**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформаційних систем та технологій**

# Індивідуальний дослідницький проєкт

**на здобуття ступеня бакалавра**
**за освітньо-професійною програмою «Інтегровані інформаційні системи»**
**спеціальності 126 «Інформаційні системи та технології»**
**на тему: «Інформаційна система для вибіркових дисциплін»**

Виконав:
студент IV курсу, групи ІА-82
Новак Антон Святославович _____

Керівник:
професор кафедри ІСТ, д.т.н.
Онищенко Вікторія Валеріївна _____

Засвідчую, що у цьому проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2022 року

# Пояснювальна записка
## до індивідуального дослідницького проєкту
## на тему: «Інформаційна система для вибіркових дисциплін»

Київ – 2022 року

**Національний технічний університет України**
**«Київський політехнічний інститут імені Ігоря Сікорського»**
**Факультет інформатики та обчислювальної техніки**
**Кафедра інформаційних систем та технологій**

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інтегровані інформаційні системи»

# ЗАВДАННЯ

**на індивідуальний дослідницький проєкт студенту**
**Новаку Антону Святославовичу**

1. Тема проєкту «Інформаційна система для вибіркових дисциплін», керівник проєкту Онищенко Вікторія Валеріївна, д.т.н.

2. Термін подання студентом проєкту: 15 червня 2022 року

3. Вихідні дані до проєкту: підтримка десктопних та мобільних браузерів, можливість інтеграції з існуючими системами для онлайн-голосування.

4. Зміст пояснювальної записки: загальні відомості, огляд існуючих рішень, розроблення діаграми прецедентів, розроблення UI/UX дизайну, вибір окремих технологій, розроблення схеми бази даних, розроблення структурної схеми, програмна реалізація.

5. Перелік графічного матеріалу: структурна схема, діаграма прецедентів, схема бази даних, схема трьохрівневої архітектури у веб застосунках

6. Дата видачі завдання 1 грудня 2021 року

Календарний план

| № з/п | Назва етапів виконання проєкту | Термін виконання етапів проєкту | Примітка |
|---|---|---|---|
| 1 | Аналіз предметної області | 02.05.22 – 08.05.22 | |
| 2 | Огляд існуючих рішень | 09.05.22 – 15.05.22 | |
| 3 | Розробка діаграми прецедентів, UI/UX дизайну, схеми бази даних та структурної схеми | 16.05.22 – 22.05.22 | |
| 4 | Вибір окремих технологій | 23.05.22 – 29.05.22 | |
| 5 | Розробка фронтенд частини застосунку | 30.05.22 – 05.06.22 | |
| 6 | Розробка бекенд частини застосунку | 06.06.22 – 12.06.22 | |
| 7 | Тестування застосунку | 13.06.22 – 19.06.22 | |
| | | | |
| | | | |

Студент Антон НОВАК

Керівник Вікторія ОНИЩЕНКО

# АНОТАЦІЯ

Новак А. С. Інформаційна система для вибіркових дисциплін. КПІ ім. Ігоря Сікорського, Київ, 2022.

Проєкт складається з восьми розділів та містить 63 с. тексту, 25 рисунків, посилання на 28 літературних джерел, 1 додаток та 4 конструкторських документи.

Ключові слова: інформаційна система, вибіркові дисципліни, ввідна лекція, онлайн голосування.

Об’єктом розробки є інформаційна система для вибіркових дисциплін.

Метою проєкту є розширення функціоналу існуючих систем для електронного голосування українських вишів за рахунок створення спеціалізованої інформаційної системи.

У дослідницькому проекті реалізовано мінімально життєздатний продукт інформаційної системи для вибіркових дисциплін. Розроблено структурну схему з декомпозицією, функціональну схему та схему бази даних. Програмна розробка системи здійснювалась за допомогою мов програмування JavaScript та Node.js. У фронтенд частині системи використовувались бібліотеки React, Redux та Tailwind. У бекенд частині системи використовувалось backend-as-service рішення від Google - Firebase.

