Text

Description automatically generated

**FACULTY OF APPLIED INFORMATION TECHNOLOGY**

**Field Of Study: Information Technology**

**Speciality: Programming**

**Nurmukhammed Rakhmet**

**Ali Shaikydin**

**Akzhan Zhailaubay**

No. of student’s record book: w67096,w67677,w67094

**PROJECT**

**Project title: DoWAI**

**Do With AI**

**Rzeszow 2024**

**Project Documentation**

1. Introduction

* 1. Project Overview

This project is a web application that allows users to manage tasks and notes associated with those tasks. Users can log in, add tasks, set priorities, add notes to the tasks, use the help of the AI to add the notes, and perform various operations on their tasks.

* 1. Goals and Objectives

Provide a user-friendly interface for managing tasks.

Implement user authentication to secure user data.

Utilize web technologies for efficient communication with the server.

Allow users to interact with a natural language processing (NLP) model to enhance task management.

* 1. Technology Stack

The project utilizes the following technologies:

Frontend: HTML, CSS, JavaScript, Bootstrap

Backend: Node.js, Express.js

Database: MongoDB

Authentication: JSON Web Tokens (JWT), bcrypt

GPT API: OpenAI GPT-3

2. Architecture

2.1 Overall Architecture

The application follows a client-server architecture. The client-side is implemented using HTML, CSS, and JavaScript. Client side consists of window for the app itself, registration and login page.

Server-side is built with Node.js and Express.js, structured using controllers, routers, models. The data is stored in a MongoDB database.

2.2 Components

Client-side Components:

index.html: html file for the login page.

registration.html: html file for the registration page.

style.css: Stylesheet for the login and registration page.

script.js: Script for handling the login and registration functions and interaction between the server for validations.

app.html, app.css, app.js : Overall implementation of the task-manager application.

calendar.html, calendar.css, calendar.js: Implementation of the calendar component.

...

Server-side Components:

Index.js: main file of the server where the app runs and connections with the database establishes.

controllers/controllers: Handles server-side logic and interactions with the database, and the requests to the gpt API.

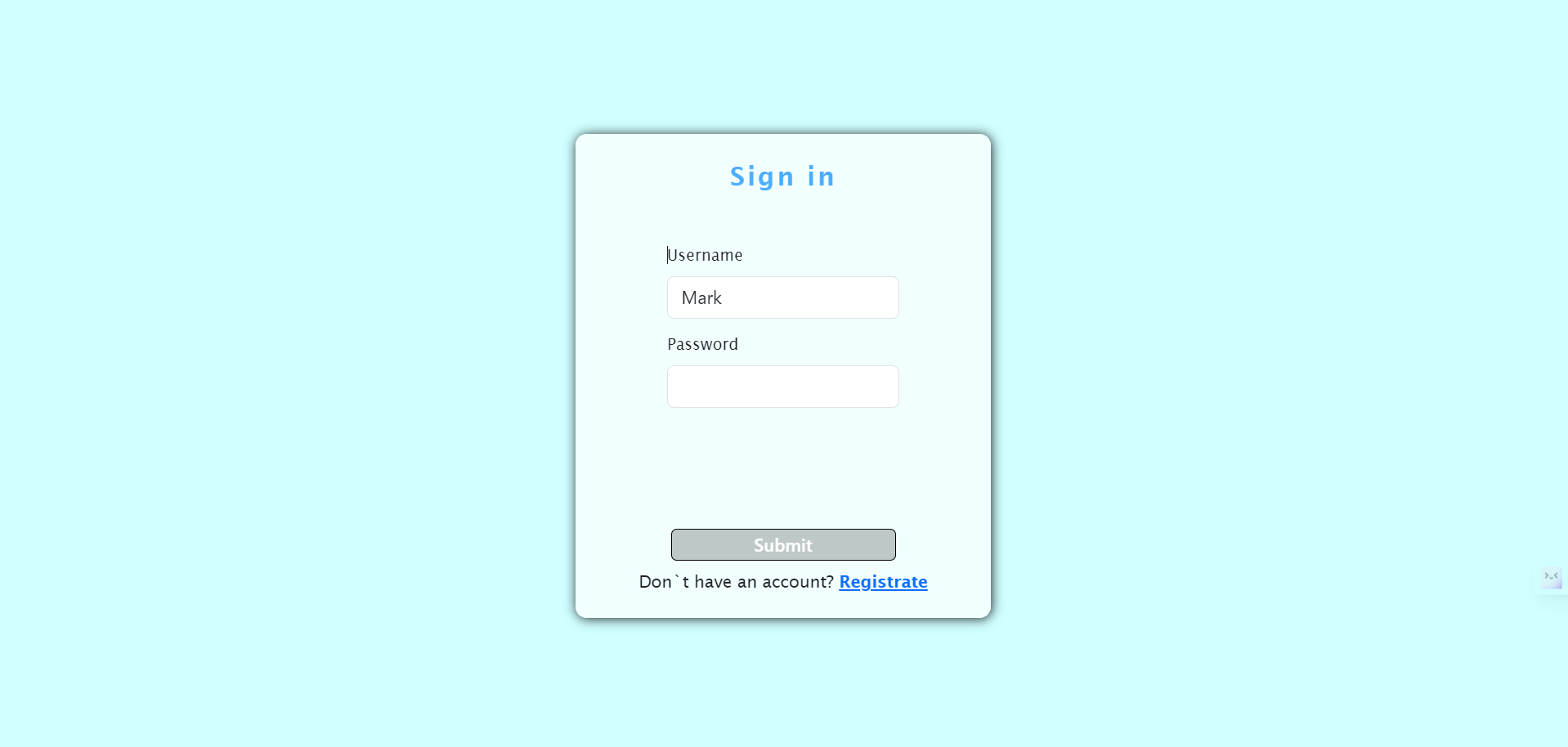
Models/User.js and Task.js: MongoDB models for user and task data.

routers/loginreg.js: the router which imports and applies routes for the functions which are imported from controllers.js.

Middlewares/authMiddleware.js: express`s middleware function which takes responses for a private routes and validates the access of the user. Using JSONWebTokens.

config.js: Configuration file for sensitive information like API keys.

