Computer Engineering Essentials
Report

MyCourseMate

Presented By

6632239821 Suttanop Chanah

6630368721 Jednipit Porapakpenjul

6630357821 Akkharachai Yongsuwankul

2110221 Semester 1/2024

Department of Computer Engineering,

Faculty of Engineering, Chulalongkorn University

Table of contents

MyCourseMate: Your Gateway to Collaborative Learning

MyCourseMate is an innovative educational platform designed to foster knowledge-sharing and collaboration among computer engineering students. The platform offers an enriching environment where users can exchange ideas, share experiences, and gain access to valuable resources.

The website comprises three core sections:

1. **Home**: A centralized hub that introduces users to the platform and it offers a To-Do List feature, allowing each particular user to efficiently organize their tasks.
2. **Thread Discussions**: A space for meaningful conversations where users can post, comment, and engage in discussions on various topics.
3. **Exam Kits**: A dedicated area for users to find educational materials, and access curated learning resources for different subjects.

MyCourseMate is more than just a website; it's a community-driven platform that empowers users to explore, learn, and grow in their academic journey. Whether you're looking to collaborate on projects, seek study tips, or share your expertise, MyCourseMate is here to elevate your learning experience.

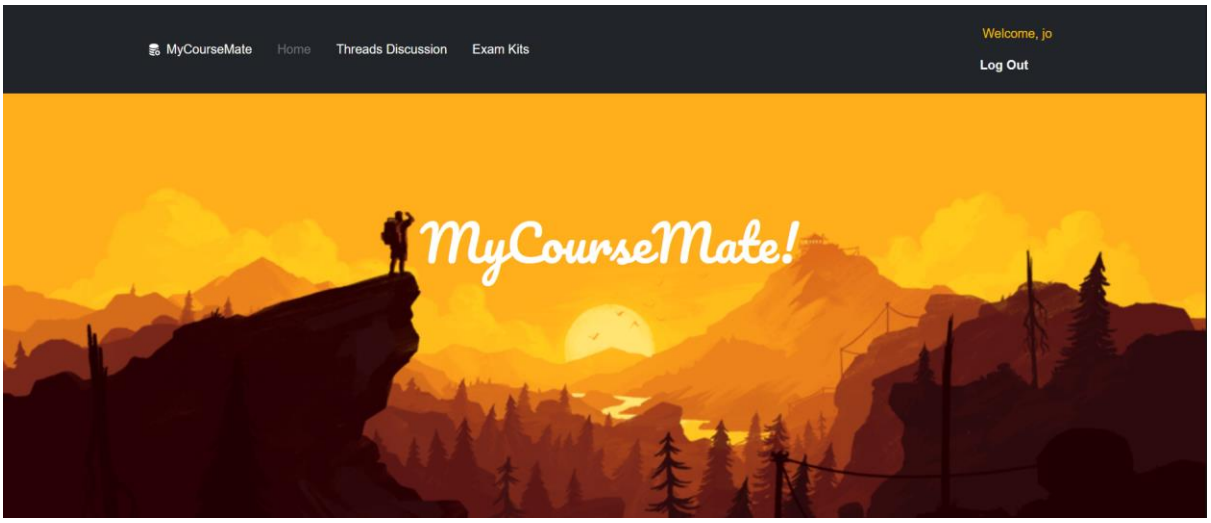

Figure 1 : MyCourseMate Home page

URL and How to use the application

URL

http://54.211.108.140:3221/

Source Code

https://github.com/RookieJoel/Final-Project-CEE.git

(branch - master)

How to use

1. Log-in or Sign Up
    - If you already have an account, simply log in.
    - If you don't have an account, sign up to create one.