Отримані результати можуть використовуватись при розробці та моделюванні систем схожого функціоналу.

# SUMMARY

Novak A. S. Information system for elective subjects. Igor Sikorsky Kyiv Polytechnic Institute, Kyiv, 2022.

The project consists of eight sections and contains 63 pages of text, 25 figures, references to 28 literature sources, 1 attachment and 4 design documents.

Keywords: information system, elective subjects, introductory lecture, online voting.

The object of development is an information system for elective subjects.

The goal of this project is to expand the functionality of existing systems for online voting of Ukrainian universities by creating a specialized information system.

The research project includes an implementation of the minimal value product of an information system for elective subjects. It also includes structural scheme with decomposition, functional scheme and database scheme. Software development of the system was carried out using the programming languages JavaScript and Node.js. The frontend part of the system used the React, Redux and Tailwind libraries. The backend part of the system used a backend-as-service solution from Google - Firebase.

The final results can be used in the development and modeling of systems of similar functionality.

**ЗМІСТ**

# ВСТУП

Сьогодні система університетської освіти зазнає кардинальних трансформацій, які значною мірою пов'язані з бурхливим розвитком інформаційно-комп'ютерних технологій та їхнім стрімким впровадженням у керування навчальним процесом вищих навчальних закладів (ВНЗ) України, а також у системи університетського менеджменту [1]. Одним із критеріїв успішності керування навчальним процесом ВНЗ постає правильна організація процесів навчання. Впровадження новітніх інформаційних технологій в даній галузі дозволяє вирішити одну з ключових проблем організації навчального процесу — задачу планування розкладів занять вищих навчальних закладів. Вибіркові дисципліни як один із елементів стали невід'ємною частиною університетських програм у всьому світі. Довільний пошук в Інтернеті показує, що університети різних країн, різного профілю та різного рівня надають студентам можливість обирати певний обсяг дисциплін.

Вибіркові дисципліни, які вводяться до навчального плану ВНЗ, сприяють загальному розвитку студентів, повніше задовільняють їх освітні запити та обираються ними самостійно із запропонованого переліку, сприяють створенню умов для розвитку особистості й творчої самореалізації кожного студента, а також стимулюють постійне підвищення якості освіти, оновлення її змісту та удосконалення організації навчального процесу ВНЗ.

Так звані вибіркові дисципліни (англ. electives) походять із концепції person-centered teaching (навчання, орієнтоване на особистість), яку в середині XX ст. систематизував американський психолог Карл Роджерс. У її основі лежить принцип, за яким учень або студент бере активну участь у формуванні своєї освітньої траєкторії, обираючи, що вчити, як вчити та як оцінювати власні знання. Викладач виконує функції не джерела знань, а провідника та фасилітаора під час руху по цій траєкторії. Цей рух супроводжується постійним самоаналізом

та тісною комунікацією викладача зі студентом чи учнем, і сама оцінка знань тяжіє більше до накопичення різних завдань та видів робіт, ніж до звичного підсумкового контролю [2].

Оскільки для більшості українських університетів практика електронного вибору дисциплін є достатньо новою порівняно з іншими європейськими університетами, можна сказати, що вона знаходиться на початковому етапі розвитку. Як у концептуальному, так і технічному плані. Створення необхідних умов для оптимізації переліку вибіркових навчальних дисциплін сприятиме забезпеченню відповідності фахової підготовки в університетах вимогам національного ринку праці, змінам, що відбуваються в державі та світі [3].

Актуальність теми полягає в тому, що запровадження більш сучасного та інтерактивного підходу щодо існуючих онлайн систем вибіркових дисциплін значно полегшить цей процес для всіх його учасників. Студентам буде легше робити кінцевий вибір та організовувати свій навчальний процес, а викладачі будуть зацікавлені в підвищенні навантаження по конкретним вибірковим дисциплінам за рахунок нових можливостей.

