

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ

ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

# Кафедра системного програмування та спеціалізованих комп’ютерних систем

**Лабораторна робота №2**

з дисципліни **Веб-дизайн**

*на тему:* **“Розробка функціональності Web-додатка мовою Javascript”**

Виконав:

студент ІII курсу

групи КВ-11

Рибалка Д. В.

Перевірив:

Київ – 2024

*Мета:* ознайомитись із засобами мови Javascript та навчитись їх застосовувати побудови Web-інтерфейсу користувача.

*Загальне завдання:* розробити функціональність для статичних сторінок Web-додатку першої лабораторної роботи із використанням шаблону MVC Інструменти розробки: мови HTML5, CSS3, Javascript.

*Посилання на Github-репозиторій*:

<https://github.com/denisrybalka-kpi/web-lab2>

*Посилання на Github-pages*:

<https://denisrybalka-kpi.github.io/web-lab2/>

*Контакт студента у месенджері Телеграм*:

<https://t.me/denisrybalka10>

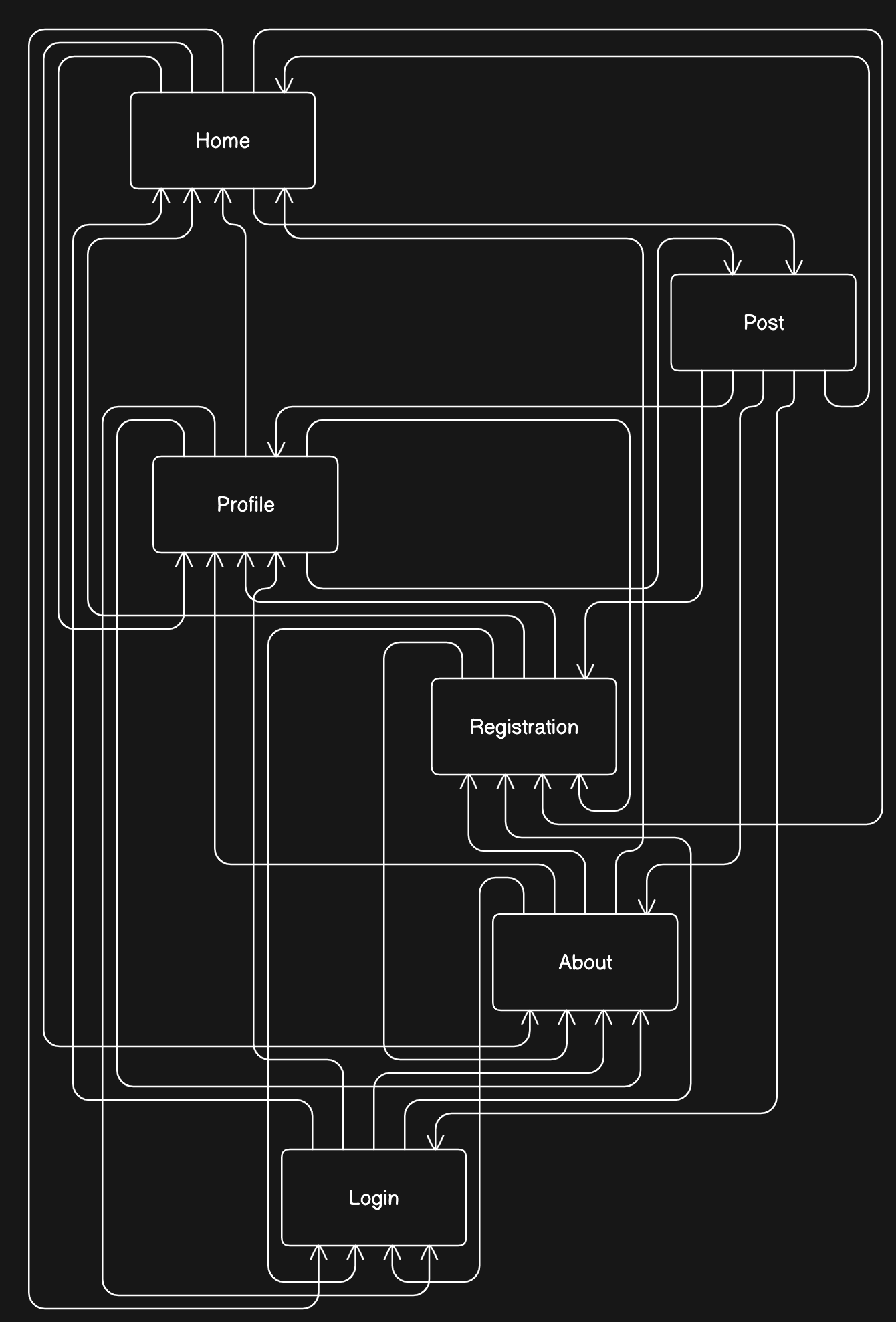
**Назва додатку:**

IdeaHub

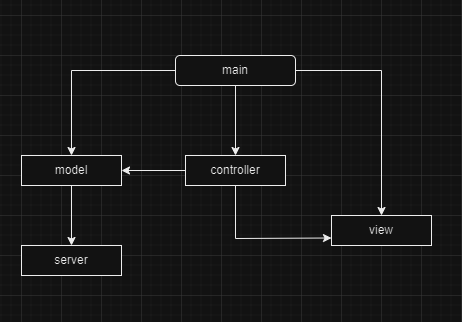
**Короткий опис його функціональності:**

Цей веб-додаток представляє ідею соціальної платформи для обміну ідеями та думками. Користувачі можуть додавати свої дописи, лайкати дописи інших, використовувати пагінацію та пошук по назві дописів.