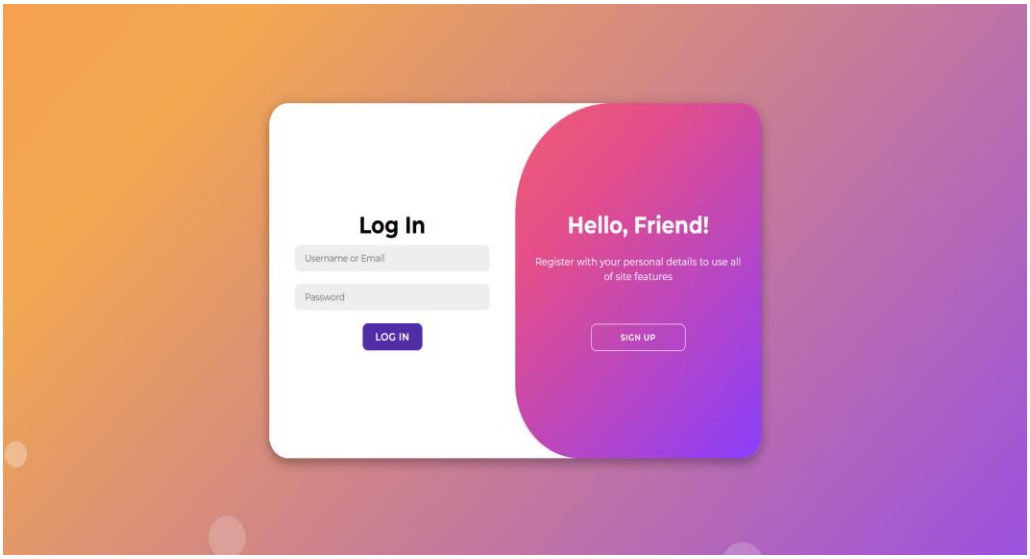

Figure 2 : log-in or sign-up page

2. Home Page
    - Use the home page to store and manage your to-do list.

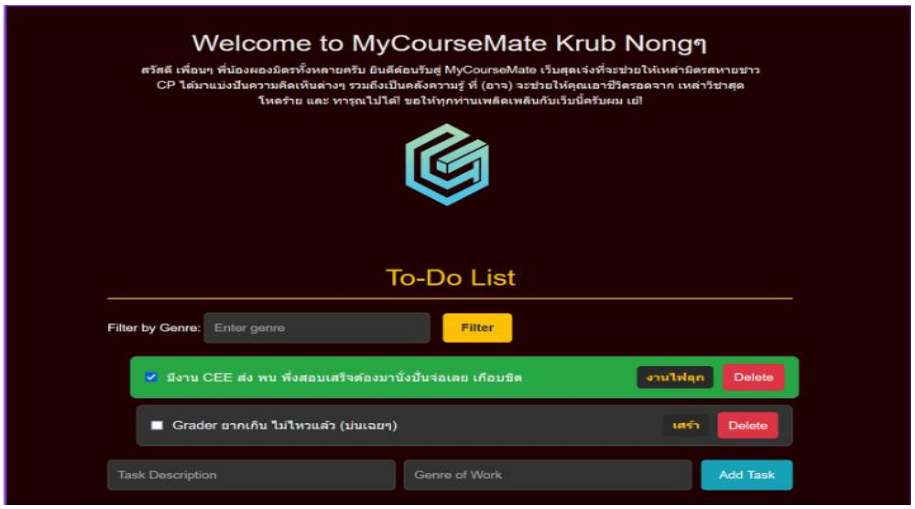

Figure 3 : To-Do list from home page

Create your todo-list and tag it for search. Press the mark to show that. Done and press the deleted button to delete.