Метою індивідуального дослідницького проєкту є розширення функціоналу існуючих систем для електронного голосування українських ВНЗ за рахунок створення спеціалізованої інформаційної системи.

Об’єктом дослідження є вибіркові дисципліни.

Предметом дослідження є удосконалення функціоналу існуючих систем для вибіркових дисциплін вищих навчальних закладів України.

Завдання роботи:

- проаналізувати існуючі системи для вибіркових дисциплін українських вишів;
- запропонувати власні рішення виявлених проблем;
- розробити додаток, який буде включати додатковий функціонал для вирішення виявлених проблем.

# 1 ЗАГАЛЬНІ ВІДОМОСТІ

Комп`ютерні технології та інформаційно-комунікаційні процеси давно стали невід'ємною частиною життєдіяльності людини в сучасному світі. Ми живемо в цифровому суспільстві, за його законами та правилами, намагаючись, формувати процес розвитку інформаційного суспільства тут і зараз.

Інформатизація освіти в Україні — один з найважливіших механізмів, що зачіпає основні напрямки модернізації освітньої системи. Сучасні інформаційні технології відкривають нові перспективи для підвищення ефективності освітнього процесу. Змінюється сама парадигма освіти. Велика роль надається методам активного пізнання, самоосвіті, дистанційним освітнім програмам.

Інформатизація освіти дає можливість отримати наступні переваги:

– створення гнучкої системи навчання з використанням сучасних інструментів і педагогічних методик, яка забезпечуватиме можливість індивідуальних траекторій навчання (програм, термінів, місця навчання та сертифікації результатів навчання);

– створення вільного та відкритого доступу навчальних ресурсів для учнів і студентів незалежно від місця та часу;

– відкритість і доступність навчання й обміну кращими практичними здобутками педагогів;

– можливість створення оптимальних систем моніторингу якості освіти, проведення експертної оцінки програм і схем навчання тощо;

– публічність процесу навчання, його відкритість і прозорість на всіх рівнях, що створює умови для формування ринку освітніх послуг країни, який базується на реальних потребах і запитах суспільства [4].

З появою Інтернету людство отримало можливість широко використовувати інформаційно-комунікаційні технології для організації навчальної, самостійної, колективної роботи учнів та студентів.

Сьогодні кожен вищий навчальний заклад України прагне мати свій власний веб-сайт, оскільки його наявність є невід'ємною умовою успішної діяльності. Якщо раніше сайти були простим сховищем HTML-файлів, то з розвитком веб-технологій з'явилась можливість реалізувати повноцінні освітні платформи з особистим кабінетом, потужною базою навчальних матеріалів та персоналізованим розкладом.

До основних функцій сайтів навчального закладу можуть бути віднесені: інформативна (забезпечує надання відкритого доступу до інформації і створення умов для інформаційного обміну), інтеграційна (забезпечує реалізацію внутрішньосистемних зв'язків), комунікаційна (дозволяє підтримувати зв'язки "всередині" себе, а також із "зовнішнім" інформаційним простором), координуюча (сайт має властивість фіксувати і представляти у взаємозв'язку контент, адресований різним суб'єктам), розвиваюча (пов'язана з розвитком інтелекту, особистих творчих якостей), культуроформуюча (в контексті формування культури і, зокрема, інформаційної культури), професійно-орієнтуюча (витікає з орієнтації на профіль професійної діяльності користувача сайту) [5].

Дієвий веб-сайт закладу освіти відкриває нові можливості в організації навчального процесу, оскільки сучасні інформаційно-комунікаційні технології дозволяють застосовувати дистанційне (електронне) навчання.