**Схема переходу:**



**Діаграма класів додатку:**



**Скріншоти усіх сторінок веб-додатку:**

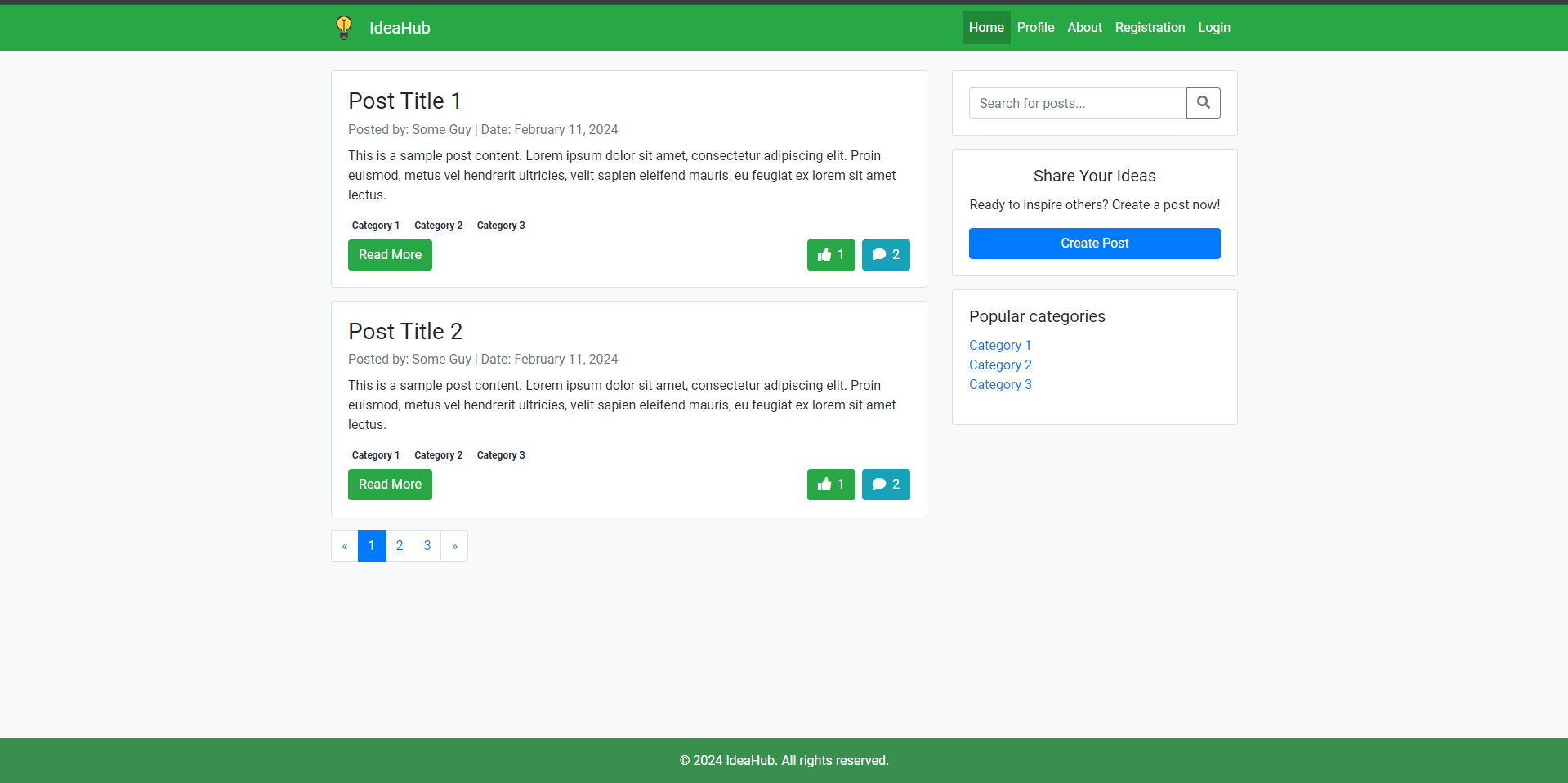


