Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

Розробка Web-застосунку «Відслідковування годин відпрацювання»

Текстова частина до курсової роботи
за спеціальністю «Комп'ютерні науки» - 122

Керівник курсової роботи
Старший викладач
Борозенний С.О
_____
(Підпис)
"___" _____ 2020 року
Виконав студент КН-3
Кочмар В.В
"___" _____ 2020 року

Київ 2020

ЗМІСТ

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри мультимед. систем,

канд.ф.-м.н.

_____О. П. Жежерун

(підпис)

„____”_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Кочмару Вадиму факультету інформатики 3 курсу

ТЕМА: Розробка web-застосунку «Відслідковування годин відпрацювання»

Вихідні дані:

- Web-застосунок, розроблений з використання фреймворків ASP.NET Core та React.

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

1.Опис предметної області

2.Проектування та розробка

3. Опис системи

Висновки

Список джерел

Додатки(за необхідністю)

Дата видачі „___” _____ 2019 р. Керівник _____

Завдання отримав _____

Тема: Розробка web-застосунку «Відслідковування годин відпрацювання»

Календарний план виконання роботи:

№ п/п	Назва етапу курсового проекту	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	Листопад 2019 р.	
2.	Огляд літератури та джерел за темою роботи.	Листопад-грудень 2019 р.	
3.	Ознайомлення з фреймворками.	Грудень-січень 2020 р.	
3.	Створення backend частини застосунку.	Січень-лютий 2020 р.	
4.	Створення frontend частини застосунку.	Лютий 2020 р.	
5.	Написання пояснювальної роботи.	Лютий-березень 2020 р.	
6.	Створення слайдів для доповіді та написання доповіді.	Березень 2020 р.	
7.	Надання роботи керівнику на перевірку.	Березень 2020 р.	
8.	Корегування роботи	Березень 2020 р.	
8.	Остаточне оформлення слайдів та роботи	Березень-квітень 2020 р.	
10.	Захист курсової роботи	18 Травня 2020 р.	

Студент _____

Керівник _____

“_____” _____

Анотація

У даній курсовій роботі розглядається та аналізується процес відслідковування годин відпрацювання студентами, а також незручності та складнощі пов’язані з цим процесом. Розроблено web-застосунок, що спрощує даний процес. Описується процес створення застосунку, його архітектуру та технології які було застосовано.

ВСТУП

Студенти НаУКМА при закінченні бакалаврської чи магістерської програм мають змогу отримати “Диплом академічної спільноти НаУКМА”, за умови виконання умов Положення про залучення студентів до розбудови НаУКМА. Одна з умов – це відпрацювання студентом 200 годин щороку. За бакалаврат в сумі має бути 800 годин.

Щоб отримати інформацію про кількість відпрацьованих годин студентам потрібно відвідати Центр кар’єри та працевлаштування студентів та випускників НаУКМА. Наразі студенти не мають змоги перевірити кількість відпрацьованих годин онлайн в будь який час та з будь якого місця.

Також є інші незручності, з якими зіштовхнулись голови студентських організацій. Наприклад, для зарахування годин від студентських організацій – голови студентських організацій мають подавати текстові звіти до керівника центру кар’єри та працевлаштування, який потім додає їх вручну до бази даних, але ніякої електронної системи для перегляду та передачі відпрацьованих годин не має.

Метою даної курсової роботи є проектування та створення електронної системи відслідковування годин відпрацювання, яка б дала змогу студентам продивлятись статистику відпрацьованих годин, а головам студентських організацій і співробітникам центру кар’єри та працевлаштування додавати відпрацьовані години в онлайн режимі.

Текстова частина роботи складається з трьох розділів.

У першому розділі йдеться про опис предметної області, збір та аналіз фактів, а також розглядаються вимоги майбутніх користувачів системи для відслідковування годин відпрацювання.

У другому розділі описано процес проектування та розробки електронної системи: розробка ER-моделі бази даних та архітектури застосунку. Також розглянуто стек технологій та інструменти, які використовуватимуться при розробці системи.

Третій розділ присвячено опису даної системи та інструкції з використання для користувачів.

Розділ 1. Опис предметної області

1.1 Загальна інформація

Кожен студент Національного університету Києво-Могилянська Академія починаючи з першого року навчання має змогу за 4 роки бакалаврської програми відпрацювати 800 годин, бравши участь у розбудові Києво-Могилянської Академії задля виконання однієї з умов корпоративної угоди, яка дає можливість отримати “Диплом академічної спільноти НаУКМА”.

Для отримання годин відпрацювання студенти можуть приймати активну участь у розбудові факультету та університету.

Наприклад, кожного року проводиться для першокурсників Вечір знайомств, прийнявши участь в якому учасники отримують n-ну кількість годин. Студенти, які мають високі успіхи у навчанні, а саме середній рейтинговий бал більше 91 за рік отримують 200 годин.

Ще одним з можливих варіантів як можна відпрацювати години - є студентські організації. У Києво-Могилянській Академії є багато студентських організацій, в які студенти мають змогу вступити. Будучи членом організації та виконуючи певну роботу можна також закривати години. Наприклад будучи членом студентської організації Fido можна приймати участь у розробці застосунків для електронного університету або займатися організаційною роботою - організовувати лекції, воркшопи та хакатони, за що також отримувати години.

Години фіксуються в Центрі кар’єри та працевлаштування студентів та випускників НаУКМА. Для отримання інформації про те скільки годин має студент та скільки залишилось відпрацювати потрібно відвідати центр працевлаштування та дізнатися інформацію у співробітника.

Для фіксування годин, голови студентських організацій в кінці навчального року подають інформацію у вигляді таблиці до Центру кар'єри та працевлаштування, де співробітник у ручному форматі заносить інформацію до бази даних.

Система яку має університет наразі може мати такі проблеми та незручності:

- Незручний процес отримання інформації про поточну кількість відпрацьованих годин;
- Труднощі з додаванням годин від студентських організацій;
- Втрата годин;
- Некоректне додавання годин;

Головною з вище перерахованих проблем – є процес перегляду кількості власних годин студентами.

1.2 Збір фактів

Для збору фактів було проведено співбесіди зі студентами та головами студентських організацій, а також було проведено аналіз та дослідження ринку на наявність аналогічних електронних систем.

1.2.1 Аналіз проведених співбесід

При проведенні співбесід було отримано такі побажання та поради:

- Система у вигляді веб-застосунку;
- Авторизація за допомогою студентського квитка;
- Пошук певного студента при додаванні годин;

- Історія додавання годин;
- Інформативна сторінка зі статистикою;
- Адаптивний застосунок;

На основі проведених співбесід можна зробити висновок, що така система користувалась би попитом у студентів. Завдяки співбесідам було отримано деякі побажання стосовно системи від студентів.

1.2.2 Дослідження

Було проаналізовано наявність аналогічних електронних систем, зокрема таких університетів, як Київський політехнічний університет ім. Ігоря Сікорського, Київський національний університет ім. Тараса Шевченка та Національний авіаційний університет. В результаті аналізу не було знайдено жодної аналогічної електронної системи.

Для деякого аналізу, можна взяти дві електронні системи – Система автоматичного запису на дисципліни та електронна система поселення в гуртожиток, що наявні в Національному університеті Києво-Могилянська Академія. САЗ полегшує процес запису студентів на дисципліни, але також має деякі недоліки:

- На одну дисципліну в якій обмежена кількість місць може записатись більше студентів, ніж обмеження;
- При основному етапі запису, через велику кількість запитів система “падає”;
- Не вірне фільтрування дисциплін;
- Не дуже зручний користувацький інтерфейс;

Електронна система поселення в гуртожиток дає змогу студентам поселись онлайн, без купи паперів, що також значно спрощує процес. Але дана система також має недоліки:

- Помилки в верстці;
- Через неправильну логіку можна помінятися місцем проживанням з людиною, що мешкає в твоїй же кімнаті;

Можна зробити висновок, що хоч і не знайдено аналогічної системи, але досвід використання системи запису на дисципліни та системи електронного поселення в гуртожиток показує, що навіть за наявності проблем та недоліків такі системи виправдовують себе та спрощують процеси для яких їх було розроблено.