За допомогою веб-сайту відбувається взаємодія студента і викладача під час навчального процесу. Веб-сайт закладу освіти надає можливість кожному учаснику навчального процесу ідентифікувати себе, створивши "електронний кабінет", що забезпечує оперативне одержання дотичної інформації. Веб-сайт закладу освіти може мати дуже розгалужену структуру з певним набором розділів і підрозділів, а також вміщувати в себе набір інформаційних систем. Сучасні комп`ютерні технології та стрімкий розвиток мов програмування на сьогодні дозволяють "наповнювати" веб-сайт різноманітною динамічною інформацією, яка призначена як для ознайомлення, так і для дистанційного спілкування учасників

навчального процесу. В умовах сьогодення така можливість набуває важливого значення. В той же час, дистанційне навчання призвело до збільшення навантаження на викладацтво, зокрема, додатковий час на створення чи перенесення своїх курсів до систем дистанційної освіти, їхнє регулярне оновлення, доповнення та спілкування зі студентами.

Тому організація якісного навчального процесу “на відстані” з використанням новітніх інформаційно-комунікаційних засобів та відкритим доступом до освітніх ресурсів може надати широкий спектр освітніх послуг як для абітурієнтів і студентів для набуття необхідних навичок та вмінь для майбутньої професійної діяльності, так і для викладачів з метою підвищення кваліфікації. Саме така форма навчання може швидко адаптуватися до вимог інформаційного суспільства та підготувати майбутнього фахівця [6].

Вдосконалення і оптимізація інформаційних процесів у складі веб-сайтів закладів освіти дозволить всім учасникам навчального процесу мінімізувати втрати часу на налагодження швидкої комунікації. Доступність і зручність використання внутрішніх веб-ресурсів закладів освіти для комунікації студентства та викладацького складу сприятиме оптимальній організації навчального процесу.

Розглянемо інформаційну систему для вибіркових дисциплін як складову структури веб-сайту “Електронного Кампусу” КПІ. Для реалізації права здобувачів вищої освіти на вибірковість дисциплін вищі навчальні заклади мають забезпечити сприятливі умови у розкладах занять. Важливого значення у процесі вибору таких дисциплін набувають інформаційні кампанії для студентів, які проводять викладачі ВНЗ, присвячені орієнтованому на студента підходу до освітньої програми [7]. Інформаційна система для вибіркових дисциплін надає таку можливість — об’єднуватися задля організації спільних презентацій вибіркових дисциплін для студентів. Окрім того, вона надає можливості з гнучкої координації навчання як для студентів, так і викладачів.

# 2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Під час пошуку можливих рішень було вирішено звернутись до існуючих інформаційних систем ВНЗ України, основним завданням яких є реалізація права студентів на вільний вибір навчальних дисциплін в онлайн форматі.

Було проведено роботу по аналізу відповідної інформації, доступної на офіційних джерелах університетів в мережі Інтернет. По завершенню даного аналізу було складено список вишів, що відповідають необхідним для дослідження критеріям. Надалі цей список було відфільтровано таким чином, щоб продемонструвати в дослідницькій роботі різноманітні підходи до одного й того ж рішення. Як наслідок, у фінальному списку залишилось чотири виші, які мають високий рейтинг серед абітурієнтів, а також регулярно з’являються у тематичних статтях, підбірках, публікаціях, тощо.

## 2.1 Київський політехнічний інститут імені Ігоря Сікорського

У Київському політехнічному інституті імені Ігоря Сікорського процес вибору дисциплін здійснюється на базі інформаційно-телекомунікаційної системи “Електронний Кампус”. Це власна розробка університету, що виконує функцію інформаційно-телекомунікаційного середовища та використовується для інформаційної підтримки повсякденної діяльності студентів, викладачів, співробітників університету, а так само для інформаційної підтримки всіх видів інноваційної діяльності в університеті [8]. Вона є багатофункціональною та включає в себе електронний журнал студента, базу з методичним забезпеченням, дошку оголошень, тощо. В розділ вибіркових дисциплін можна потрапити натиснувши на кнопку у навігаційному меню, в особистому кабінеті студента. Відкриється сторінка, на якій відображені обрані дисципліни та шаблони вибіркових дисциплін. На рисунку 2.1 зображена інформаційна система для вибору дисциплін у КПІ.