Рисунок 1 – Головна сторінка додатку

Рисунок 2 – Сторінка профілю

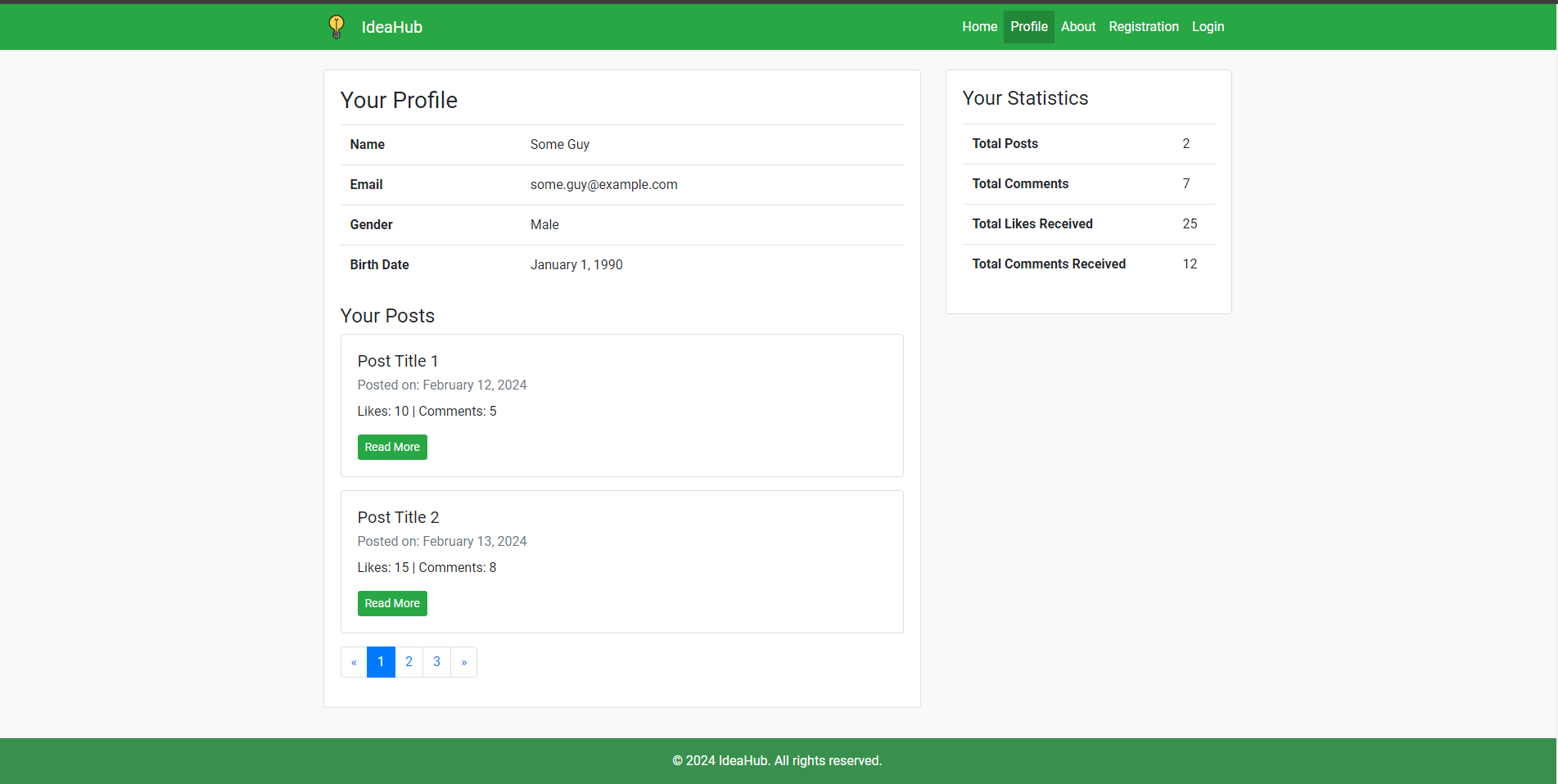


Рисунок 3 – Сторінка про додаток