1.3 Технічні вимоги

1.3.1 Опис групи користувачів електронної системи

Система буде мати таких користувачів:

- Співробітник центру працевлаштування;
- Голова студентської організації;
- Студент;
- Адміністратор;
- Деканат;

1.3.2 Технічні вимоги користувачів до функціональності системи

Співробітник центру працевлаштування повинен мать можливість вирішувати такі задачі:

- Реєструвати студентів в системі;
- Додавати години;
- Підтверджувати години від СО;
- Проглядати інформацію по студентам;

Голова студентської організації повинен мати можливість вирішувати такі задачі:

- Додавати студентам години від СО, але за умови, що запит підтвердив керівник центру працевлаштування;
- Переглядати статус власної кількості годин;

Студент повинен мати можливість вирішувати такі задачі:

- Переглядати статистику власних години;

Адміністратор повинен мати можливість вирішувати такі задачі:

- Реєструвати користувачів в системі;
- Додавати години;
- Віднімати години;
- Підтверджувати години від СО;
- Проглядати інформацію по студентам;

Деканат повинен мати можливість вирішувати такі задачі:

- Додавати години студентам свого факультету;

1.4 Висновок

У розділі було проаналізовано предметну область, проведено співбесіди зі студентами та проаналізовано ринок на наявність аналогічних систем. В результаті проведених співбесід було отримано деякі побажання від студентів. При аналізі ринку на наявність аналогічних систем такої

системи не було знайдено, але при опитуванні студентів така б система користувалася б попитом в НаУКМА. Також у розділі було описано групу користувачів та технічні вимоги до користувачів.

Розділ 2. Проектування та розробка

Електронна система для відслідковування годин відпрацювання буде являти собою веб-застосунок, яким можна буде користуватись з будь якого девайсу, наприклад чи то з комп’ютера чи телефона.

Основними перевагами веб застосунків є:

- їх не треба встановлювати й отже вони не займають місце на пристрої;
- можна користуватись з будь якого пристрою, який підключений до мережі Інтернет;

Розробку системи поділено на дві частини – розробка сервера та розробка клієнта. Для розробки серверу було обрано такі інструменти: ASP.NET Core, MSSQL, Entity Framework Core.

Для розробки клієнту – React, @material-ui .

2.1 Розробка сервера

2.1.1 Використані інструменти для розробки сервера

Сервер написано на мові C# з використанням фреймворку для веб-застосунків ASP.NET Core.

2.1.1.1 ASP.NET Core

ASP.NET Core – сучасний фреймворк, що був розроблений та підтримується Microsoft, який призначений для створення різних веб-

застосунків починаючи від невеличких веб-сайтів до великих веб-застосунків.

Однією з важливих переваг ASP.NET Core є те, що він є кросплатформенним і може бути розгорнутим на таких операційних системах, як Windows, Mac та Linux. Таким чином, на ASP.NET Core можна створювати кросплатформенні додатки, які можуть бути розгорнуті на різних операційних системах.

При створенні нового проекту ASP.NET Core є можливість вибрати різні шаблони для додатку: API, web-app MVC, Angular, React.js. Для розробки застосунку для відслідковування годин відпрацювання було обрано шаблон API, тому що сервер використовує принципи REST.

2.1.1.2 MS SQL Server

ASP.NET Core дає можливість працювати з різними популярними СКБД, наприклад MySQL, MSSQL, PostgreSQL, але під час написання електронної системи використовувалась система керування базами даних MS SQL Server.

СКБД – програмне забезпечення, яке дозволяє користувачам створювати, підтримувати та здійснювати контрольований доступ до бази даних

SQL Server –популярна система керування реляційними базами даних від Microsoft. Дана СКБД використовує реляційну модель бази даних.

Основними характеристиками MS SQL Server є:

- Висока продуктивність;
- Безпека;
- Простота;
- Можливість працювати в хмарному середовищі;

2.1.1.3 Entity Framework Core

ASP.NET Core дає можливість працювати з ORM. Компанія Microsoft має ORM Entity Framework для своєї платформи .NET.

ORM(Object-Relational Mapping) – це так звана обертка над якоюсь реляційною базою даних, яка дозволяє писати запити до бази даних використовуючи об’єктно-орієнтовану парадигму. Тобто використовуючи ORM розробник працює з базами даних на рівні з більш високим рівнем абстракції. Якщо на фізичному рівні розробник працює з таблицями, індексами, первинними ключами і т.д., то на концептуальному рівні розробник працює з об’єктами. Це значно пришвидшує процес розробки, тому що ORM спрощує процес написання коду.

Основними перевагами ORM є:

- менше коду;
- більша продуктивність запитів;
- простота;

Головною концепцією ORM є поняття сутності. Сутність визначає набір даних, які зв’язані з певним об’єктом. Тому розробник працює з об’єктами, а не з таблицями. Приклад з система: сутність описує Студента і було виділено такі властивості, як повне ім’я студента, факультет, курс, електронна пошта, студентська організація.

Те, що відрізняє Entity Framework від інших ORM – є використання запитів LINQ для вибірки даних з бази даних. LINQ – це технологія від Microsoft, яка додає синтаксис для запитів, який дуже схожий на SQL. Розробник може формувати різні запити за допомогою LINQ для вибірки об’єктів, а Entity Framework при виконанні запиту перетворює вираз LINQ в вираз, який є зрозумілим для системи керування базою даних, в вираз SQL.

2.1.1.4 Postman

Postman – це зручний HTTP-клієнт для тестування API. За допомогою Postman можна створювати HTTP запити. Postman призначений для перевірки запитів з клієнта на сервер і отримання відповіді від серверу. Так як спочатку було розроблено сервер, то саме для перевірки коректності API й було застосовано Postman.

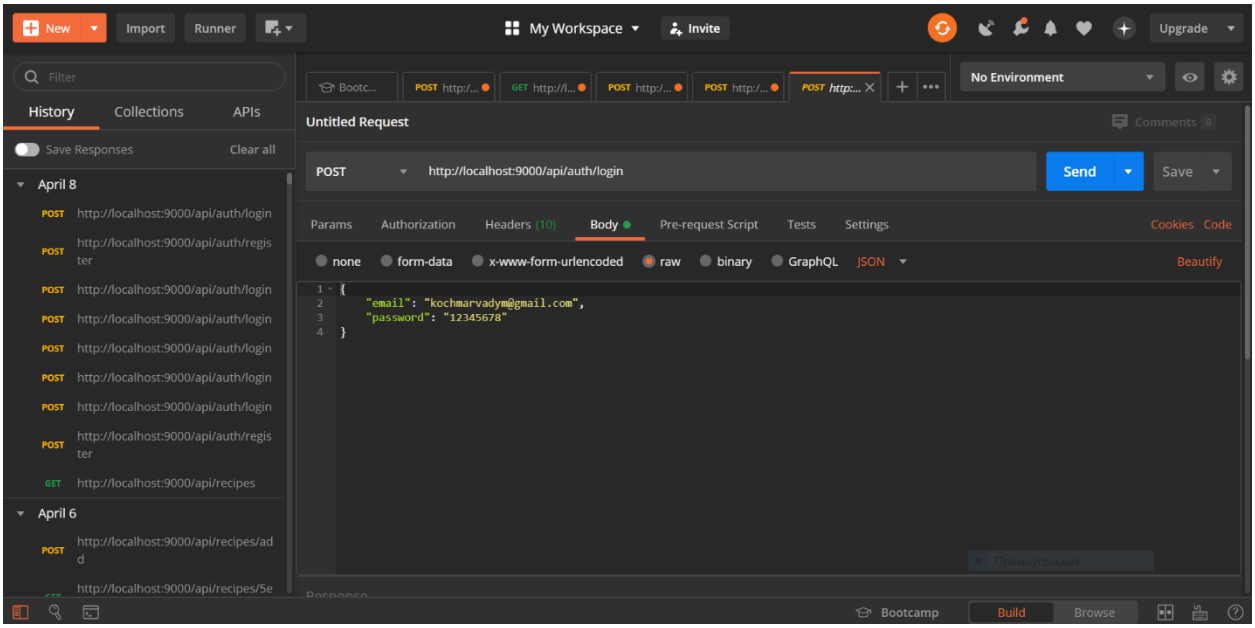


Рис 2. 1 Створення запиту в Postman

2.1.2 ER-модель бази даних

ER-модель – інформаційна модель концептуального рівня ‘’сутність-зв’язок’’.