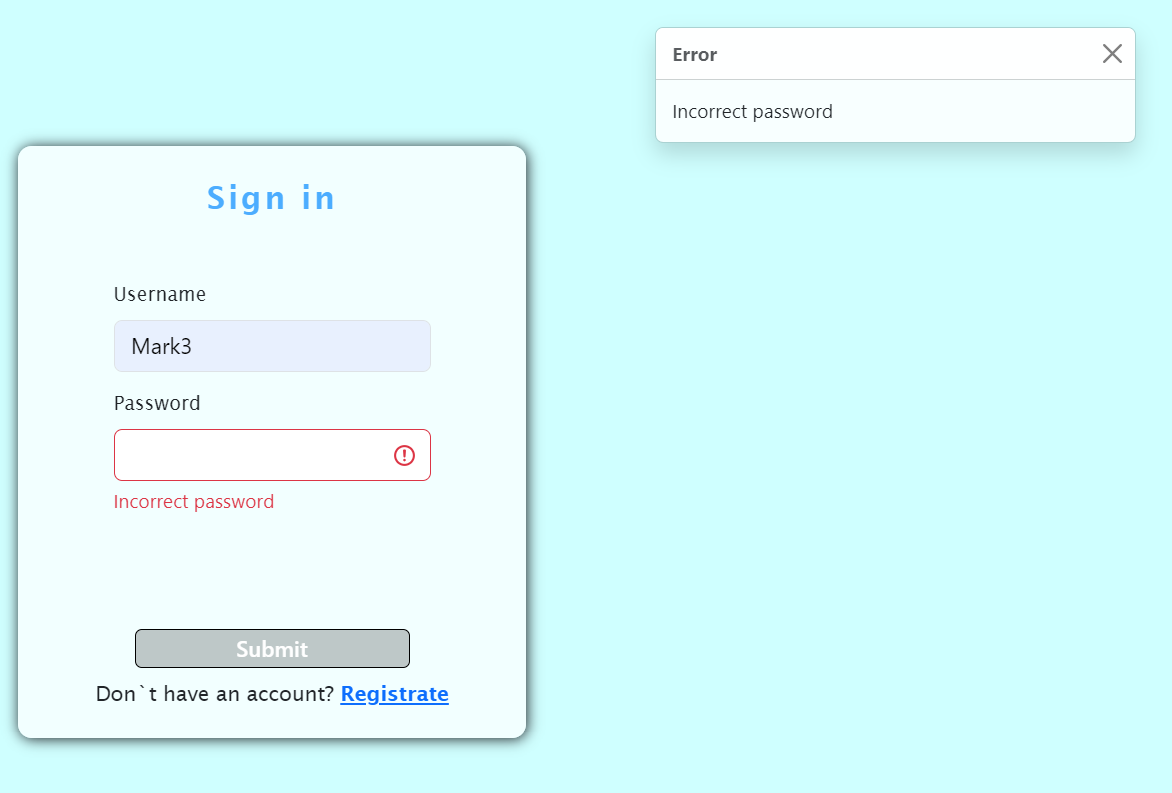
3. Functionality

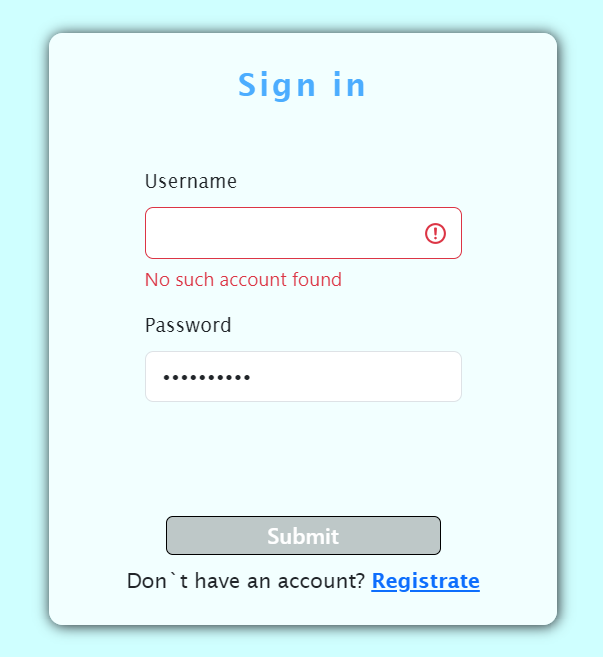
3.1. Login Window:

Possibility to redirect for the registration page

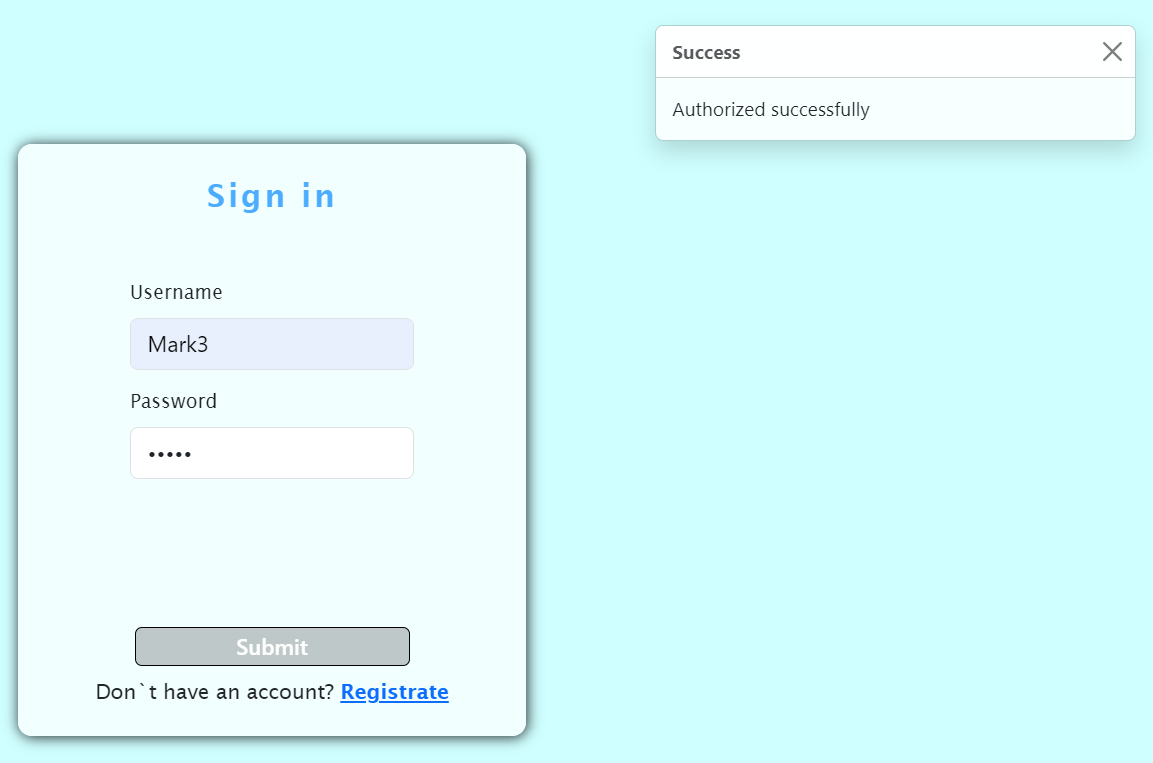


Min-max **html** validation.