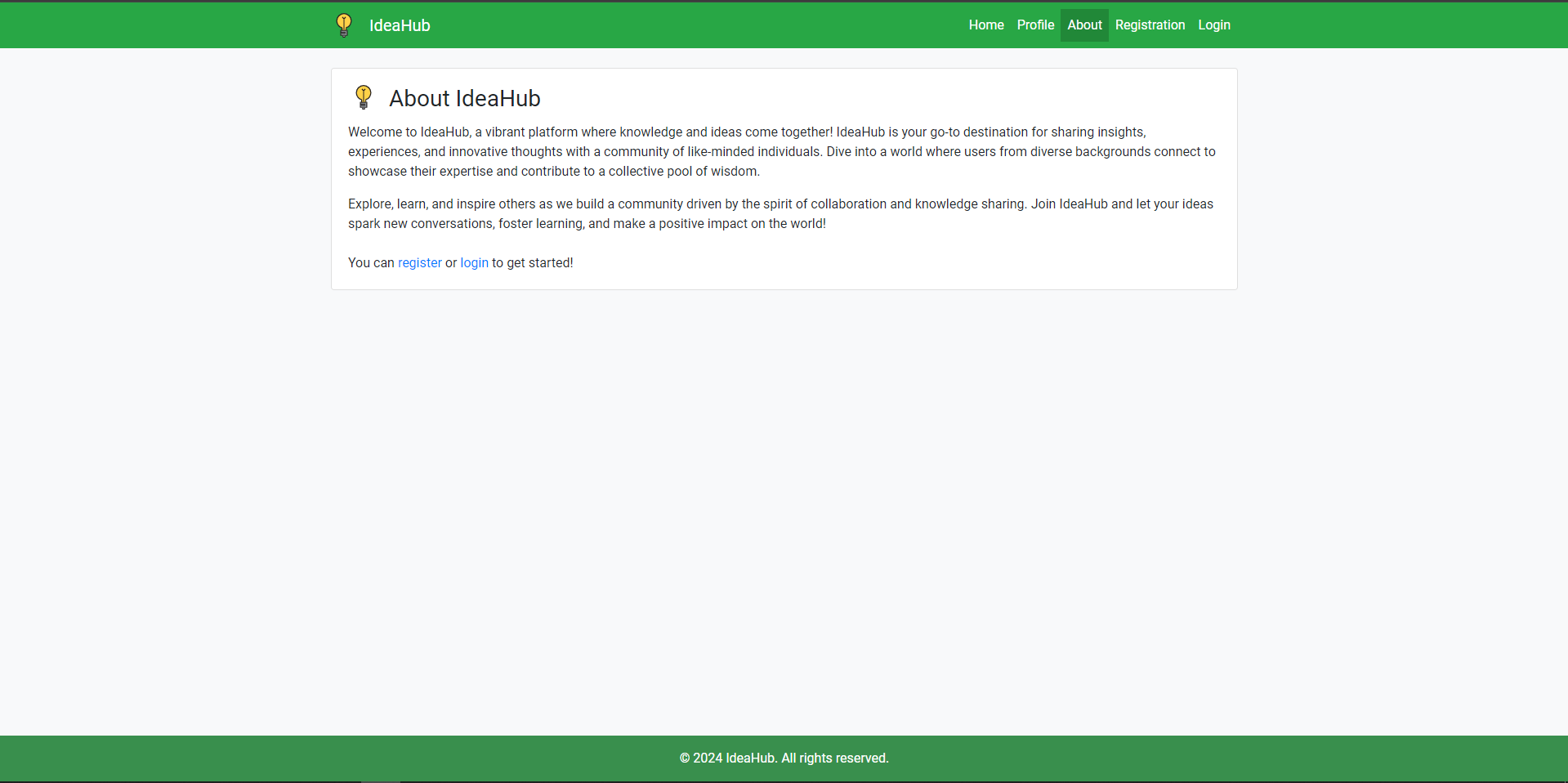


Рисунок 4 – Сторінка реєстрації

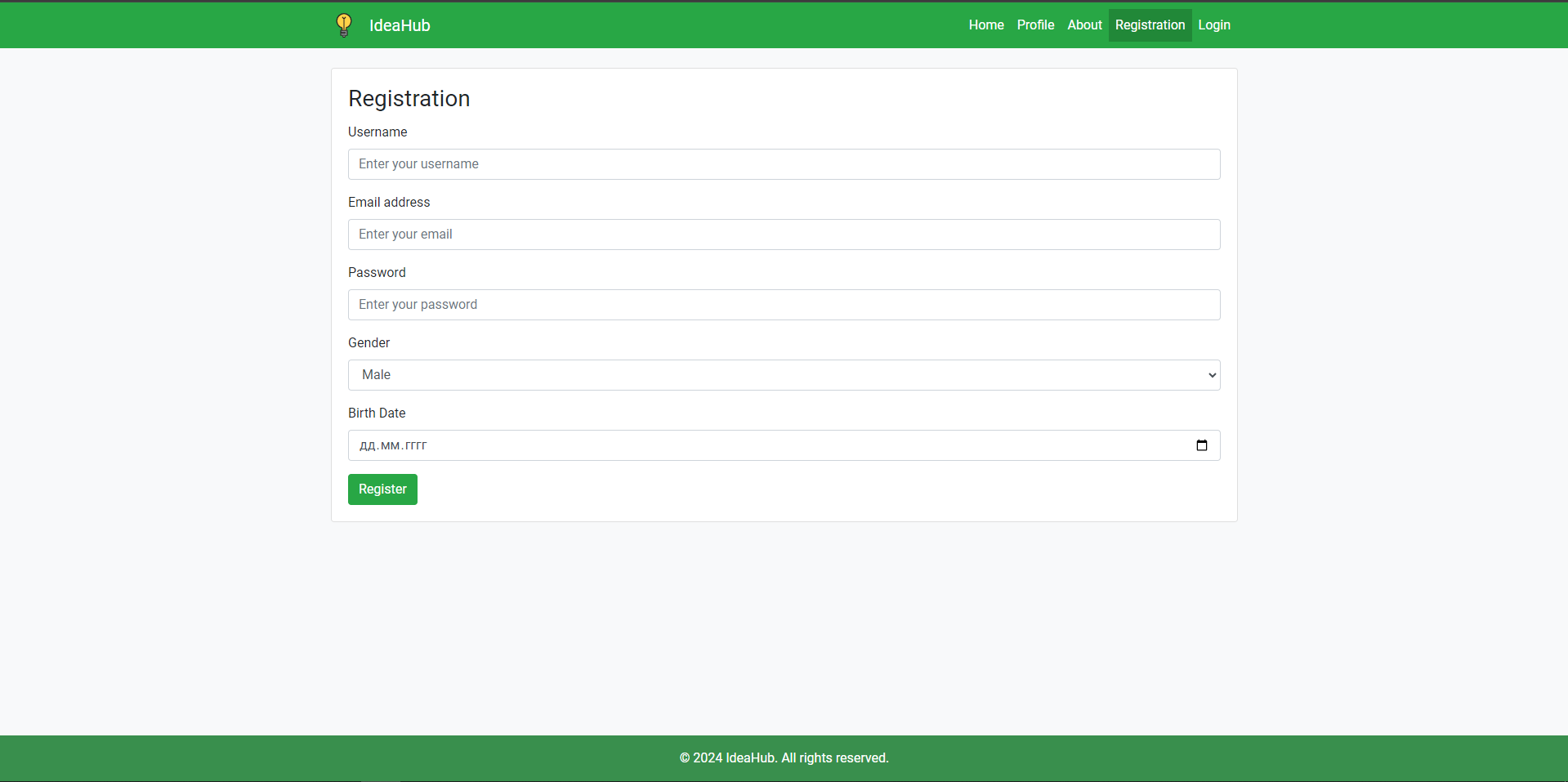