Нижче подано ER-модель бази даних з сутностями та зв’язками.

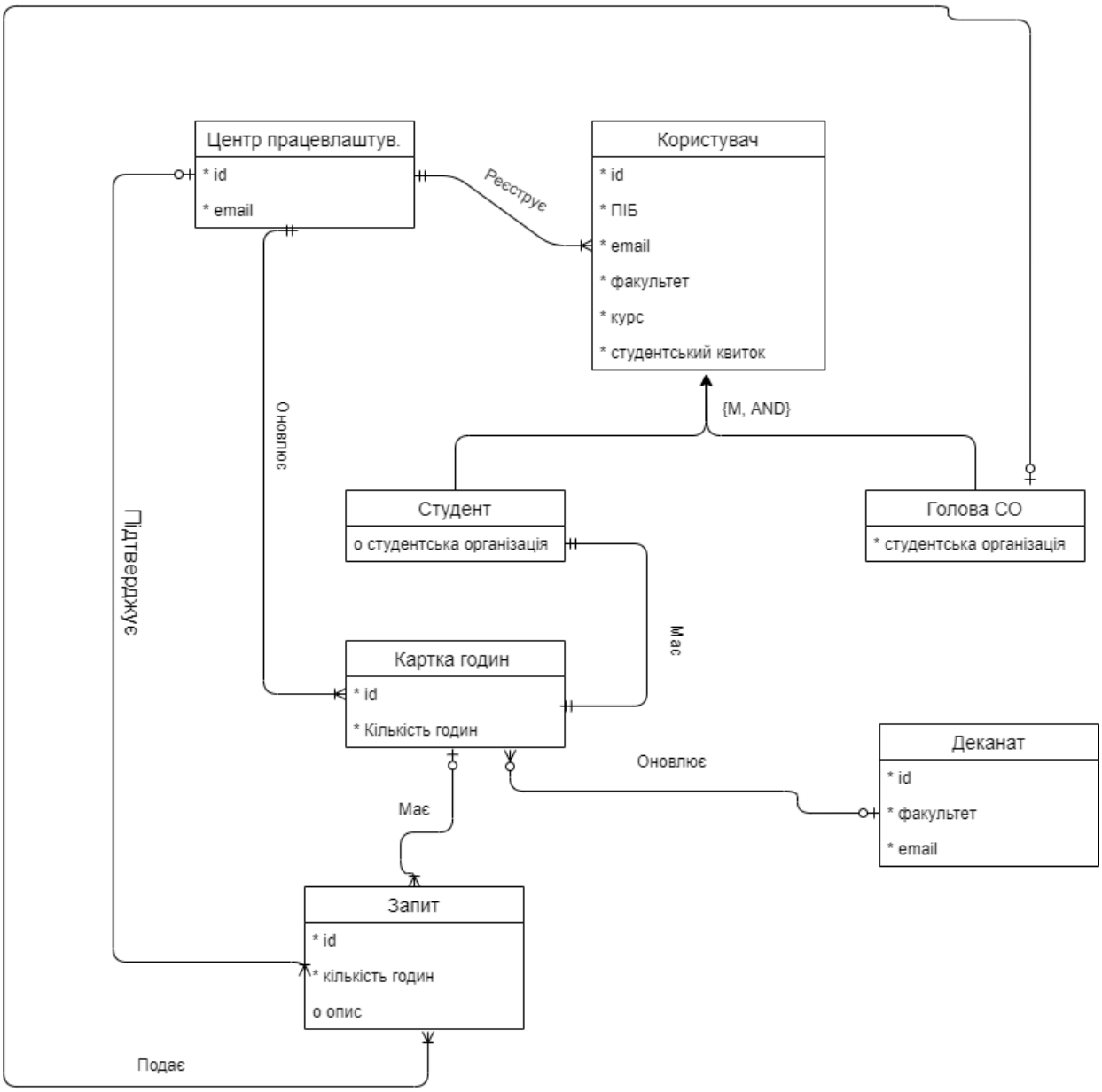


Рис 2. 2 ER-модель застосунку

2.1.3 REST API

Сервер та клієнт комунікують один з одним використовуючи REST API. Rest API часто використовують при розробці Single Page

Applications(SPA), які використовують фреймворк для розробки клієнтської частини – React, який було використано для розробки фронтенду.

API – це інтерфейс за допомогою якого застосунки комунікують між собою, в даному застосунку – frontend комунікує з сервером(backend).

REST – це принципи для доступу до даних в застосунку у вигляді зв’язаних об’єктів та колекцій. Спілкування відбувається за HTTP протоколом. Особливість REST є те, що сервер не запам’ятовує стан клієнта між запитами. Одним з найбільш важливих принципів є те, що має бути обов’язково клієнт-серверна архітектура.

Так як спілкування відбувається за HTTP, то взаємодія з сервером відбувається за допомогою методів HTTP:

- GET – для отримання даних;
- POST – для додавання нових даних;
- PUT – для оновлення даних;
- DELETE – для видалення даних;

В ASP.NET Core для методів, що відповідають для обробки запитів створюється клас з атрибутов [ApiController] та з атрибутом маршрута [Route(“”)], який називається api-контролером. Для вказання типу запиту використовуються спеціальні атрибути [HttpGet], [HttpPost], [HttpPut], [HttpDelete], які застосовуються до методів контролеру.

Нижче подано таблицю з API запитами застосунку для відслідковування годин відпрацювання:

Таблиця 1.1. Таблиця запитів

Запит	Відповідь

POST /api/svgv/register
 	
{
 	
“StudNum” – required string
 	
“Password” – required string
 	
“Email” – required string
 	
“FullName” – required string
 	
“Faculty” – required string
 	
“Course” – required integer
 	
“Role” – required string
 	
}
	{
	
	“Ok” – required Boolean
	
	“Errors” – not required list
	
	}

POST /api/svgv/login
 	
{
 	
“Login” – required string
 	
“Password” – required string
 	
}
	{
	
	“Token” – required string
	
	“Errors” – not required list
	
	}

GET /api/svgv/getStudents
 	
{
 	
}
	{
	
	“Students” – required list
	
	“Errors” – not required list
	
	}

PUT /api/svgv/addHours
 	
{
 	
“UserId” – required string
 	
“Hours” – required integer
 	
}
	{
	
	“Ok” – required Boolean
	
	“Errors” – not required list
	
	}

GET /api/svgv/card/{UserId}
 	
{
 	
}
	{
	
	“Id” – required string
	
	“Hours” – required integer
	
	“UserId” – required string
	
	“Errors” – not required list
	
	}

GET /api/svgv/user/{UserId}
 	
{
 	
}
	{
	
	“FullName” – required string
	
	“Faculty” – required string
	
	“Course” – required int
	
	“StudOrg” – not required string
	
	“Errors” – not required list
	
	}

POST /api/svgv/SendRequest { “Hours” – required integer	{ “Ok” – required Boolean “Errors” – not required list

“FromUserId” – required string “ToUserId” – required string }	}

PUT /api/svgv/changeStatus
 	
{
 	
“Status” – required integer
 	
“Hours” – required integer
 	
“RequestId” – required string
 	
“FromUserId” – required string
 	
“ToUserId” – required string
 	
}
	{
	
	“Ok” – required Boolean
	
	“Errors” – not required list
	
	}

GET /api/svgv/getRequests
 	
{
 	
}
	{
	
	“Requests” – required list
	
	“Errors” – not required list
	
	}

GET /api/svgv/myRequests/{UserId}
 	
{
 	
}
	{
	
	“Requests” – required list
	
	“Errors” – not required list
	
	}

2.1.4 Безпека застосунку

Одним із головних аспектів застосунку є – безпека інформації користувачів. Так як застосунок має справу з користувацькими даними: пароль, студентский і т.д., то саме їх захист зайняв важливу частину проектування системи.