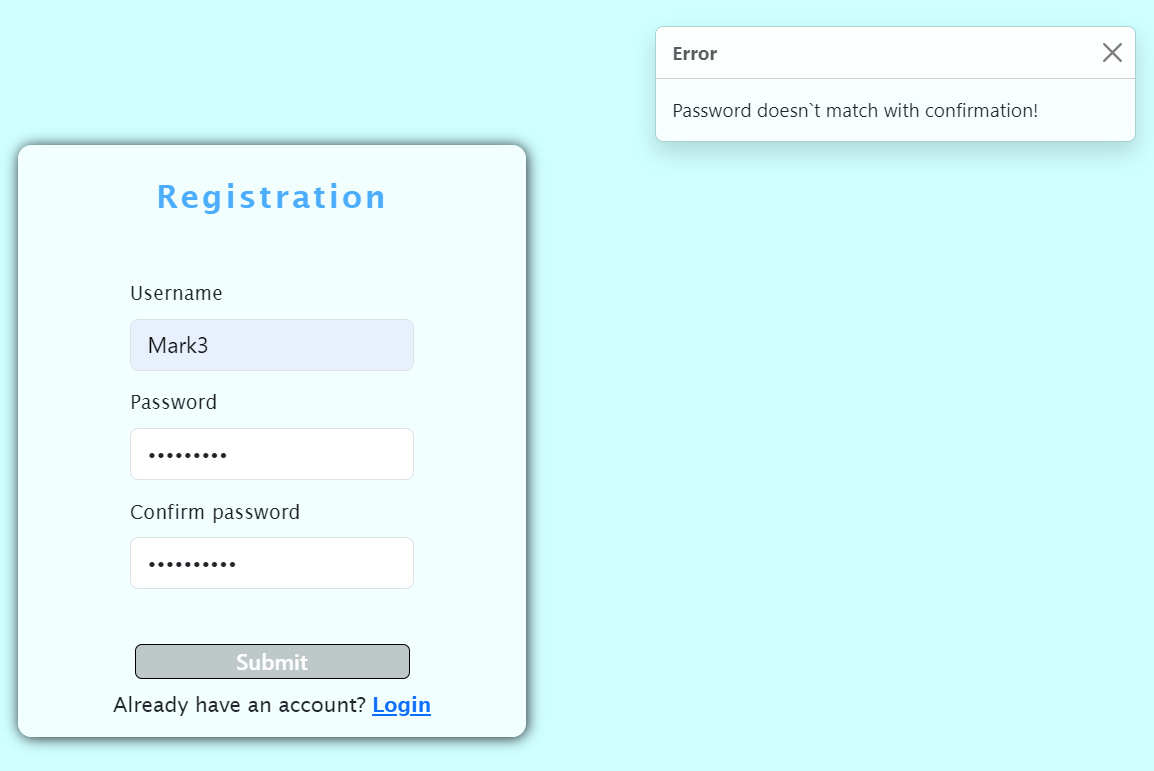




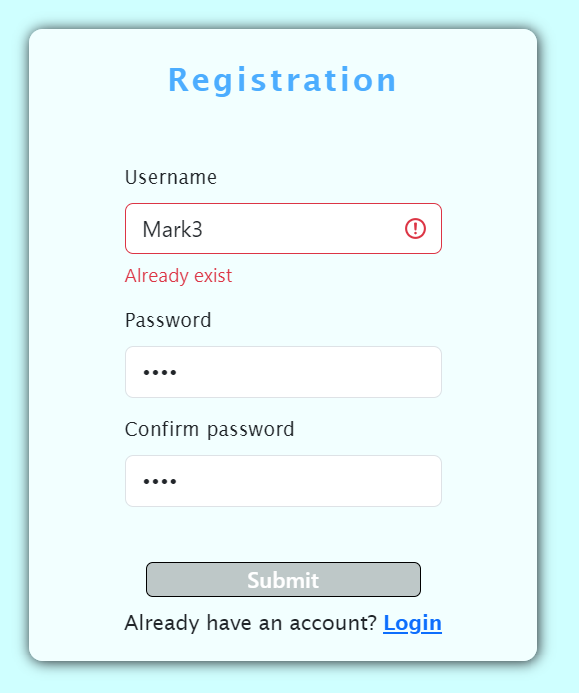
Server side validations (added specific validation mistake messages to show that every case is handled on the server side).