3. Thread Page
    - Engage in discussions, leave comments, or create new topics to chat with others.

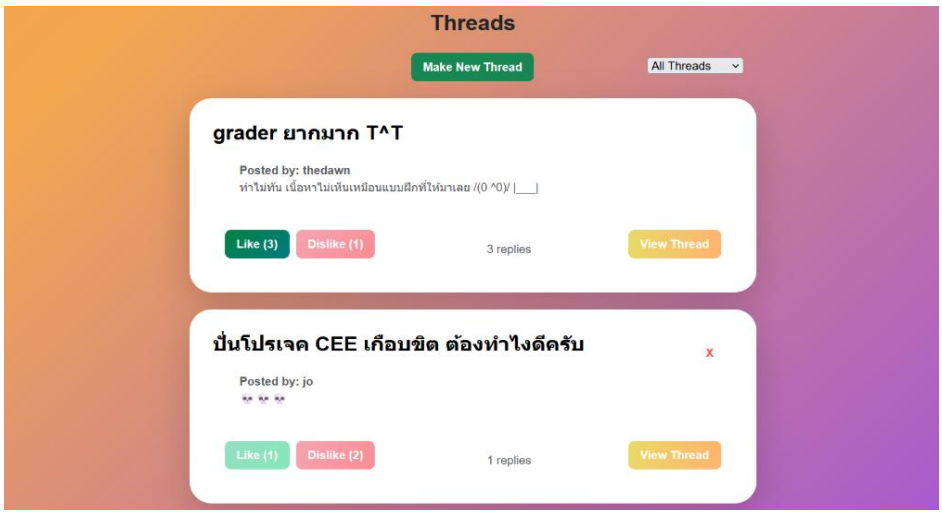

Figure 4 : Thread Discussions for sharing and posting threads

You can click a make new threads button to make a new thread, like or dislike page to your preferences.