2.1.4.1 Хешування паролів

Під час авторизації користувач використовує пароль. Важливим є те, що небезпечно зберігати пароль в звичайному вигляді в базі даних. Пароль має зберігатися у вигляді хешу, який створюється за допомогою криптографічної функції. При реєстрації пароль хешується за допомогою хеш функції та зберігається в базі даних. А при наступних авторизаціях, користувач вводить свої логін та пароль. Застосунок знаходить користувача в базі даних за ідентифікатором. Якщо користувача було знайдено, то введений пароль при авторизації хешується хеш функцією та порівнюється з хешом, який зберігається в базі даних даного користувача Якщо хеші співпадають, то пароль було введено вірний і користувач авторизується в системі, а якщо – не співпадають, то користувач ввів невірний пароль.

Саме так це й було зроблено в додатку. ASP.NET Core має клас PasswordHasher. Даний клас використовує хеш-функцію SHA-256. Для хешування паролю використовується функція HashPassword(User, password). Для порівняння хеша при авторизації використовується функція VerifiedHashedPassword(User, inputPassword, storedHashedPassword).

2.1.4.2 JWT

В проекті виконується авторизація на основі JWT. JWT – JSON Web Token, який генерується на сервері при кожній новій авторизації користувача та є унікальним. При кожному наступному запиті клієнт передає токен через header-запиту, який перевіряється на стороні серверу. Якщо токени співпадають, то запит виконується успішно, а в іншому випадку буде помилка. Він використовується для безпечної передачі інформації між двома сторонами(сервером та клієнтом). В даного токена є час існування, при закінченні якого токен стає не дійсним. Для вирішення цього можна використовувати Refresh Token, який використовується для створення нового JWT. В ASP.NET Core робота з JWT виконується за допомогою пакету Nuget AspNetCore.Authentication.JwtBearer.

2.2 Розробка клієнтської частини

2.2.1 Використані інструменти для розробки клієнту

Для розробки клієнту було використану бібліотеку для побудови користувацьких інтерфейсів React та material-ui – бібліотеку компонентів React.

2.2.1.1 Бібліотека React

React – це одна з найбільш популярних javascript-бібліотек, розроблена Facebook, для побудови користувацьких інтерфейсів. В React було втілено нову ідею стан(state) інтерфейсу, який зберігається в дереві вузлів. Кожен вузол дерева – це компонент інтерфейсу, який описується певною функцією. Дерево називається віртуальний DOM, який потім перетворюється в HTML.

При зміні стану будується новий віртуальний DOM і сам React перетворює й застосовує зміни до реального HTML тільки, те що змінилось – різницю. Всі інші вузли де одні й ті ж самі дані не будуть перемальовуватись. Такий підхід значно підвищив швидкість оновлення сторінки при зміні та оновленні даних.

Ще однією особливістю React є JSX. JSX – це синтаксичний цукор над мовою програмування JavaScript, який спрощує роботу розробникам – він дуже схожий на HTML. Головною задачею JSX - є створюти React компоненти. Компоненти в React - це функції JavaScript, які зберігають стан та повертають React елементи, які потім з’являються на веб сторінці.

Варто звернути увагу на те, що React має дуже гарну документацію з необхідним матеріалом для початку розробки та з описом основних особливостей цієї бібліотеки, з якою легко розібратися. Під час розробки системи основним джерелом була саме офіційна документація.

2.2.1.2 Material-UI

Зовнішній вигляд застосунку відіграє основну роль, так як користувач взаємодіє саме з frontend частиною, тому для цього було обрано популярний в світі frontend бібліотеку компонентів інтерфейсу Material-UI.

Material-UI - це бібліотека React компонентів для користувацького інтерфейсу. Ідея бібліотеки полягає в тому, що якщо розробнику треба в проекті якийсь компонент, наприклад меню, яке випадає, то розробник імпортує цей компонент з бібліотеки, не створюючи його з “нуля”. Це значно пришвидшує розробку. Стилі всіх компонентів можна змінювати, як за допомогою звичного CSS так і за допомогою JSS, який дозволяє використовувати JavaScript для опису стилів.

2.2.2 Сторінки та компоненти

Для сторінок застосунку було обрано кольорову гамму – синій та білий як основні кольори.

На даному етапі було визначено, які сторінки будуть на сайті та які компоненті будуть на сторінках:

- Сторінка авторизації;
- Сторінка зі списком студентів;
- Сторінка реєстрації користувачів;
- Сторінка з запитами;
- Сторінка зі статистикою годин певного користувача;

2.2.3 Звернення до API

Для роботи з запитами на клієнті було створено хук useHttp, який використовує нативний API браузера fetch, але у форматі хуків. Fetch має інтерфейс для виконання запитів та отримання відповіді HTTP. Він дає метод fetch(), який дозволяє отримувати дані по мережі.

В хуці є функція request, яка робить запит на сервер. Ця функція приймає в себе url, метод HTTP, тіло запиту, а також хедери. Саме в цій функції викликається браузерний метод fetch, який першим параметром приймає url, а другим параметром – набір опцій(метод, тіло та хедери).

```
const request = useCallback( callback: async(url, method='GET', body=null, headers = {}) => {
    setLoading(true);
    try{
        if(body){
            body = JSON.stringify(body);
            headers['Content-Type'] = 'application/json';
            headers['Access-Control-Allow-Origin'] = '*';
        }

        const response = await fetch(url, init: {method, body, headers});
        const data = await response.json();

        if(!response.ok){
            throw new Error('Щось пішло не так');
        }

        setLoading(false);

        return data;
    }catch (e) {
        setLoading(false);
        setError(e.message);
        throw e;
    }
}, deps: []);
```

Рис 2. 3 Функція request

2.3 Висновок

У розділі описано, які інструменти було використано для розробки серверної та клієнтської частин. Висвітлено їх основні переваги, в першу чергу для розробника. Подано розроблену ER-модель бази даних та таблицю з необхідними API запитами. Було проаналізовано комунікування сервера та клієнта за допомогою REST API, описано за допомогою яких методів це відбувається. При розробці було звернено увагу на хешування паролів та використання токенів для авторизації, що підвищує безпечність застосунку.

Розділ 3. Опис системи

3.1 Загальний опис системи

Дана система була розроблена для студентів та співробітників університету, щоб полегшити задану задачу – відслідковування годин відпрацювання, а саме, щоб студенти могли в зручний для себе час та з будь якого пристрою переглянути години, а голови студентських організацій та центр працевлаштування могли додавати години відповідним студентам. Отримана система наприкінці розробки справляється з заданими задачами та має весь необхідний функціонал для цього.