Рисунок 5 – Сторінка логіну

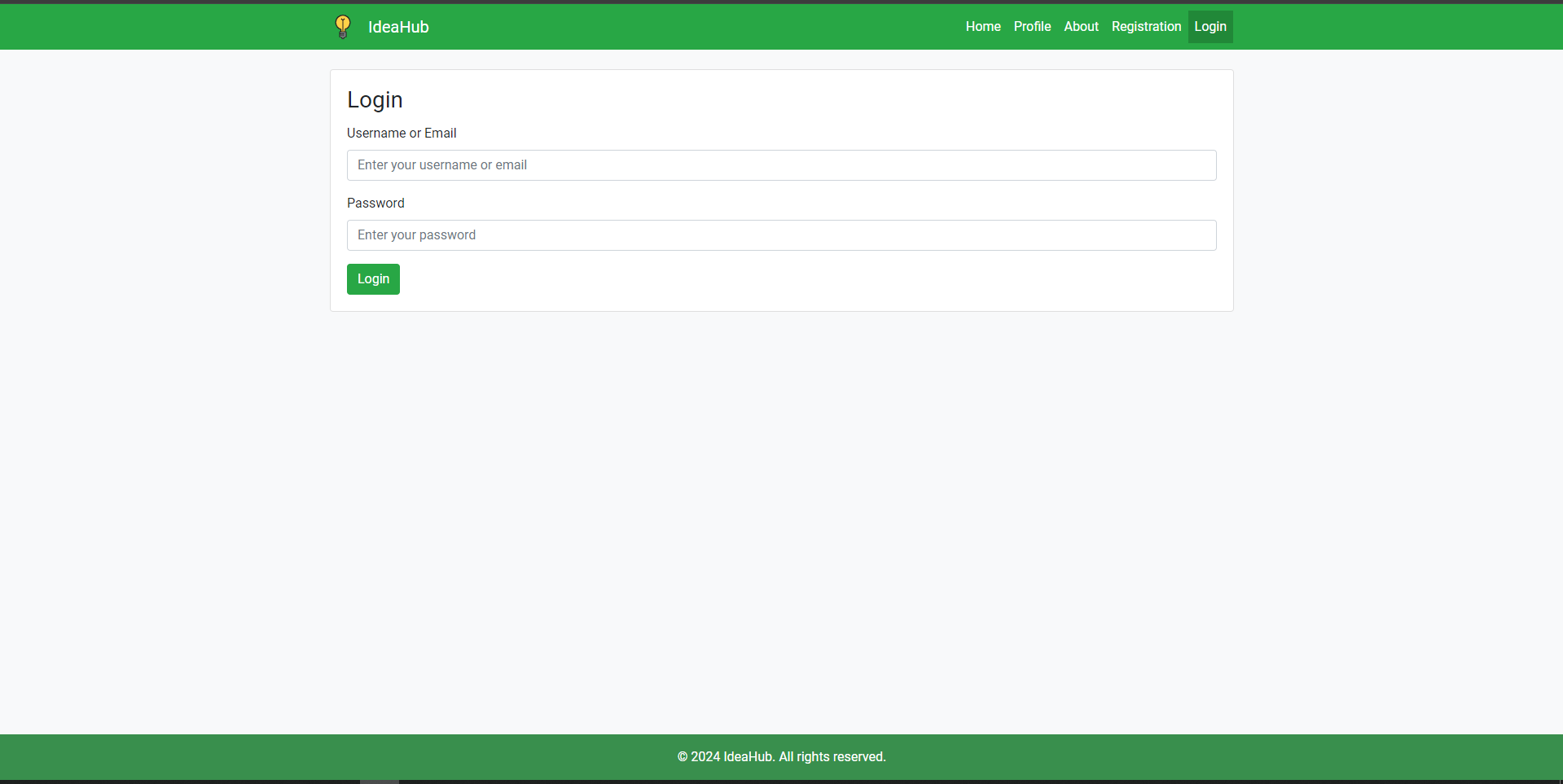


Рисунок 6 – Сторінка перегляду посту

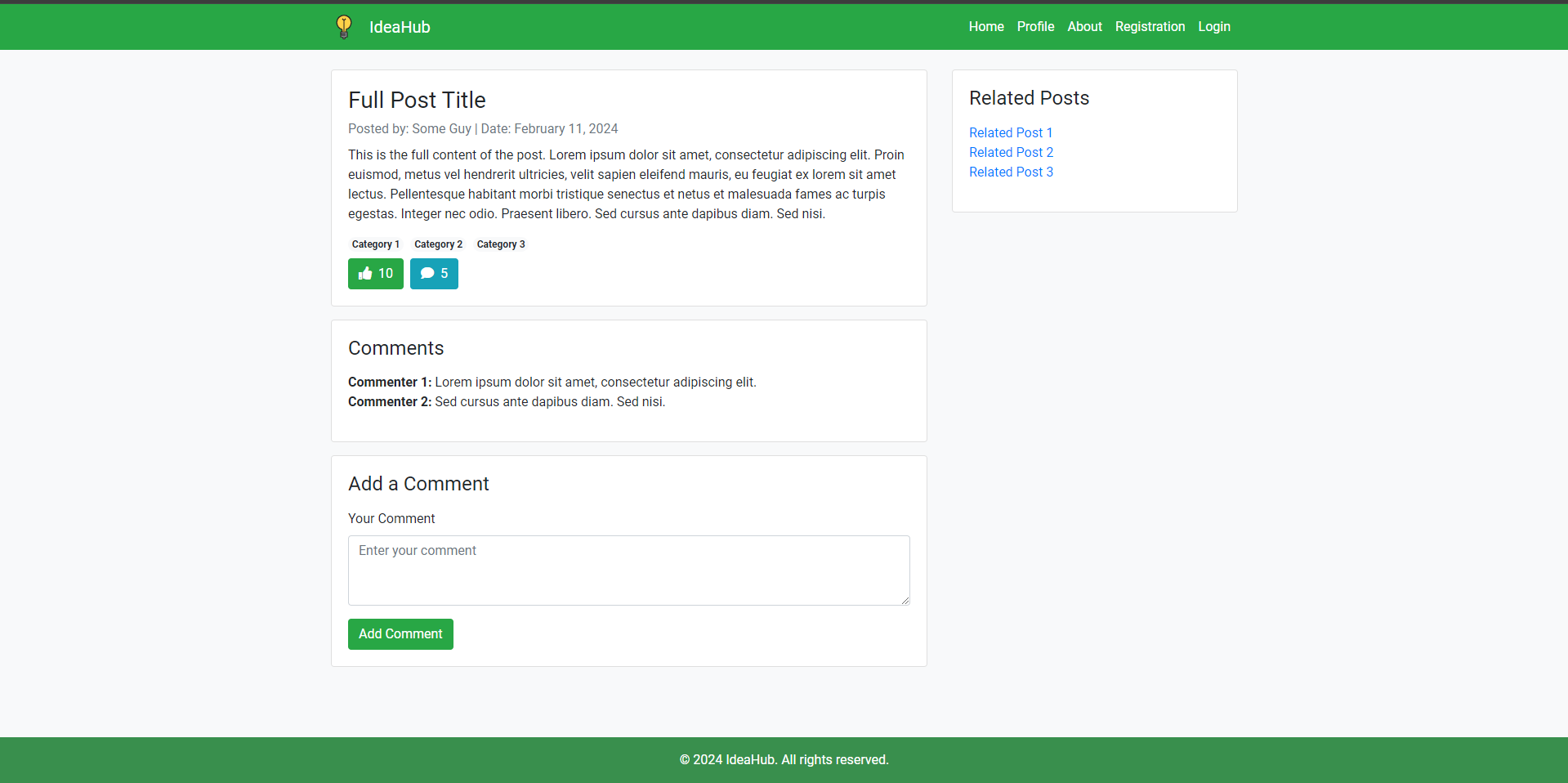