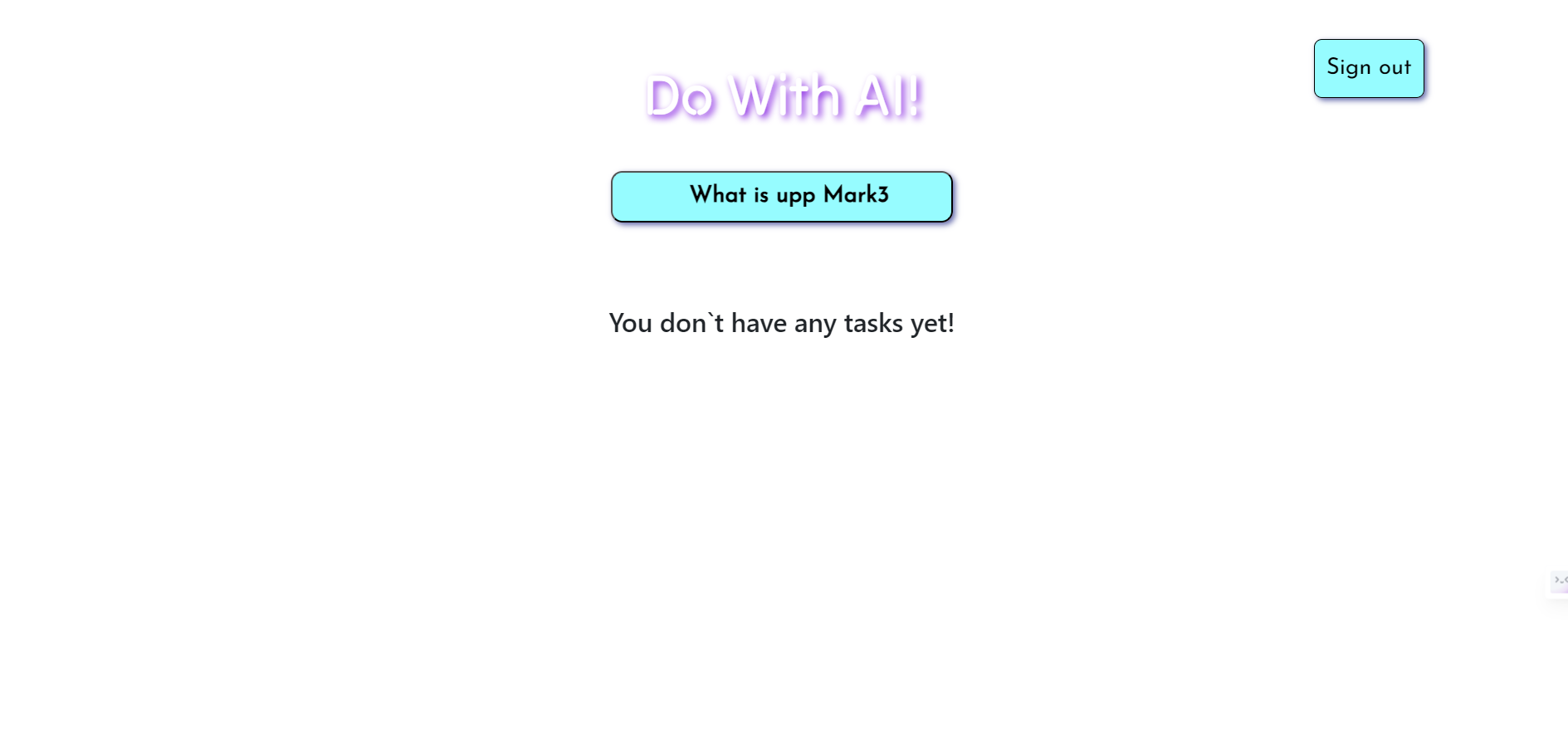
3.2. Registration page:



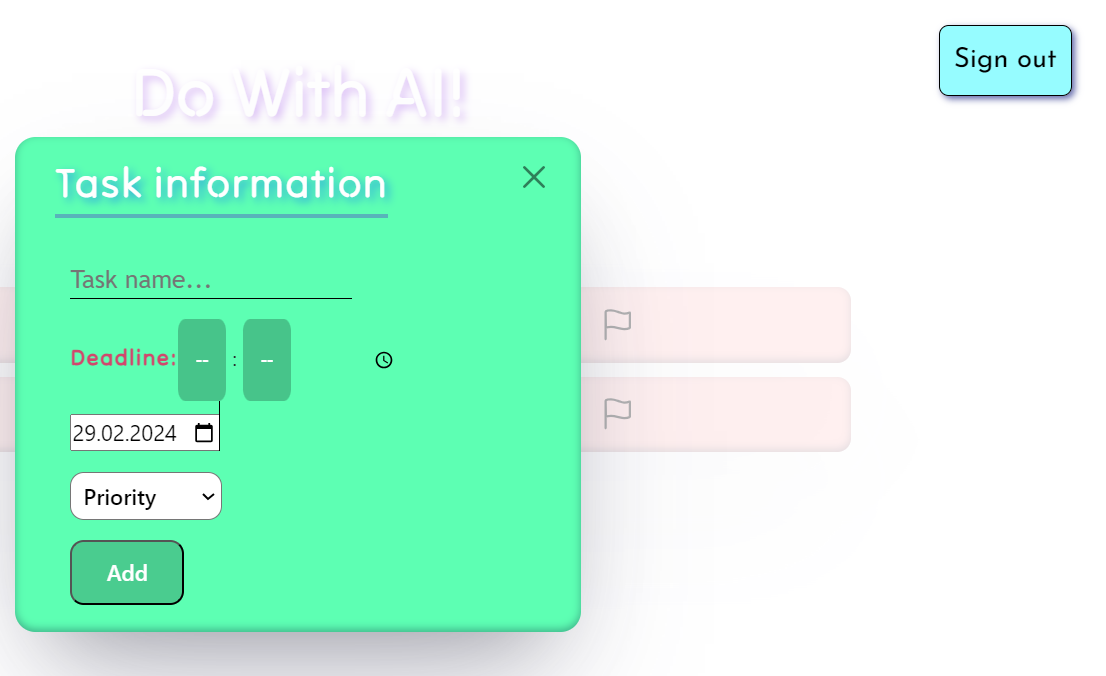
Registered passwords are securely hashed using bcrypt



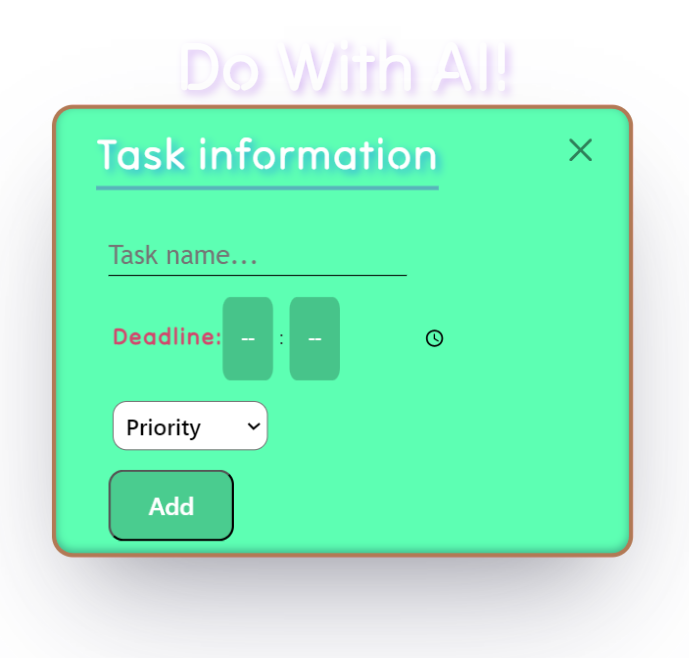
3.3. Application`s page:



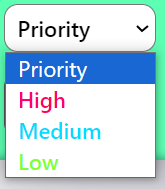
Button to add tasks and button to sign out

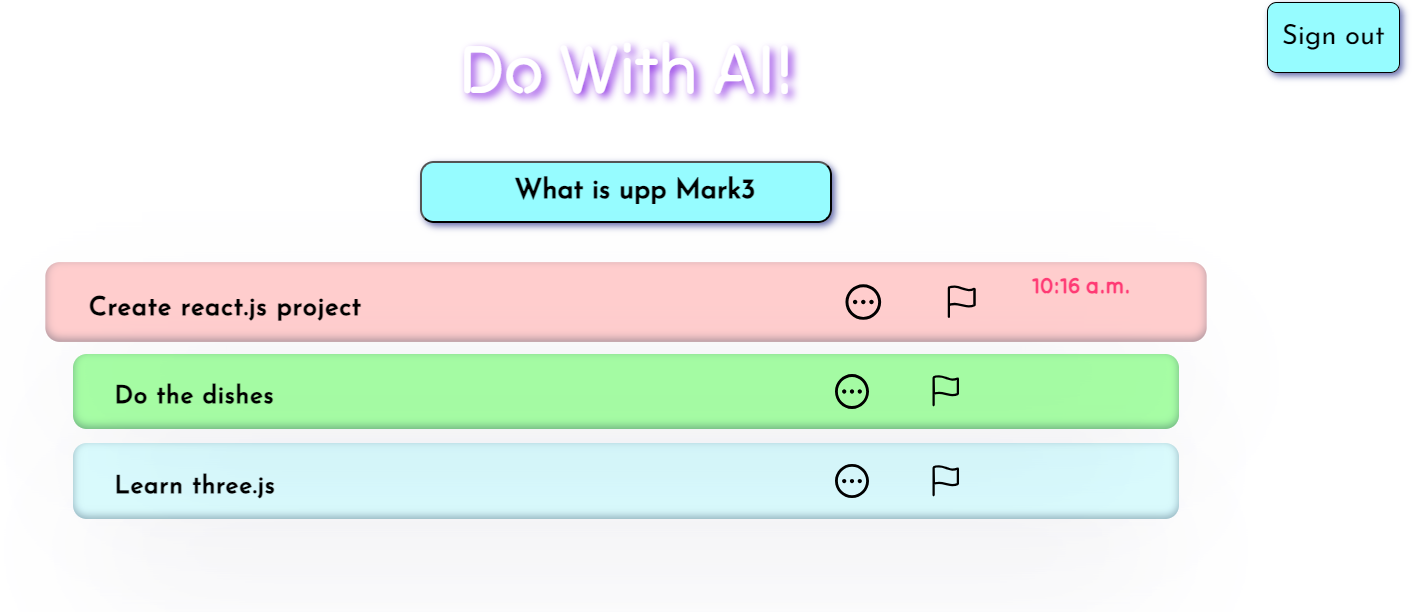


Modal window on clicking the What`s up button. Adds task to a list and requires task name, deadline and priority is optional.

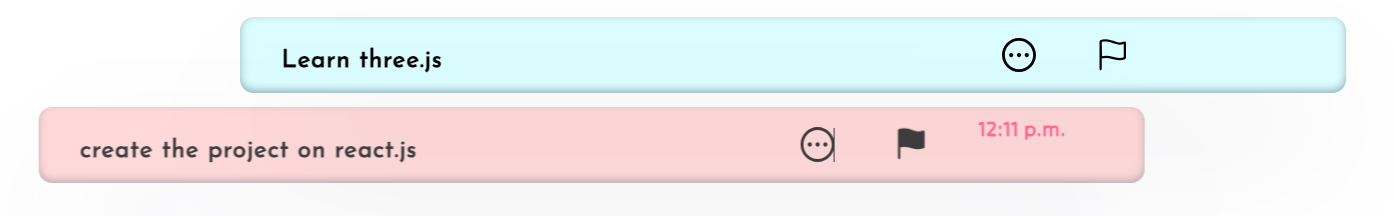


Validation in case if title was not entered.