В системі наявні п’ять ролей користувачів: студент, голова студентської організації, центр працевлаштування, деканат та адміністратор. Користувачі виконують всі описані вимоги в розділі 1.3.2. В залежності від ролі, користувач має різний доступ до сторінок сайту та різне наповнення сторінок.

Дизайн сторінок розроблено в кольорах, які легкі на сприйняття. В дизайні не має занадто яскравого кольору, який зливається з текстом чи іншими частинами сторінки. Розмір шрифту оптимальний для читання. Загалом дизайн є мінімальним. Інтерфейс web-застосунку є легкозрозумілим та інтуїтивним. Користувачам буде не складно зрозуміти, наприклад навіщо якась кнопка потрібно. Навігація між сторінками добре зрозуміла для всіх ролей. Також верстка на сторінках є адаптивною. Це говорить про те, що web-застосунком можна користуватись з різних пристроїв – як з комп’ютера так і планшета чи мобільного телефона з різним розширенням екрана.

3.2 Інструкція з використання

Для використання системи, кожен користувач повинен увійти до системи, тому перш за все користувачі потрапляють на сторінку входу до системи (Додаток Б). Студенти та голови студентських організацій входять до системи за допомогою номера студентського квитка та паролю, яким є дата народження студента. Центр працевлаштування, деканати та адміністратор авторизуються в системі за допомогою власних ідентифікаторів та паролів.

Центр працевлаштування. Після успішної авторизації користувач, який увійшов, як центр працевлаштування потрапляє на головну сторінку (Додаток В). На цій сторінці міститься три карточки, що ведуть до таких сторінок: реєстрація студентів, додавання годин, перегляд запитів та відповідні описи по кожній з них. При натисканні на якусь з карток відбудеться перехід на ту сторінку, яка зазначене на картці. Перейшовши на сторінку Реєстрація, користувач потрапляє на сторінку з формою (Додаток Г), на якій можна зареєструвати студентів та голову студентської організації. Якщо нового студента було успішно зареєстровано з’явиться повідомлення “Студента було успішно зареєстровано”. Якщо користувач з таким студентським вже існує, або не всі обов’язкові поля було заповнено то з’явиться повідомлення “Щось пішло не так”. При переході на сторінку додавання годин (Додаток Д) співробітник центру працевлаштування зможе додавати години відповідним студентам. На цій сторінці є фільтри за ім’ям, факультетом, курсом та за студентськими організаціями. Можна робити пошук відразу за декількома категоріями, наприклад за факультетом та курсом. При натисканні кнопки Додати години з’являється вікно з полем для вводу бажаної кількості додавання годин. При підтвердженні дії з’явиться повідомлення про успішне додавання годин. Після цього сума годин студента, якому було додано години зміниться. На сторінці запити

співробітник центру працевлаштування спочатку побачить запити, які очікують або підтвердження або скасування (Додаток Е). При натисканні кнопки підтвердити кількість годин що були зазначені на картці будуть додані заданому студенту. Також з'явиться повідомлення, що запит було підтверджено. Якщо натиснути кнопку відмовити, то годин додано не буде. Якщо змінити фільтр на підтверджені, то буде показано запити, які було підтверджено, колір підтверджених запитів синій (Додаток Ж). При зміні фільтру на скасовані буде показано запити, які було скасовано, колір цих запитів – червоний (Додаток И).

Студент. При успішний авторизації студент потрапляє на сторінку зі статистикою годин (Додаток К). На цій сторінці можна побачити кількість відпрацьованих годин, кількість у відсотках, а також кругову діаграму прогресу.

Голова студентської організації. При успішній авторизації голова студентської організації потрапляє на головну сторінку, на якій є три картки, які ведуть до сторінок: години, статистика та запити. На сторінці години голова студентської організації може подати запит на додавання годин обраному студенту. Пошук студента можна виконати за допомогою декількох фільтрів: ім'я, факультет, курс. При натисканні додати години з'явиться форма, в яку потрібно ввести кількість годин, яку бажає додати голова. При успішному підтвердженні з'явиться повідомлення, що запит було успішно додано. На сторінці статистика, голова студентської організації бачить статистика власних годин, як студент. На сторінці з запитами, можна відслідковувати статус запитів, а також фільтрувати їх за статусом: очікують, підтверджені та скасовані.

Деканат. Після авторизації деканат потрапляє на сторінку, де може додавати години студентам свого факультету. Також можна відшукати потрібного студента за допомогою фільтрів. При натисканні кнопки додати

години напроти обраного студента з’являється форма в яку потрібно ввести кількість годин та натиснути підтвердити для додавання годин.

Адміністратор. Після авторизації адміністратор потрапляє на головну сторінку, на якій вказано назви сторінок та опис дії, яка може бути виконана на тій сторінці. На сторінці реєстрація адміністратор може реєструвати користувачів до системи. Окрім студентів та голів студентських організацій, адміністратор може зареєструвати деканат та співробітника центру працевлаштування. На сторінці години, адміністратор може корегувати години обраному студенту. Відшукати потрібного студента можна за допомогою фільтрів. На сторінці запити адміністратор може підтверджувати запити на додавання годин від студентських організацій, а також переглядати історію запитів за допомогою фільтрації по статусу підтверджені, скасовані та очікують.

3.3 Висновок

У розділі описано для яких задач розроблена система й було визначено, що з усіма потрібними задачами вона справляється. Функціонал кожної з ролей відповідає вимогам. Інтерфейс, який було розроблено зручний та зрозумілий для користувачів. Відмічено, що застосунком можна користуватись на будь яких пристроях. Також подано інструкцію з використання системи для всіх ролей, яка допоможе зорієнтуватися під час використання системи.

Висновки

У курсовій роботі йдеться про вирішення проблеми з годинами відпрацювання – їх відслідковування студентами та додавання годин від студентських організацій. Результатом виконання курсової роботи є розроблена електронна система у вигляді web-застосунку, яка допомагає спростити ці процеси. Для підвищення зручності, було прийнято до уваги поради від студентів, що були озвучені під час співбесід. Отримана система може бути використаною для загального користування. За потреби додаткових функцій система може бути легко розширена. До системи під час розробки було додано корисну можливість – це перегляд статусу поданих запитів головою студентської організації. Одне з можливих розширень системи - є впровадження авторизації користувачів за допомогою Office365.

Роблячи висновки, можна сказати, що дана система буде мати достатній попит після впровадження в НаУКМА.

Список джерел

1. Metanit [Електронний ресурс]: https://metanit.com/sharp/
2. Entity Framework Documentation [Електронний ресурс] : https://docs.microsoft.com/en-us/ef/
3. React documentation [Електронний ресурс]: https://reactjs.org/
4. Material-UI documentation [Електронний ресурс]: https://material-ui.com/
5. Introduction to JSON Web Tokens [Електронний ресурс]: https://jwt.io/introduction/

ДОДАТКИ

Додаток А

(Обов’язковий)

Співбесіди зі студентами та головами студентських організацій:

Співбесіда 1:

...

Опитувач: На скільки мені відомо, ти дуже активний студент Києво-Могилянської Академії. Чи зручно тобі наразі дізнаватись скільки

відпрацьованих годин ти вже маєш?

Студент: Чесно кажучи, я навіть точно не знаю скільки маю відпрацьованих годин починаючи з цього навчального року. Останній раз я дізнавався в кінці минулого навчального року. Раз на рік я відвідую центр працевлаштування для отримання цієї інформації. Тому не можу назвати цей процес зручним для студентів.

Опитувач: Як би була електронна альтернатива, щось на зразок електронного запису на дисципліни або електронної системи поселення в гуртожиток, ти б користувався такою системою для відслідковування власних годин?

Студент: О, без сумніву можу сказати, що користувався б. Це досить гарна ідея. Тоді, я зараз зміг би тобі точно сказати скільки відпрацьованих годин маю наразі.