Рисунок 7 – Модальне вікно для створення посту

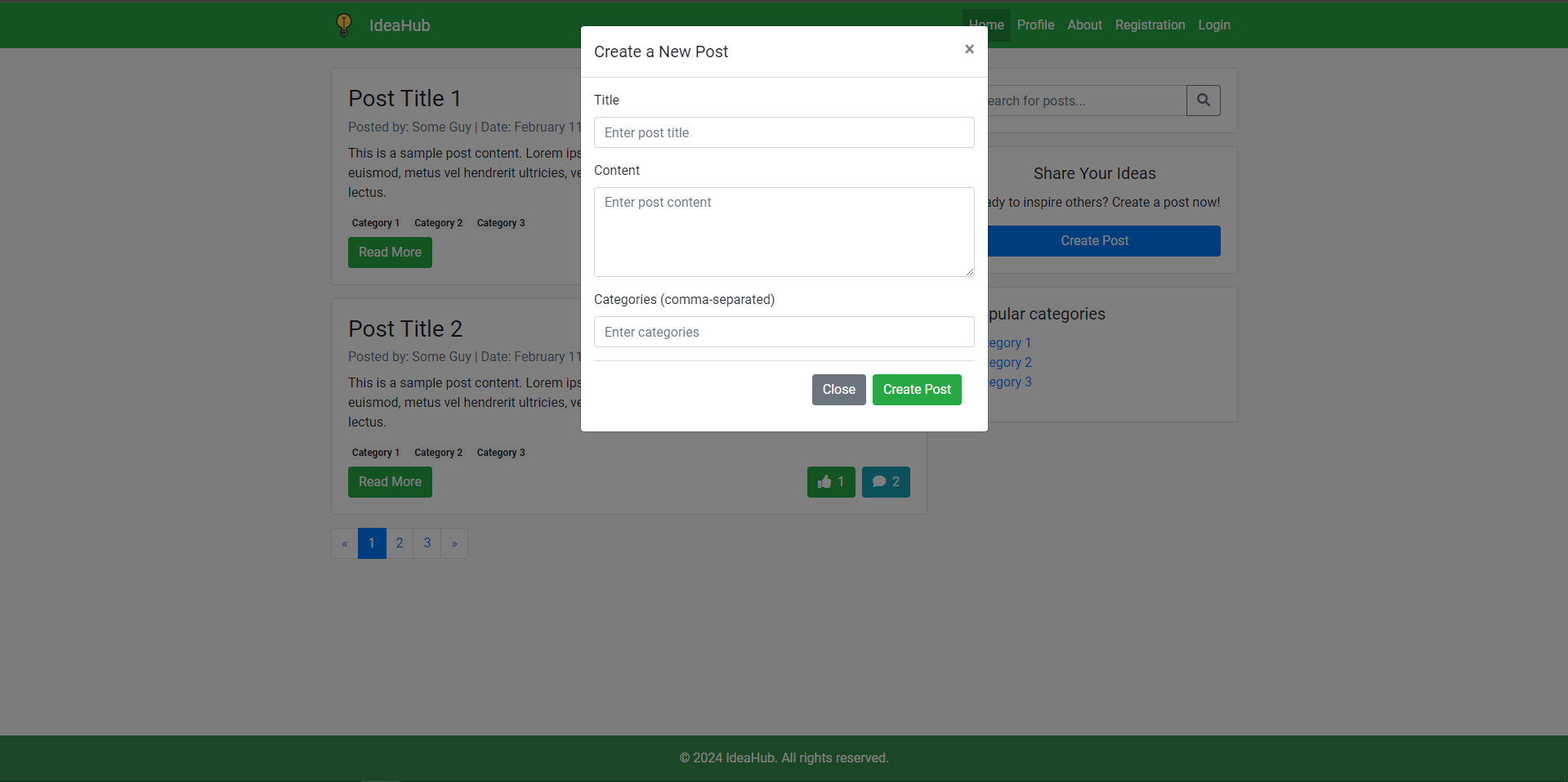
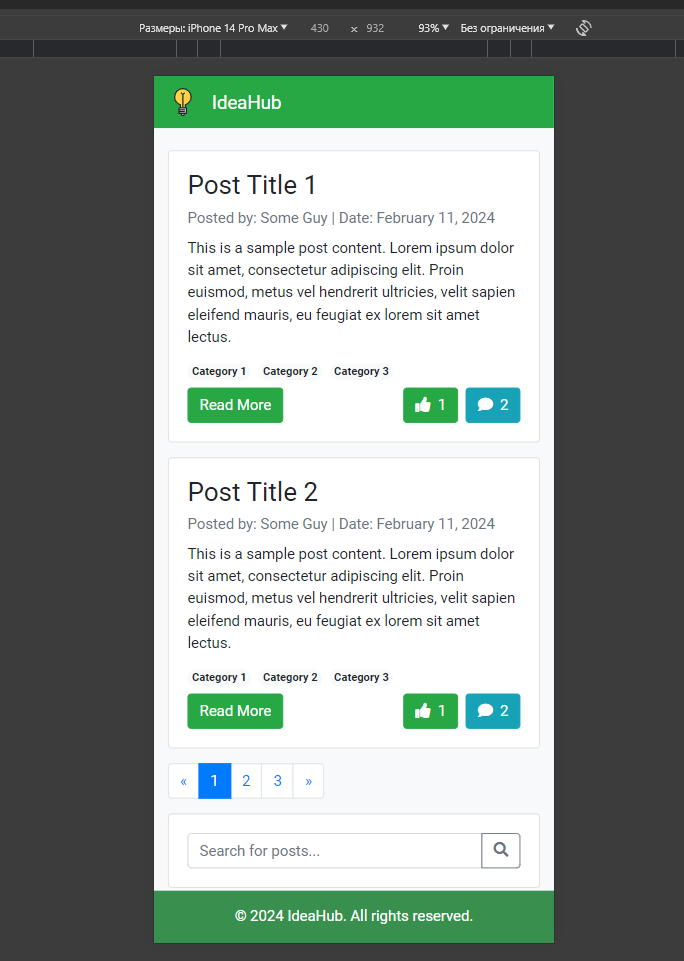


