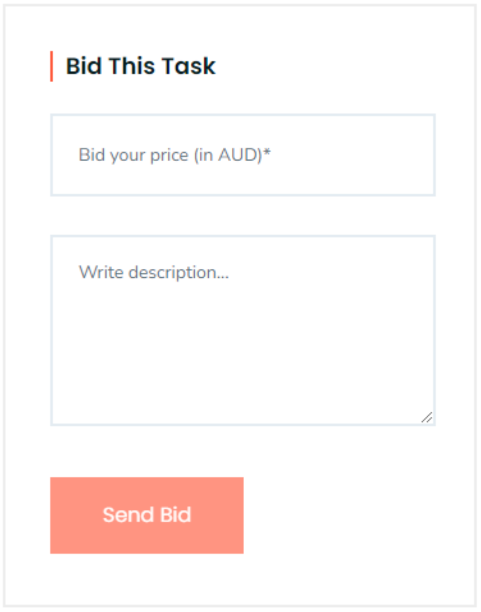
Edit this milestone

testes'

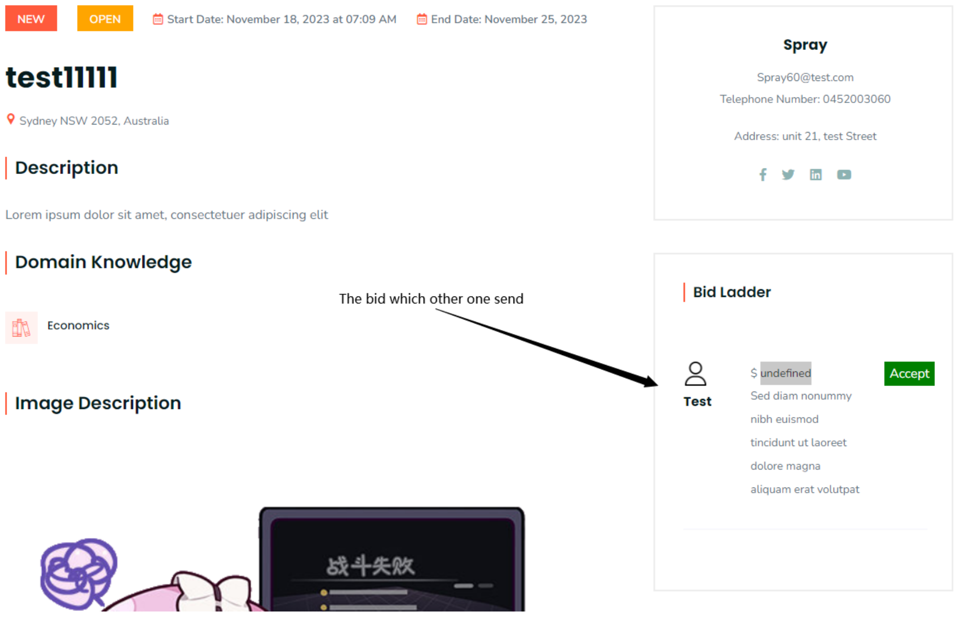
Priority	Not Started
Status	
Due Date and time	November 19, 2023 at 07:17 AM

V. Bid Management

On the right-hand side bar of the "Task Details" page, as a tutor, you can bid on this task.



After bidding, the task owner could see the bid just sent.


NEW OPEN Start Date: November 18, 2023 at 07:09 AM End Date: November 25, 2023

test11111

Sydney NSW 2052, Australia

Description

Lorem ipsum dolor sit amet, consectetuer adipiscing elit

Domain Knowledge

Economics

Image Description

Spray

Spray60@test.com

Telephone Number: 0452003060

Address: unit 21, test Street

Bid Ladder

Test

$ undefined

Sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat

Future Work

Due to the time limit and the inefficient experience with large real-world projects, some challenges occurred during the development affected the development speed and software quality, more specifically, some front end functions are not implemented to satisfy the requirements. The future work of the team is to implement a more stable and complete front end to provide 100% and valuable user experience. The parts implemented in back end server but not on front end interface include:

1. Task filter
2. Communication System
3. Payment
4. Bid Acception

References

1) MongoDB Documentation. (2023). MongoDB. https://www.mongodb.com/docs/

2) API Reference 4.x. (n.d.). Express. https://expressjs.com/en/4x/api.html

3) React Reference. (2023). React. https://react.dev/reference/react

4) Bcrypt. (2023, August). Bcrypt. https://www.npmjs.com/package/bcrypt

5) O'hara, C., Nandaa, A., Kemboi, E., & Aissi, S. (2023, August). Validator.Js. Npm. https://www.npmjs.com/package/validator

6) Google Maps Platform Documentation. (n.d.). Google Maps Platform. https://developers.google.com/maps/documentation