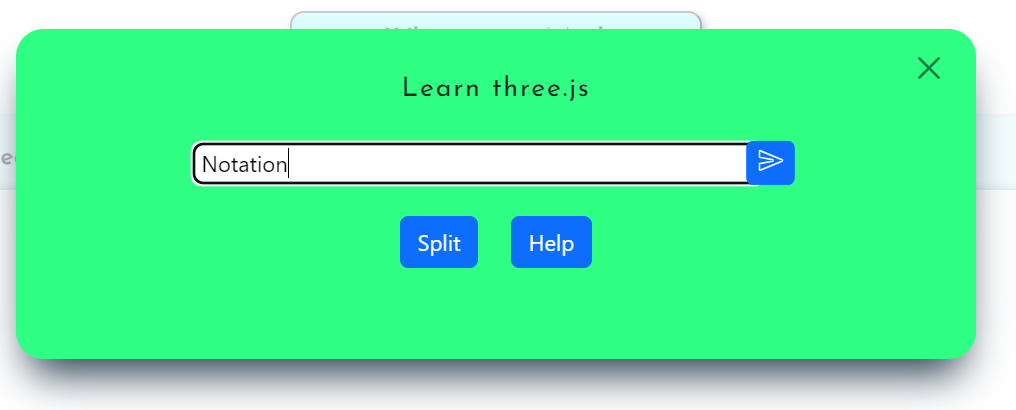




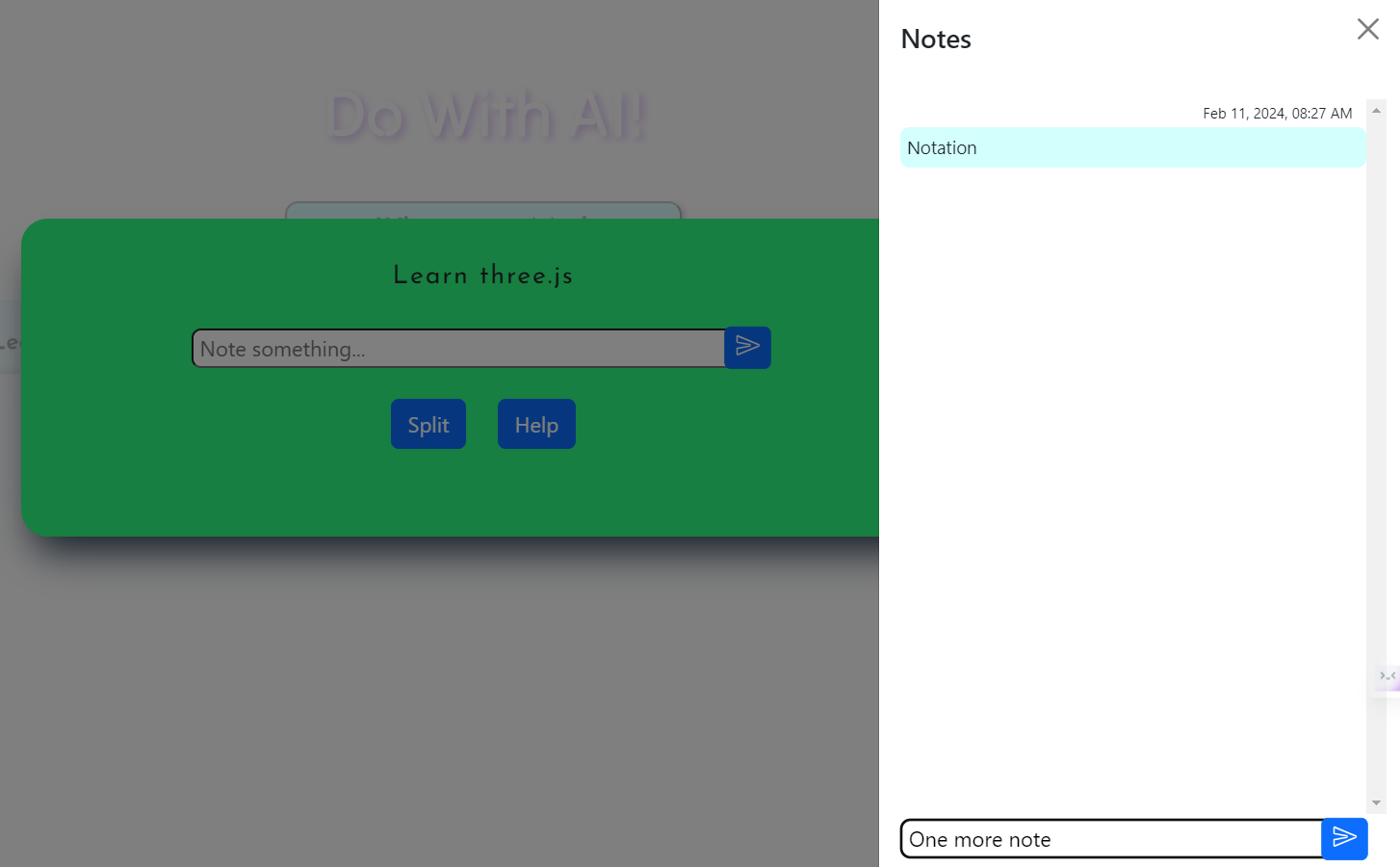
Intuitive color difference based on priority of the task