Рисунок 8 – Перегляд головної сторінки на мобільному



**Код JavaScript файлів**

*main.js*

import { PostModel } from "./model.js";

import { PostView } from "./view.js";

import { PostController } from "./controller.js";

import { MockServer } from "./server/index.js";

document.addEventListener("DOMContentLoaded", () => {

const model = new PostModel(MockServer);

const view = new PostView();

new PostController(model, view);

});

*view.js*

export class PostView {

constructor() {

this.postContainer = document.querySelector(".postsContainer");

this.addPostButton = document.querySelector("#addPost");

this.titleInput = document.querySelector("#postTitle");

this.contentInput = document.querySelector("#postContent");

this.categoriesInput = document.querySelector("#postCategories");

this.closeModalButton = document.querySelector("#closeAddPostModal");

this.paginationContainer = document.querySelector(".pagination");

this.searchInput = document.querySelector("#searchInput");

this.paginationContainer = document.querySelector(".pagination");

}

getPostDetails() {

return {

title: this.titleInput.value,

content: this.contentInput.value,

categories: this.categoriesInput.value.split(",").map((s) => s.trim()),

};

}

resetForm() {

this.titleInput.value = "";

this.contentInput.value = "";

this.categoriesInput.value = "";

}

closeAddPostModal() {

this.closeModalButton.click();

}

updateLikeButton(postId, isLiked, likes) {

const likeButton = document.querySelector(

`.like-button[data-post-id="${postId}"]`

);

if (likeButton) {

likeButton.classList.toggle("liked", isLiked);

const likeCountElement = likeButton.querySelector(".like-count");

if (likeCountElement) {

likeCountElement.textContent = likes;

}

}

}

displayPagination(totalPosts, currentPage, pageSize) {

const totalPages = Math.ceil(totalPosts / pageSize);

this.paginationContainer.innerHTML = "";

for (let i = 1; i <= totalPages; i++) {

const pageItem = document.createElement("li");

pageItem.className = `page-item ${i === currentPage ? "active" : ""}`;

pageItem.innerHTML = `<a class="page-link" href="#">${i}</a>`;

this.paginationContainer.appendChild(pageItem);

}

}

showPagination() {

this.paginationContainer.style.display = "flex";

}

hidePagination() {

this.paginationContainer.style.display = "none";

}

displayPost(post) {

const {

title,

date,

categories,

content,

creator,

likes,

comments,

id,

isLiked,

} = post;

const postElement = document.createElement("div");

postElement.className = "card mb-3 postItem";

postElement.innerHTML = `

<div class="card-body d-flex flex-column">

<h3 class="card-title">${title}</h3>

<p class="card-subtitle text-muted mb-2">

Posted by: ${creator} | Date: ${date}

</p>

<p class="card-text">${content}</p>

<div class="d-flex flex-wrap mb-2">

${categories

.map(

(category) =>

`<span class="badge badge-light mr-2 text-bg-dark rounded-pill hoverable">${category}</span>`

)

.join("")}

</div>

<div class="d-flex justify-content-between align-items-center mt-auto">

<a href="pages/post.html" class="btn btn-success">Read More</a>

<div class="d-flex gap-2">

<span class="btn btn-success mr-2 like-button ${

isLiked ? "liked" : ""

}" data-post-id="${id}">

<i class="fas fa-thumbs-up mr-2"></i><span class="like-count">${likes}</span>

</span>

<span class="btn btn-info">

<i class="fas fa-comment mr-2"></i>${comments}

</span>

</div>

</div>

</div>

`;

this.postContainer.insertAdjacentElement("beforeend", postElement);

}

}

*controller.js*

export class PostController {

constructor(model, view) {

this.model = model;

this.view = view;

this.init();

}

async init() {

await this.displayPosts(this.model.getCurrentPage());

this.bindSearchEvent();

this.bindPaginationEvents();

this.addPostButtonListeners();

this.addLikeButtonListeners();