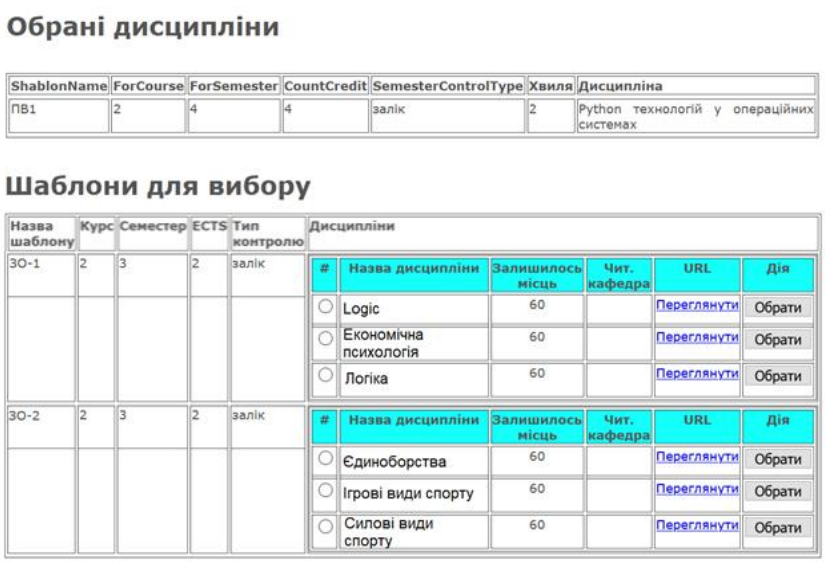



Рисунок 2.1 — Інформаційна система для вибору дисциплін у КПІ

Проведемо короткий огляд інтерфейсу інформаційної системи для вибору дисциплін у КПІ, він складається з наступних частин:

- блок обраних дисциплін — призначений для відображення дисциплін, остаточно обраних студентом;
- блок шаблонів — містить шаблони з освітньою інформацію про вибіркові дисципліни, наприклад курс, семестр, кількість кредитів та тип контролю;
- блок вибіркових дисциплін — реалізує функціонал вибору дисциплін студентами.

Проведемо короткий огляд функціоналу інформаційної системи для вибору дисциплін у КПІ, він складається з наступних можливих дій:

- вибір предмету — здійснюється натисканням радіокнопки навпроти бажаної дисципліни на початку таблиці;
- збереження результатів вибору — здійснюється натисканням кнопки напроти бажаної дисципліни в кінці таблиці;
- перегляд результатів вибору — обрані дисципліни відображаються в блоці вибіркових дисциплін.

## 2.2 Львівський національний університет імені Франка

У Львівському національному університеті імені Франка процес вибору дисциплін здійснюється на базі автоматизованої системи управління вищим навчальним закладом від ПП “Політек-СОФТ”. Цей та інші продукти компанії є універсальними рішенням для організації та підтримки навчального процесу в вищих навчальних закладах України I-IV рівнів акредитації [9]. Вони надають комплексний функціонал для автоматизації планування та обліку навчального процесу, автоматизації діяльності приймальної комісії, автоматизації тестування студентів, тощо. В розділ вибіркових дисциплін можна потрапити натиснувши відповідну кнопку у навігаційному меню, в особистому кабінеті студента. На рисунку 2.2. зображений інтерфейс інформаційної системи для голосування за вибіркові дисциплін у ЛНУ.