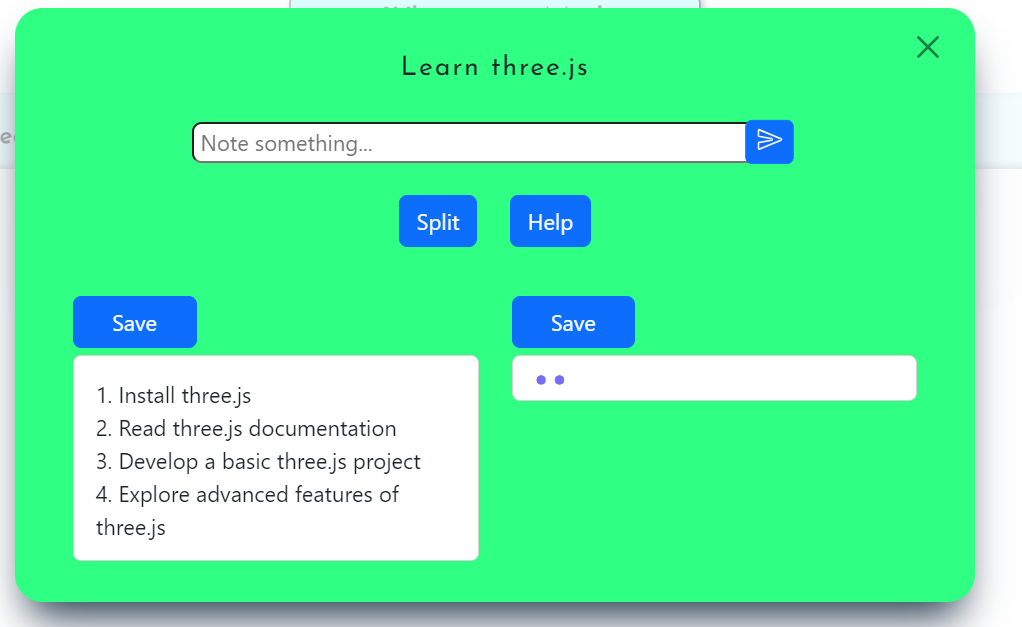


Finish(delete) button animation

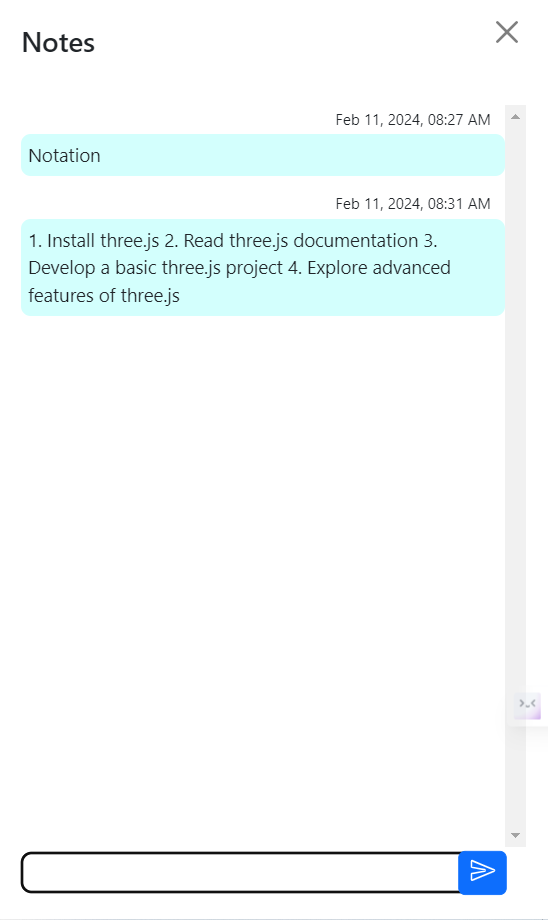


Ability to add notations for each task

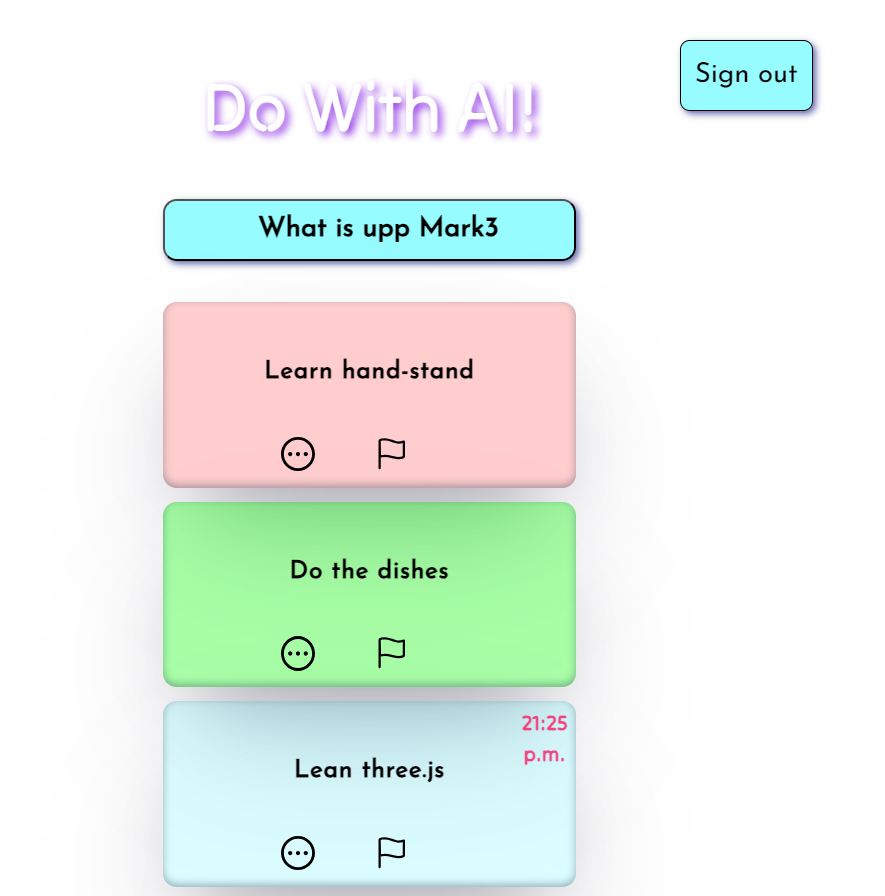
Notes window



Splitting the task based on title via AI.



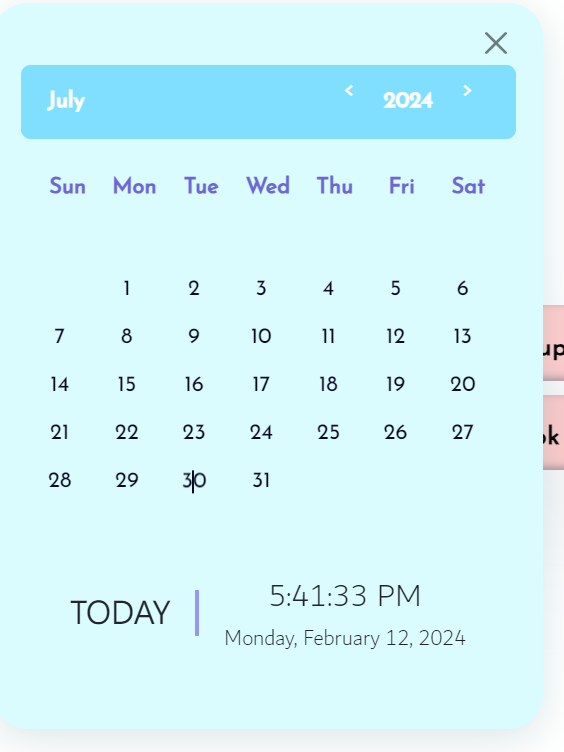
Button “Save” saves the AI`s suggestion to the notations.

3.4. Responsiveness:   


3.5.Calendar:

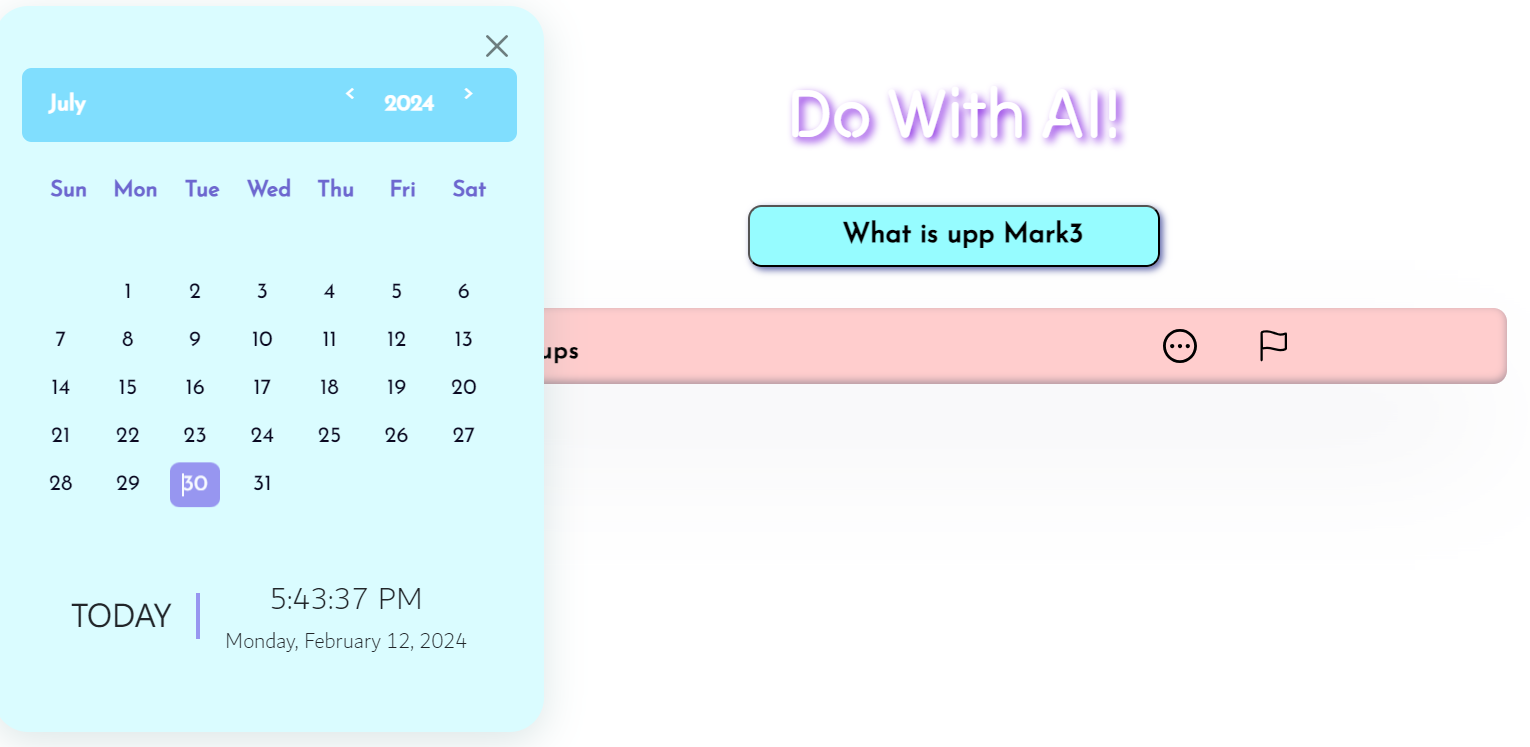


Calendar button on the left top corner

On click:  


The calendar allows to sort the tasks based on the date to which they were assigned.

At the bottom it shows the current date.



4. Installation and Configuration

To set up the project locally, follow these steps:

* Clone the repository.
* Install Node.js and MongoDB.
* Run npm install to install project dependencies.
* Configure environment variables (e.g., API keys):
* Connect MongoDB user.
* Connect your gptAPI-key.

5. Getting Started Guide

To start using the application:

Open the project in a code editor.

Run the server using npm run dev script.

Access the application through the localhost:3000 address.

6. Development

6.1. Client side:

**Initialization Function**:

init: Fetches data from the server ("/main" endpoint) asynchronously, updates the UI, and stores task information in local storage.

**Event Handlers:**

addButton: Handles the "click" event for adding tasks.

more: Handles the "click" event for expanding task descriptions.

**UI Manipulation Functions:**

taskManipul: Creates the html code based on inputted parameters to the function.

refreshNotes: Refreshes the displayed notes in the UI.

**Server Interaction Functions:**

sendData: Sends task data to the server when adding one.

toDel: Deletes tasks by interacting with the server.

**Helper Functions:**

formatDate: Formats dates based on specified options.

fetchData: Fetches data from the server based on provided values and options.

**User Action Functions:**

finish: Marks a task as finished.

split: Splits task details and fetches additional information.

addNode: Adds notes to tasks.

saveNote: Saves notes to tasks.

**User Interface Elements:**

addButton, modalWindow, main, descripted, more, taskAdButton, tasks: References to various HTML elements.

**Session Management and Miscellaneous Functions:**

logOut: Logs the user out by clearing local storage.

toClose: Common close button handler.

**Calendar management Functions:**

isLeapYear(year):

Determines whether a given year is a leap year.

Returns true if the year is a leap year, and false otherwise.

getFebDays(year):

Uses isLeapYear to get the number of days in February for a given year.

Returns 29 for a leap year and 28 for a non-leap year.

generateCalendar(month, year):

Generates a calendar for a specified month and year.

Dynamically creates HTML elements to represent each day in the calendar.

Highlights the current date if applicable.

pickDay(event):

Handles the click event on a calendar day.

Toggles the "current-date" class for the selected day.

Retrieves and displays tasks related to the selected date from the local storage.

Other functions:

There are several other functions related to handling UI interactions, such as changing the displayed month, updating the year, and formatting and displaying the current date and time.

The code also sets up event listeners for user interactions with the calendar, such as clicking on the month picker or navigating to the previous/next year.

6.2. Server side:

**openai:** Configures OpenAI with the API key.

**chatGPT:** Uses OpenAI's GPT-3.5-turbo model to generate responses based on user messages.

**Authentication and User Management Functions:**

login: Handles user login by comparing provided credentials with the stored data and generating JWT tokens.

loginGet: Sends the main HTML file for login page rendering.

registration: Manages user registration by hashing passwords and creating new user accounts.

usersGet: Sends user information in response to a GET request.

main: Retrieves user information and associated tasks for the main page.

**AI Interaction Function:**

askAi: Handles requests to interact with the OpenAI chat model, generating responses based on user queries.

**Task Management Functions:**

addTask: Adds a new task to the database.

delTask: Deletes a task from the database based on profile name and task details.

updateTask: Updates a task in the database with new information.

**Express Route Export:**

module.exports: Exports all the defined functions to be used as routes in the Express application.

7. API Documentation

Endpoints:

/login: POST - User login.

/registration: POST - User registration.

/main: GET - Retrieve user information.

/add: POST - Add a new task.

/delete: DELETE - Delete a task.

/change: PATCH - Update task details.

8. Security

User authentication is implemented using JWT tokens.

Data transmitted between the client and server is encrypted.

8. Conclusion:

This project delivers a task management system with user authentication, task CRUD operations, and integration with OpenAI's GPT-3.5-turbo. Users can register, log in, and manage tasks through a responsive interface. The system securely handles user data with JWT tokens and leverages AI for natural language processing, offering insights and suggestions based on task titles. Overall, it provides a seamless blend of user-centric features and AI interaction for efficient task management.

Content

**Title1**

**Introduction2**

**Architecture3**

**Functionality4**

**Installation and Configuration15**

**Getting started Guide16**

**Development17**

**API Documentation and Conclusion19**