}

addLikeButtonListeners() {

this.view.postContainer.addEventListener("click", async (event) => {

const button = event.target.closest(".like-button");

if (!button) return;

event.preventDefault();

const postId = parseInt(button.dataset.postId, 10);

const post = await this.model.toggleLike(postId);

this.view.updateLikeButton(postId, post.isLiked, post.likes);

});

}

addPostButtonListeners() {

this.view.addPostButton.addEventListener("click", this.addPost.bind(this));

}

async addPost(event) {

event.preventDefault();

const { title, content, categories } = this.view.getPostDetails();

await this.model.addPost(title, content, categories);

this.displayPosts(1);

this.view.resetForm();

this.view.closeAddPostModal();

}

async displayPosts(page) {

const posts = await this.model.getPostsByPage(page);

const totalPosts = this.model.getTotalPosts();

this.view.postContainer.innerHTML = "";

posts.forEach((post) => this.view.displayPost(post));

this.view.displayPagination(totalPosts, page, this.model.getPageSize());

}

bindSearchEvent() {

this.view.searchInput.addEventListener("keyup", async () => {

const query = this.view.searchInput.value.trim();

if (query.length > 0) {

this.view.hidePagination();

const searchResults = await this.model.searchPosts(query);

this.view.postContainer.innerHTML = "";

searchResults.forEach((post) => this.view.displayPost(post));

} else {

this.view.showPagination();

this.displayPosts(1);

}

});

}

bindPaginationEvents() {

this.view.paginationContainer.addEventListener("click", (e) => {

if (e.target.tagName === "A") {

const page = parseInt(e.target.textContent);

if (!isNaN(page)) {

this.model.setCurrentPage(page);

this.displayPosts(page);

}

}

});

}

}

*model.js*

export class PostModel {

constructor(MockServer) {

this.server = new MockServer();

this.pageSize = 2;

this.currentPage = 1;

this.totalPosts = null;

}

async toggleLike(postId) {

return await this.server.toggleLike(postId).then((updatedPost) => {

return updatedPost;

});

}

async addPost(title, content, categories) {

const post = {

title,

content,

categories,

likes: 0,

comments: 0,

creator: "User",

date: new Date().toLocaleDateString("en-GB"),

};

return await this.server.addPost(post).then((newPost) => {

return newPost;

});

}

async searchPosts(query) {

return await this.server.searchPosts(query);

}

async getPostsByPage(pageNumber) {

return await this.server

.fetchPosts({ pageNumber, pageSize: this.pageSize })

.then(({ totalPosts, posts }) => {

this.totalPosts = totalPosts;

return posts;

});

}

setCurrentPage(page) {

this.currentPage = page;

}

getCurrentPage() {

return this.currentPage;

}

getPageSize() {

return this.pageSize;

}

getTotalPosts() {

return this.totalPosts;

}

}

*server.js*

export class MockServer {

constructor() {

this.posts = [

{

id: 1,

title: "The Secret of Making Perfect Coffee",

content:

"Ever wondered what makes coffee taste so good? It's not just about the beans. This post dives into the art of coffee making, exploring the importance of water temperature, grind size, and brewing time. Join us on a journey to unlock the secret of the perfect cup of coffee.",

categories: ["Lifestyle", "Food & Drink", "DIY"],

likes: 15,

comments: 4,

creator: "Coffee Connoisseur",

date: "07/04/2024",

},

{

id: 2,

title: "Exploring the Night Sky: Tips for Amateur Astronomers",

content:

"The stars have fascinated humanity since time immemorial. This post is your guide to exploring the night sky, with tips on choosing the right telescope, finding constellations, and understanding celestial events. Whether you're a seasoned astronomer or a curious beginner, the cosmos has wonders waiting for you.",

categories: ["Science", "Education", "Hobbies"],

likes: 22,

comments: 7,

creator: "Stargazer",

date: "14/03/2024",

},

{

id: 3,

title: "The Future of Renewable Energy",

content:

"Renewable energy is pivotal in our fight against climate change. This post explores the latest advancements in solar, wind, and hydroelectric power. We discuss the challenges and opportunities that lie ahead in the quest for a sustainable energy future. Learn about the technologies that will power our world tomorrow.",

categories: ["Technology", "Environment", "Innovation"],

likes: 30,

comments: 12,

creator: "Eco Innovator",

date: "22/02/2024",

},

];

}

getMockServerResponseDelay() {

return Math.random() \* 250;

}

async fetchPosts({ pageNumber, pageSize }) {

return new Promise((resolve) => {

setTimeout(() => {

const start = (pageNumber - 1) \* pageSize;

const end = start + pageSize;

resolve({

posts: this.posts.slice(start, end),

totalPosts: this.posts.length,

});

}, this.getMockServerResponseDelay());

});

}

async addPost(post) {

return new Promise((resolve) => {

setTimeout(() => {

const newPost = { ...post, id: this.posts.length + 1 };

this.posts.unshift(newPost);

resolve(newPost);

}, this.getMockServerResponseDelay());

});

}

async toggleLike(postId) {

return new Promise((resolve) => {

setTimeout(() => {

const postIndex = this.posts.findIndex((post) => post.id === postId);

if (postIndex !== -1) {

this.posts[postIndex].isLiked = !this.posts[postIndex].isLiked;

this.posts[postIndex].likes += this.posts[postIndex].isLiked ? 1 : -1;

resolve(this.posts[postIndex]);

} else {

resolve(null);

}

}, this.getMockServerResponseDelay());

});

}

async searchPosts(query) {

return new Promise((resolve) => {

setTimeout(() => {

const searchResults = this.posts.filter((post) =>

post.title.toLowerCase().includes(query.toLowerCase())

);

resolve(searchResults);

}, this.getMockServerResponseDelay());

});

}

}