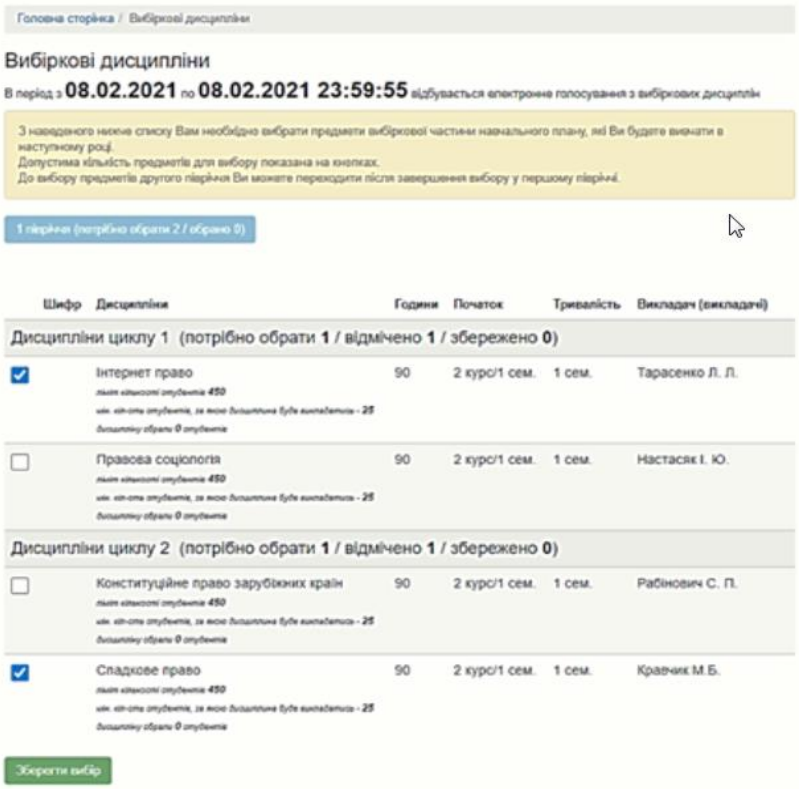



Рисунок 2.2 — Інформаційна системуа для вибору дисциплін у ЛНУ

Проведемо короткий огляд інтерфейсу інформаційної системи для вибору дисциплін у ЛНУ, він складається з наступних частин:

- інформаційний блок — містить короткі інструкції щодо голосування та дані про період його проведення, а також кількість обраних дисциплін;
- блок вибіркових дисциплін — реалізує функціонал вибору дисциплін, а також містить мінімальну інформацію про предмет.

Проведемо короткий огляд функціоналу інформаційної системи для вибору дисциплін у ЛНУ, він складається з наступних можливих дій:

- вибір предмету — здійснюється натисканням чекбоксу навпроти бажаної дисципліни на початку таблиці;
- збереження результатів вибору — здійснюється натисканням кнопки “Зберегти” внизу сторінки.

## 2.3 Національний авіаційний університет

У Національному авіаційному університеті процес вибору дисциплін здійснюється за допомогою вузькоспеціалізованої автоматизованої системи “Формування індивідуальної освітньої траєкторії”, яка є власною розробкою навчального закладу. ЇЇ єдине призначення — забезпечення студентів можливістю вибору дисциплін у зручному, електронному форматі. Тому якщо студент бажає отримати доступ, наприклад, до методичних матеріалів чи електронного журналу — йому слід скористатись окремими інформаційними системами, запропонованими Національним авіаційним університетом. Для входу в систему для вибіркових дисциплін необхідно ввести корпоративну пошту та пароль. Облікові дані видаються кафедрою та обов’язково належать до домену “gmail.com” [10]. На рисунку 2.3 зображено типовий інтерфейс інформаційної системи НАУ, з переліком вибіркових дисциплін.


Рисунок 2.3 — Інформаційна система для вибору дисциплін у НАУ

Проведемо короткий огляд інтерфейсу інформаційної системи для вибору дисциплін у НАУ, він складається з наступних частин:

– блок рекомендованих вибіркових дисциплін — автоматично запропонований студенту перелік дисциплін. Він також буде затверджений як обраний. якщо студент не братиме участь у голосуванні. виділений блакитним кольором;
– блок альтернативних вибіркових дисциплін — перелік дисциплін, доступних студенту для вибору на альтернативу рекомендованим. виділений помаранчевим кольором.