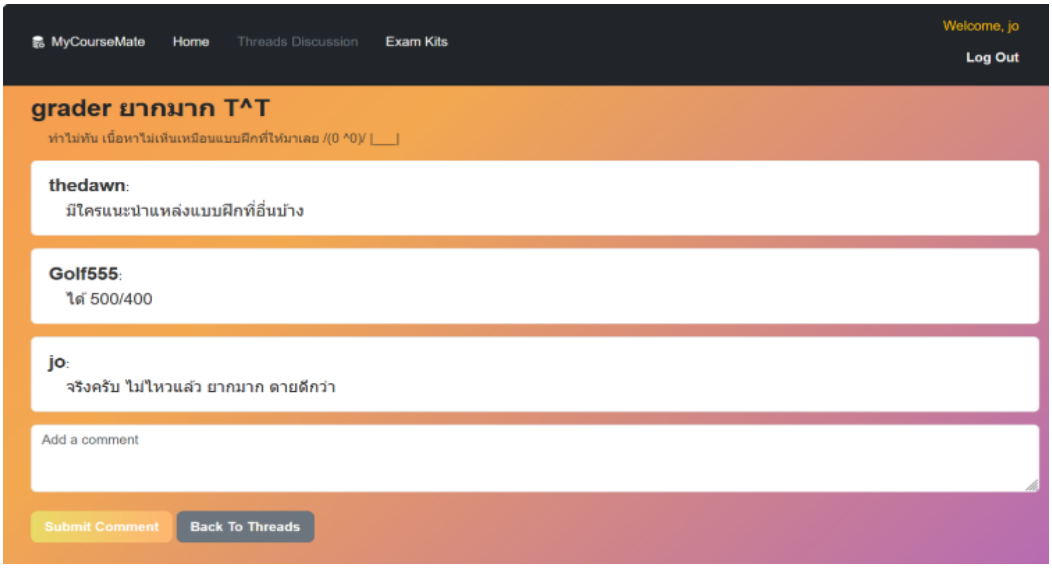

Figure 5 : Comments Section in Thread

4. Exam Kits Page
    - Access past exam papers and educational videos for your studies.

Figure 6 : Exam Kits Page

5. Log Out
    - Remember to log out when you're done using the system.

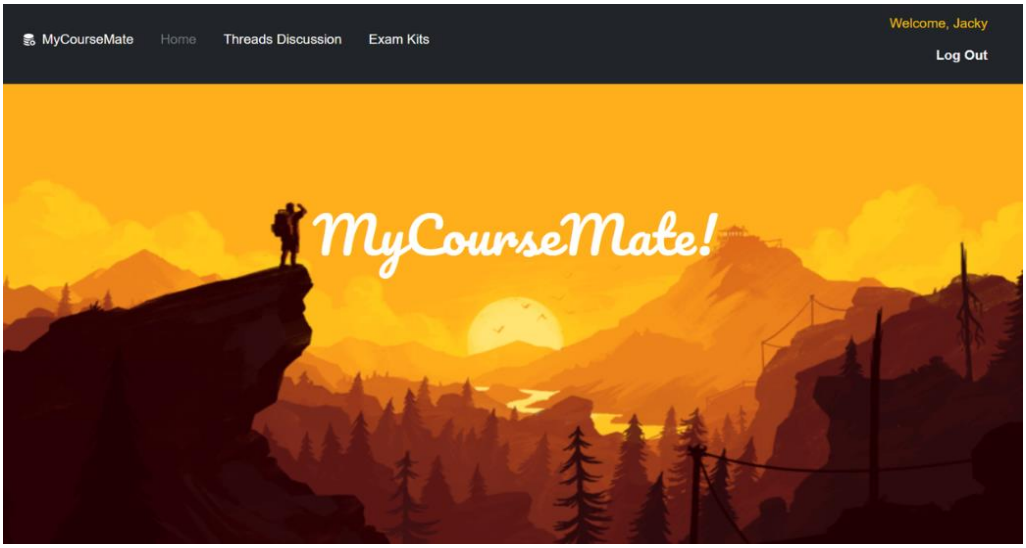

Figure 7 : Log out button is on the right top of the nav bar

Discussion regarding basic requirements

The development of MyCourseMate aligns with the stated basic requirements in the following ways:

1. **Multi-User-Support**

   MyCourseMate supports multiple user accounts, allowing users to log in, sign up, and access personalized features such as the To-Do list, thread discussions, and access to exam kits. This functionality mirrors the requirements outlined in Activity 6, ensuring multi-user capabilities are effectively implemented.

2. **Front-End-Development**

   The front-end of MyCourseMate adheres strictly to the guidelines by avoiding any additional libraries, frameworks, or plugins outside of those permitted in Activities 5 and 6. The web application utilizes **JavaScript**, **HTML**, and **CSS** while leveraging modern Web APIs for animations and dynamic effects (e.g., parallax scrolling, button interactions, carousel, and To-Do list animations). These implementations ensure compliance while enhancing the user experience.

3. **Back-End-Development**

   The back-end is built without relying on unauthorized libraries, frameworks, or plugins, strictly adhering to the tools and methods used in Activities 9 and 11. This ensures all server-side functionality, including user authentication, data management, and API integration, meets the requirements. It ensures compatibility with the project scope and maintains simplicity and efficiency in the development.

4. **Team Contribution and Collaboration**

   The project was developed collaboratively by all team members, with clear roles and responsibilities agreed upon in advance. Each member contributed to specific components of the web application:

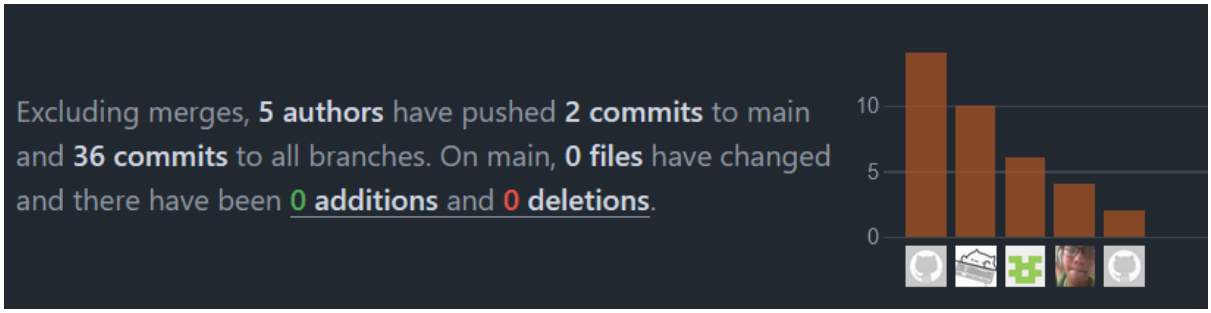

Figure 8 : Contribution on Github

From the graph, it is evident that each team member has fulfilled their responsibilities according to the planned tasks. Initially, each contributor committed their individual feature or assigned portion of the project, reflecting balanced contributions in the early stages of development.

Following this, the team transitioned to working collaboratively on a shared environment hosted on **AWS EC2**. This collaborative phase required centralized code management and adjustments, leading to a significant number of commits from one contributor. This individual likely acted as the main integrator or coordinator, managing the merging of different features and resolving conflicts to ensure smooth integration, which explains their higher number of commits.

Project Structure

Frontend Structure

```
frontend/
│
├── node_modules/
├── public/
│       ├── IMG/
│       ├── script /
│       │       ├── dashboard.js
│       │       ├── goto.js
│       │       ├── login.js
│       │       ├── mainPage.js
│       │       ├── signup.js
│       │       ├── threadScripts.js
│       │
│       ├── Exam_Kits.css
│       ├── thread.css
│       ├── home.css
│       ├── Exam_Kits.html
│       ├── home.html
│       ├── login.html
│       ├── signup.html
│       ├── thread.html
│
├── package-lock.js
├── package.js
├── server.js
```

Backend Structure

```
Backend/
│
├── node_modules/
│
├── src/
│   ├── confic/
│   │   ├── db.js/
│   │
│   ├── controller/
│   │   ├── storeUser.js
│   │   ├── threadController.js
│   │   ├── todoController.css
│   │
│   ├── models/
│   │   ├── thread.js
│   │   ├── todo.js
│   │   ├── user.js
│   │
│   ├── routes/
│   │   ├── auth.js
│   │   ├── threadRoutes.js
│   │   ├── todoRoutes.js
│
├── .env
├── package.json
├── package-lock.json
├── server.js
```

API List		
No.	Endpoint	Detail
1	GET/api/auth/session	Login Authentication
2	POST/api/auth/logout	Logout
3	GET/api/threads	Get Thread Data
4	POST/api/threads	Create New Thread And Post
5	GET/api/threads/:id/comments	Get Comment Data
6	POST/api/threads/:id/comments	Create New Comment And Post
7	DELETE/api/threads/:id	Delete Thread
8	PATCH/api/threads/:id/likes	Like/Dislike Thread
9	GET/api/todos	Get User Todo-list
10	POST/api/todos	Create New User Todo-list
11	DELETE/api/todos	Delete User Todo-list

```
import mongoose from 'mongoose';

const todoSchema = new mongoose.Schema({
  task: { type: String, required: true },
  genre: { type: String, required: true },
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
  completed: { type: Boolean, default: false }, // Add completion field
});

// Export with default
const Todo = mongoose.model('Todo', todoSchema);
export default Todo;
```

Figure 9 : ToDo Schema

```
const userSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true }, // Add username
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
});
```

Figure 10 : User Schema

```
const threadSchema = new mongoose.Schema({
  id: String,
  title: String,
  description: String,
  username: { type: String, required: true },
  likes: [String],
  dislikes: [String],
  comments: [
    {
      user: String,
      text: String,
      likes: { type: Number, default: 0 },
      dislikes: { type: Number, default: 0 }
    }
  ]
});
```