Опитувач: На твій погляд, чим би тобі було зручніше користуватись – веб сайтом чи окремим мобільним додатком?

Студент: Як на мене, то краще б було мати сайт, щоб можна було зайти як з комп’ютера так і з мобільного пристрою, так як це зроблено в системі автоматизованого запису на дисципліни.

Опитувач: Єдине питання яке залишилось в тебе запитати – який вид авторизації був би зручним?

Студент: Мені було б зручніше авторизовуватись за даними студентського, але було б непогано мати можливість також зайти за даними Office 365.

...

Співбесіда 2:

...

Опитувач: Що ти можем сказати про додавання годин відпрацювання від твоєї студентської організації? Чи зручно тобі?

Голова СО: Даний процес займає чимало часу для заповнення шахматки. Також, після цього таблицю потрібно подати до Центру кар’єри та працевлаштування.

Опитувач: Як би був електронний аналог, який би дозволяв тобі додавати години студентам за виконану роботу, чи використовував би ти його?

Голова СО: Так, це б забирало в мене меньше часу, та думаю, що точно було б зручніше.

Опитувач: Які б вимоги до системи ти хотів би бачити не тільки як голова студентської організації, а й як студент?

Голова СО: Перш за все, як голові СО я хотів би мати таблицю зі студентами, по якій я міг би робити швидкий пошук та додавати вибраному студенту певну кількість годин. Також було б добре було бачити історію кому скільки додав.

Як студент – я хочу, щоб в мене був зручний та інформативний dashboard з якого б я легко міг зрозуміти стан свого відпрацьовування.

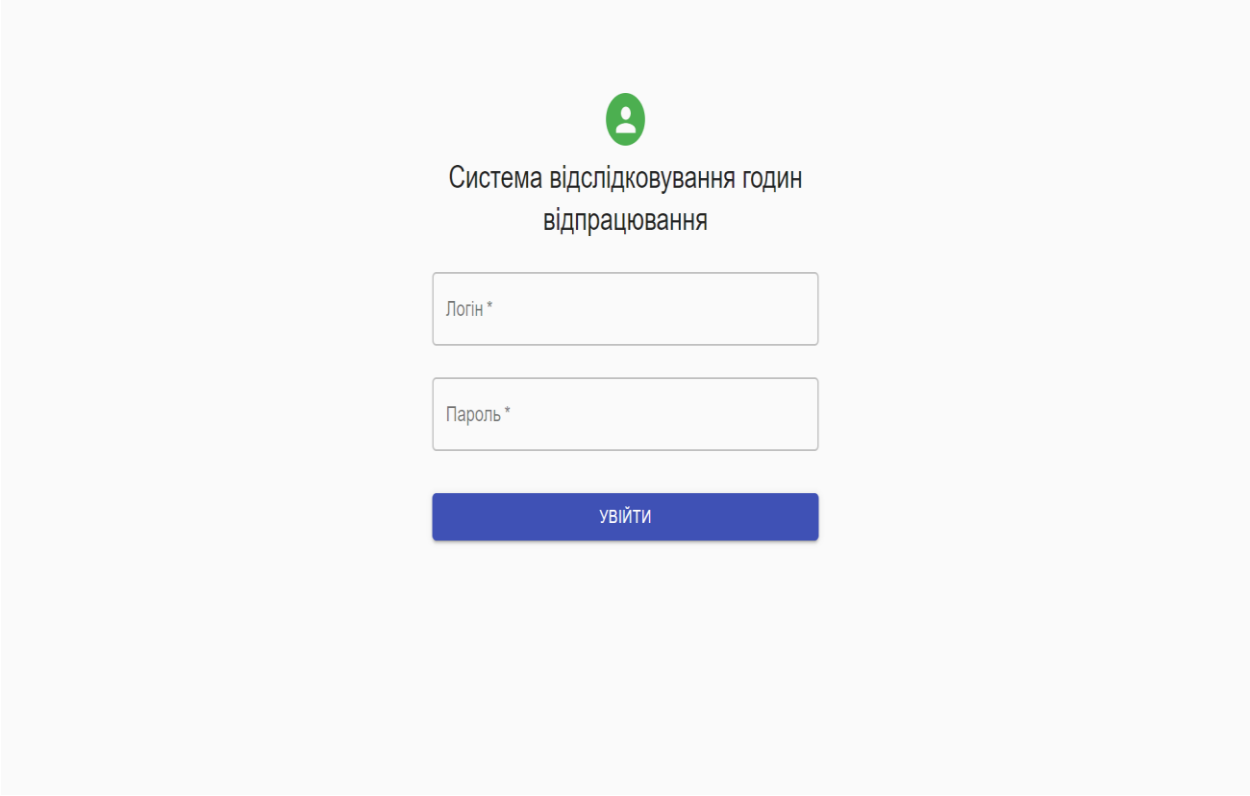
Опитувач: Маю один важливий момент – як би тобі було зручно авторизовуватись?

Голова СО: За допомогою студентського.

...

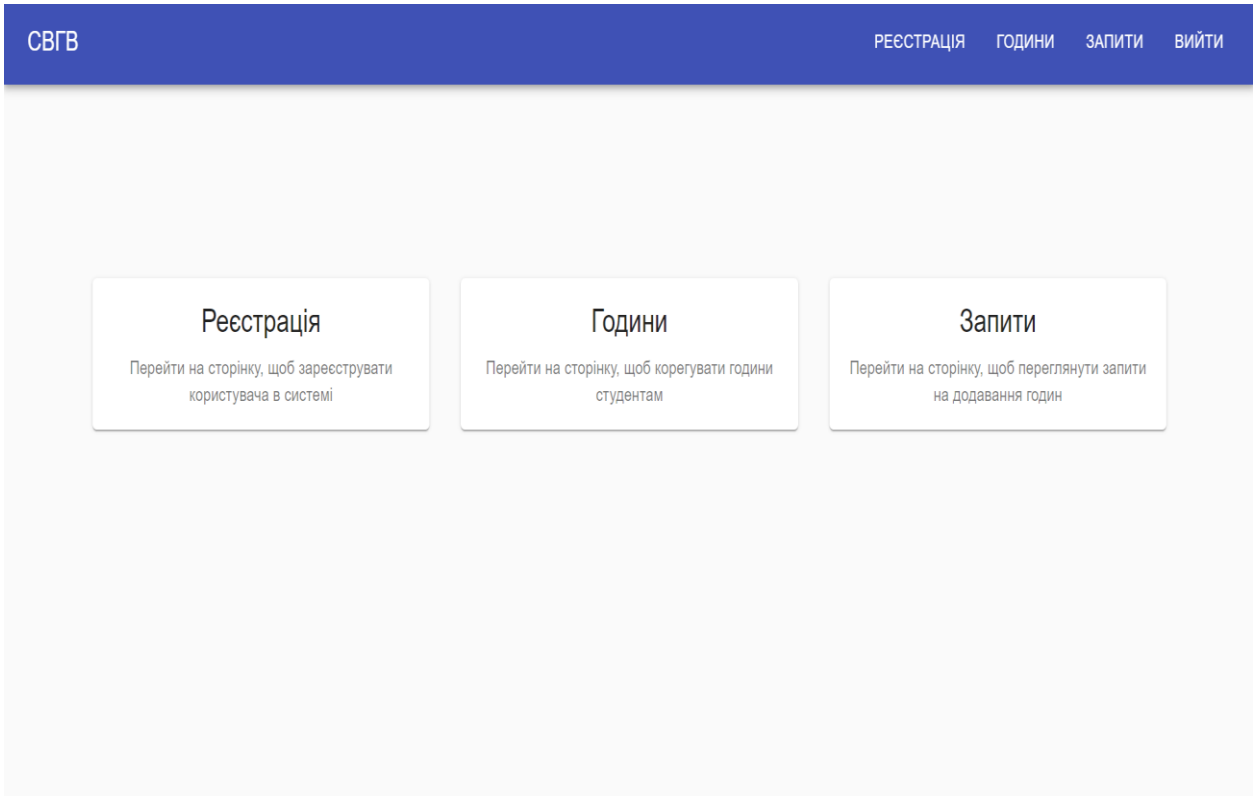
Додаток Б
(Обов’язковий)