Figure 11 : Thread Schema

Discussion regarding challenging requirements

1. Responsive Web Application

Model: IPhoneSE

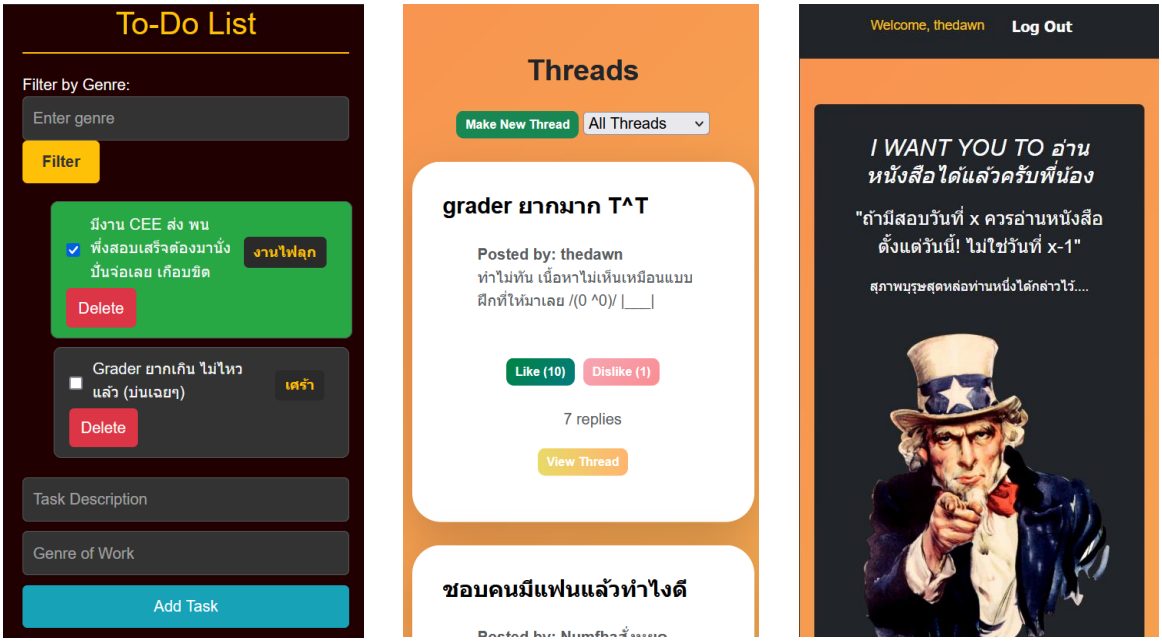

Figure 12 : Responsive website for IPhone

Model: IPadMini

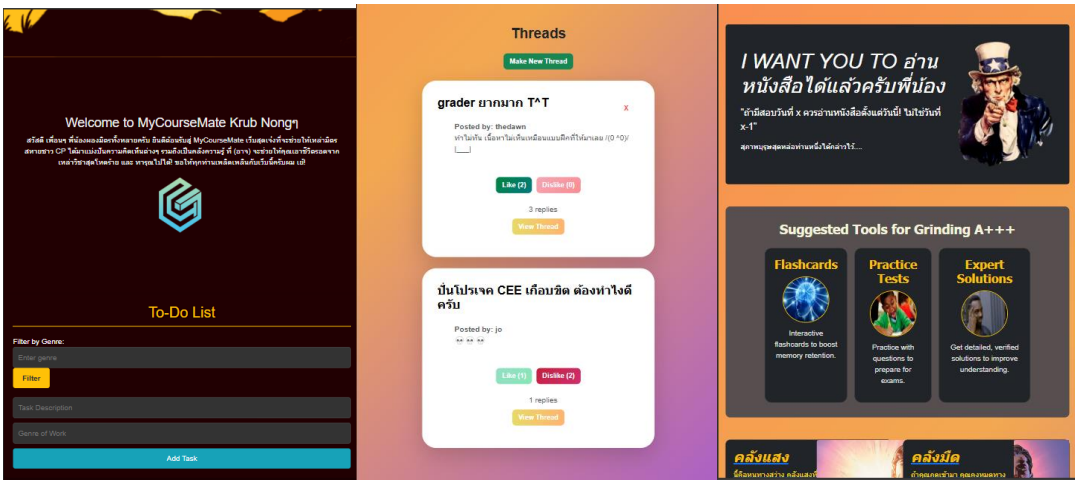

Figure 13 : Responsive website for IPad

2. Nice look

all pages of our website have 300+ css code and have animation in some pages.