Сторінка авторизації



Додаток В
(Обов’язковий)

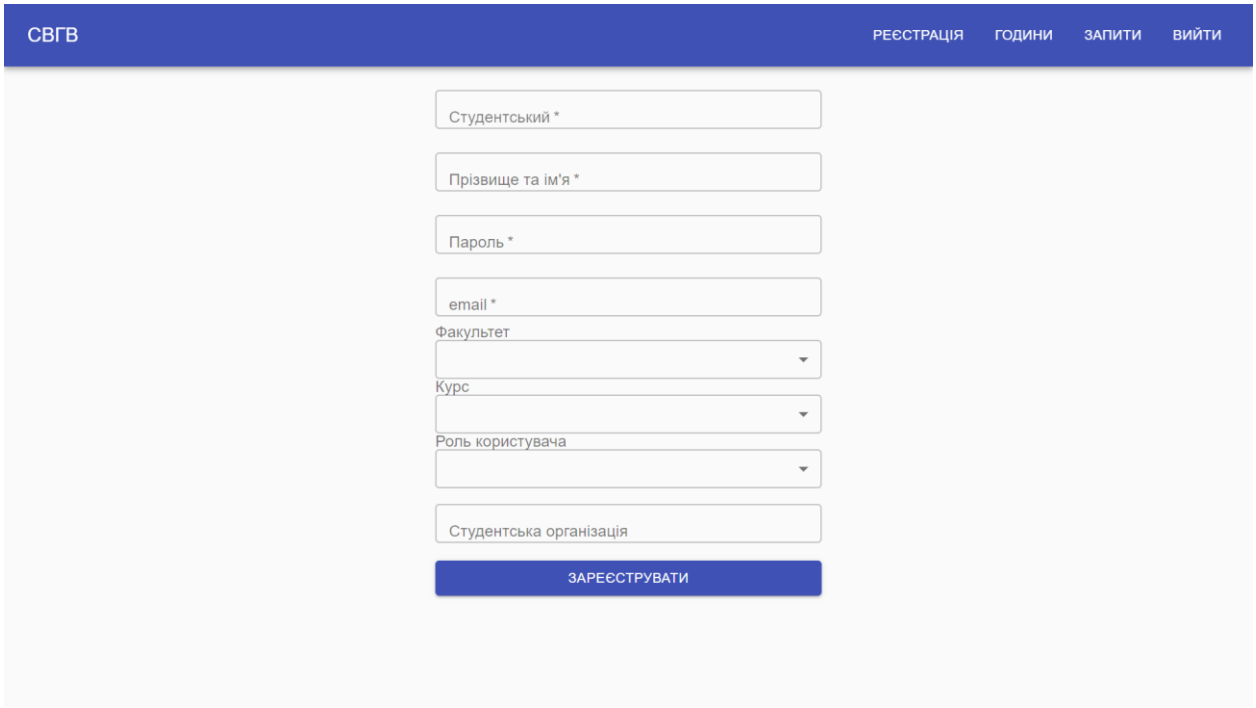
Головна сторінка центру працевлаштування



Додаток Г

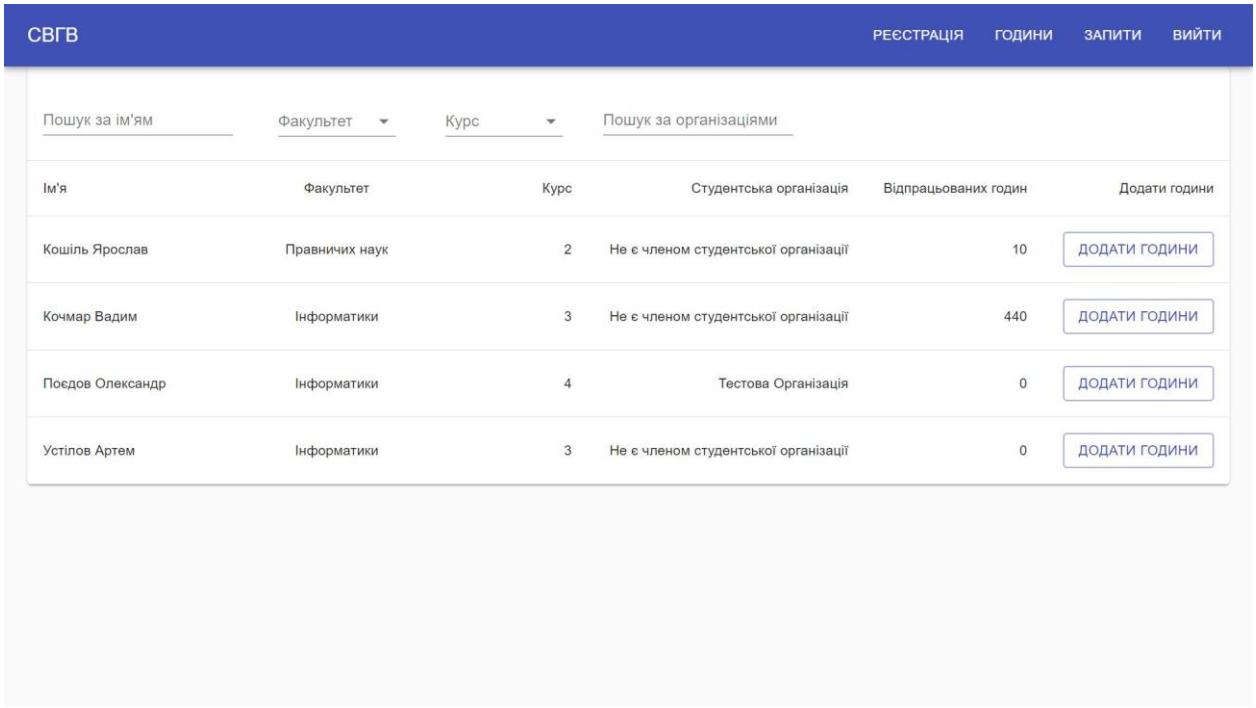
(Обов’язковий)

Сторінка реєстрації студентів



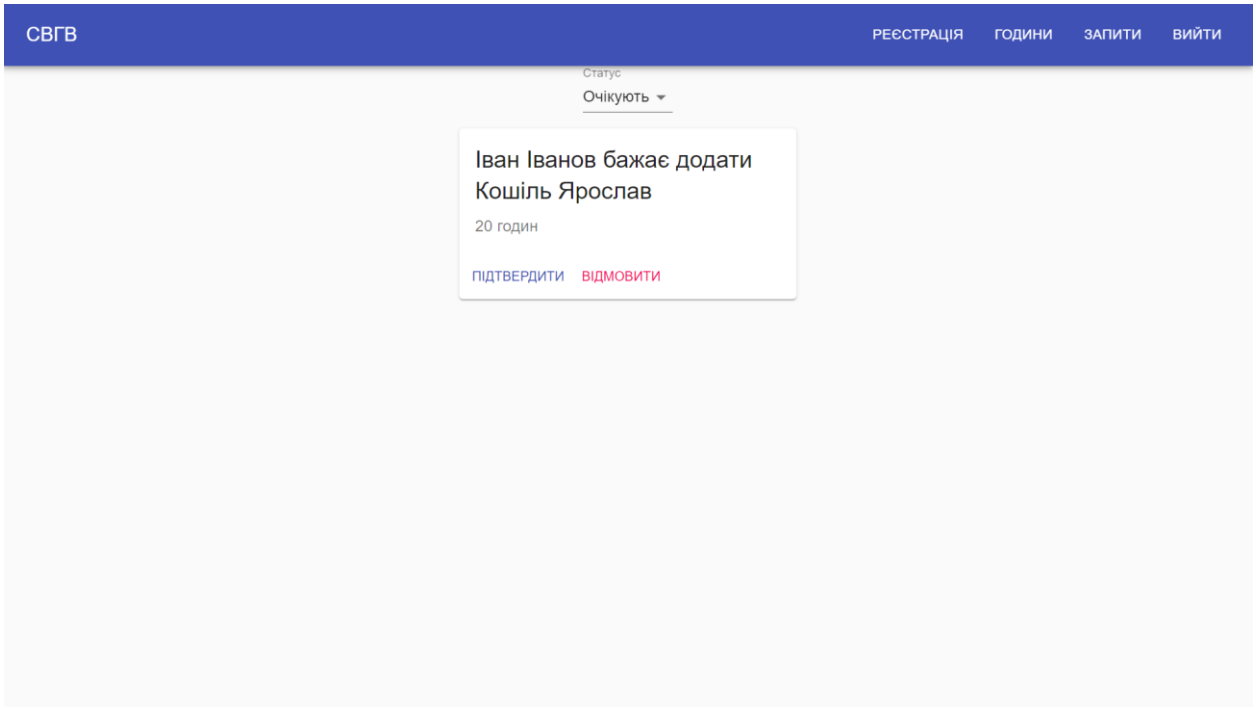
Додаток Д
(Обов’язковий)

Сторінка корегування годин



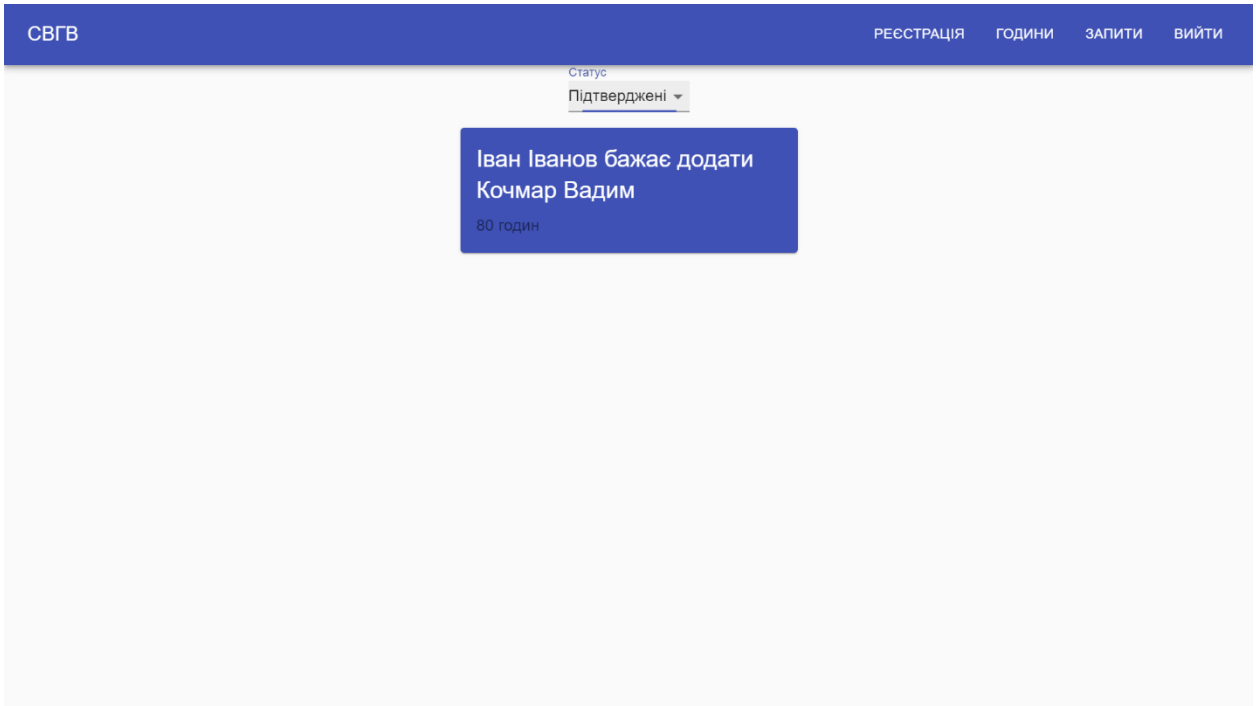
Додаток Е
(Обов’язковий)

Сторінка з запитами, що очікують



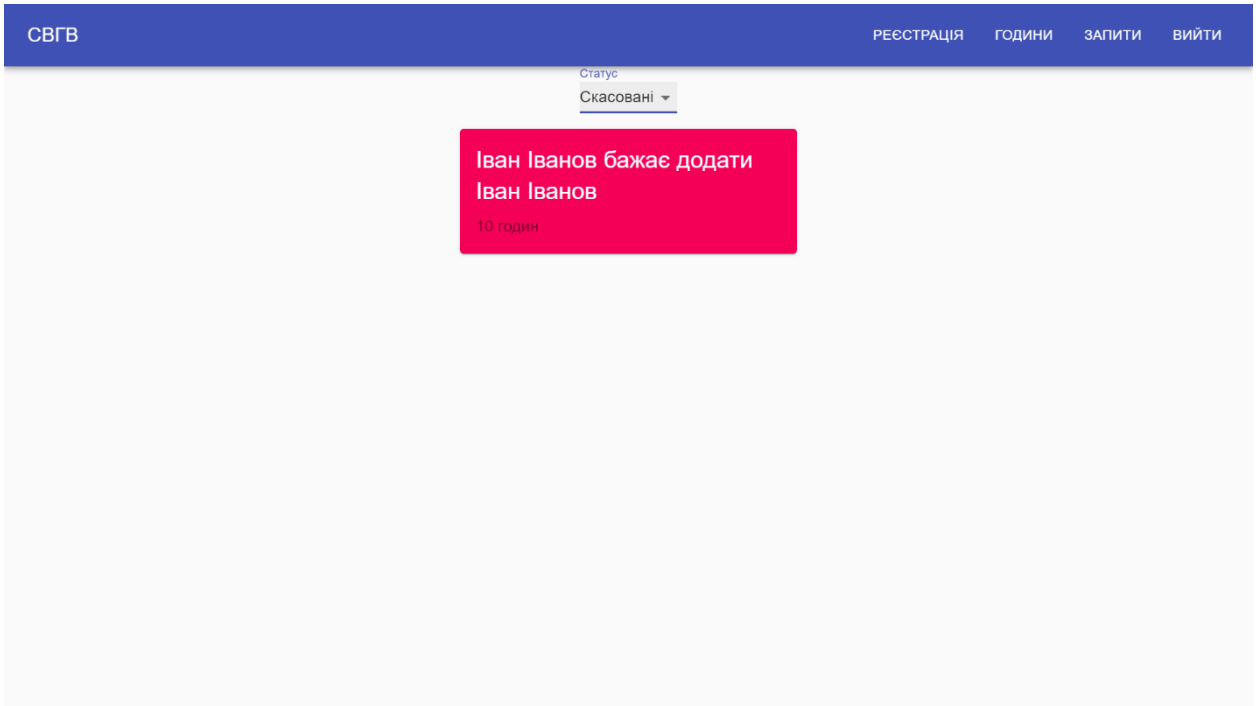
Додаток Ж
(Обов’язковий)

Сторінка з запитами, що підтверджені



Додаток И
(Обов’язковий)

Сторінка з запитами, що скасовані



Додаток К
(Обов’язковий)

Сторінка зі статистикою годин