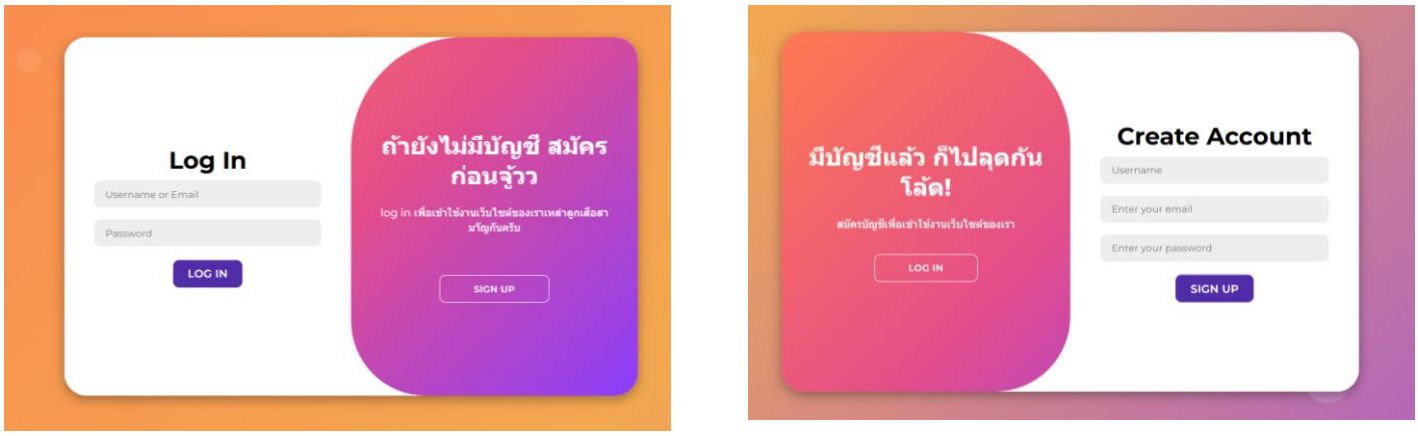

Figure 14 : Responsive website for IPad

The **Log-In and Sign-Up pages** feature a captivating gradient background paired with bubble animations, creating a modern and dynamic first impression. Utilizing JavaScript, seamless animations enable users to switch between Log-In and Sign-Up forms within the same page, eliminating the need to fetch separate pages and enhancing user experience.

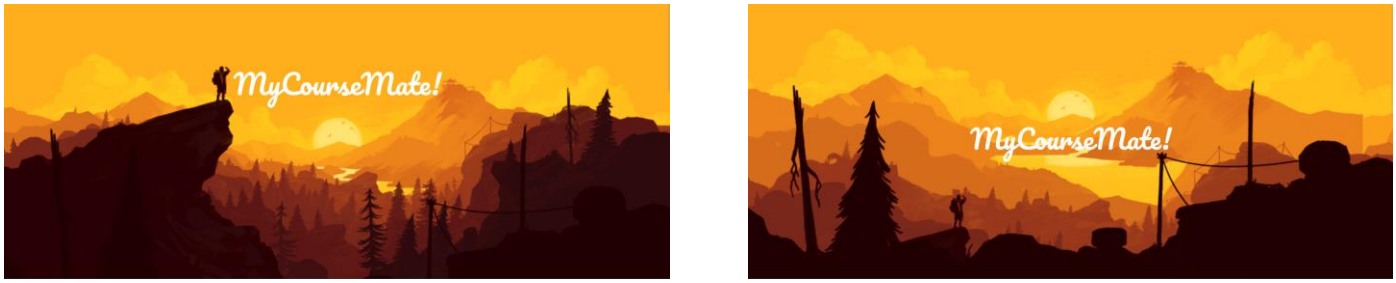

Figure 15 : Parallax Design for home page

The Home page incorporates a stunning **parallax** design, ensuring a visually engaging and smooth browsing experience. Across the platform, various animations bring the website to life: button interactions, thread animations, and the marking of To-Do list items all contribute to a dynamic and responsive interface. Additionally, a carousel feature allows users to easily browse and select their desired subjects, combining beauty and practicality to create an immersive and enjoyable user journey.

3. Unique feature that enhances user experience.

One unique feature that significantly enhances the user experience of **MyCourseMate** is its **interactive and visually dynamic design elements**, seamlessly integrated with core functionalities:

1. **Interactive Animations**:
    - **Thread Discussion Page**: Animations enhance user interactions when posting or commenting, making the platform more engaging.
    - **To-Do List Feature**: Users experience satisfying animations when marking tasks as complete, with checkmarks turning green to visually indicate completion, making the system intuitive and easy to use.
    - **Carousel for Subject Search**: The carousel feature allows users to browse subjects interactively, combining ease of navigation with an aesthetically pleasing design.

2. **Advanced To-Do List Features**: The to-do list allows users to add tags freely, enabling flexible filtering options for tasks. This improves task organization and helps users quickly find relevant entries.

3. **Thread Sorting and Filtering**:

    - **Sort by Popularity**: Threads are automatically sorted based on the number of likes/dislikes, ensuring the most interesting or engaging posts are easily visible.
    - **Filter System**: Users can filter threads by categories, tags, or specific topics, simplifying the search for relevant discussions.

Project plan and actual execution

1. Project plan

- Planning Section: We are brainstorming to determine which project ideas will be accepted for the requirement,which makes us do MyCouseMate website.
- Design Section: We design our website to 3 parts, Homepage, Thread and Exam Kit. Then, choose the theme of our website.
- Front-end Dev: Developing front-end form each page.
- Back-end Dev:Developing back-end form each page and login system.
- Integration: Connect all code together.
- Testing: Check all systems that may have some bug. Then fix it.
- Deploying:Send this website to our friends and get feedback.

Gantt Chart														
List	8 Nov	9 Nov	10 Nov	11 Nov	12 Nov	13 Nov	14 Nov	15 Nov	16 Nov	17 Nov	18 Nov	19 Nov	20 Nov	21 Nov
Plan														
Design														
Front-end Dev														
Back-end Dev														
Integration														
UAT Test														
Deploying														
Review & Present														

Gantt Chart														
List	22 Nov	23 Nov	24 Nov	25 Nov	26 Nov	27 Nov	28 Nov	29 Nov	30 Dec	1 Dec	2 Dec	3 Dec	4 Dec	5 Dec
Plan														
Design														
Front-end Dev														
Back-end Dev														
Integration														
UAT Test														
Deploying														
Review & Present														

2. Project Excecute

2.1 Front-end Development

- Set all environment like activity in class to initialize this project(create folder like plan that we set).
- Write HTML code to set the structure of the website. Then write CSS code to make our website prettier.

- Write Js code for some function in frontend(some will connect to backend)
- Connect the frontend with the backend code(Fetch).
- Test function in frontend, fix bugs and set responsiveness.

2.2 Back-end Development

- Set all environment like activity in class to initialize this project(create folder like plan that we set).
- Write Js code in part models, controllers and routes.
- Integrate all to the server and test run it.
- Test function in backend and fix bugs.