Проведемо короткий огляд функціоналу інформаційної системи для вибору дисциплін у НАУ, він складається з наступних можливих дій:

– заміна рекомендованої дисципліни — здійснюється натисканням чек-боксу в блакитному та помаранчевому блоках, по одному предмету з кожної сторони. Після цього необхідно натиснути кнопку заміни і дисципліни помінюються місцями;

– збереження результатів вибору — здійснюється натисканням кнопки “Зберегти” внизу сторінки.

## 2.4 Запорізький національний університет

У Запорізькому національному університеті процес вибору дисциплін здійснюється на основі навчальної платформи “Moodle”. Вона призначена для об'єднання педагогів, адміністраторів і студентів в одну надійну, безпечну та інтегровану систему для створення персоналізованого навчального середовища [11]. Важливо зазначити, що це рішення є не лише безкоштовним та відкритим (англ. open source), але й надзвичайно популярним та поширеним як в Україні, так і у всьому світі. Завдяки функціоналу, який пропонує “Moodle” можна створювати як прості інформаційні сайти, так і складні та розгалужені освітні системи.

У розділ вибіркових дисциплін на сайті ЗНУ можна потрапити прямо з особистого кабінету студента. Система автоматично перенаправить користувача на сторінку, де студент отримає можливість проголосувати за певні предмети [12]. Інтерфейс інформаційної системи для вибору дисциплін ЗНУ наведено на рисунку 2.4.

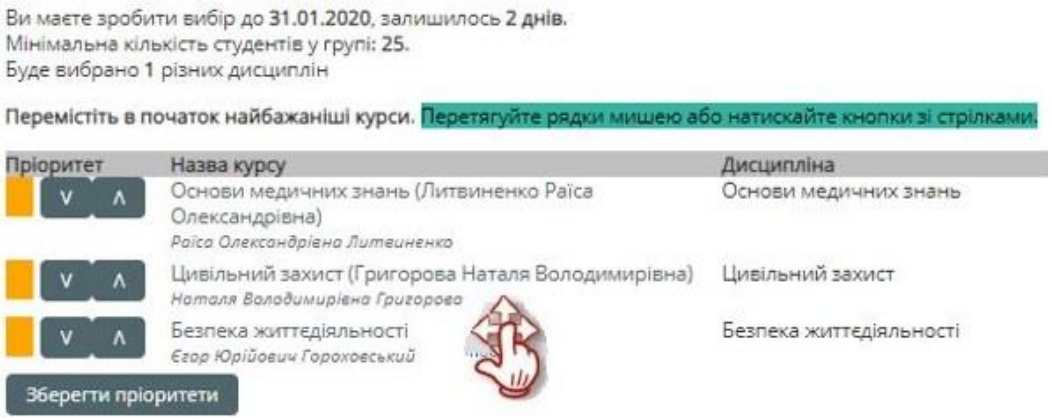



Рисунок 2.4 — Інформаційна система для вибору дисциплін у ЗНУ

Проведемо короткий огляд інтерфейсу інформаційної системи для вибору дисциплін у ЗНУ, він складається з наступних частин:

- інформаційний блок — містить дані про період проведення голосування, мінімальну кількість студентів у групі та кількість дисциплін, що буде обрано. Також наявні вказівки щодо самого процесу вибору;
- блок вибіркових дисциплін — реалізує функціонал вибору дисциплін, а також містить мінімальну інформацію про предмет.

Проведемо короткий огляд функціоналу, він складається з наступних можливих дій:

- зміна пріоритету вибіркової дисципліни — здійснюється шляхом перетягування предмету на початок переліку мишею, або натисканням кнопок на початку рядка;
- збереження результатів вибору — здійснюється натисканням кнопки “Зберегти пріоритети” внизу сторінки.

## 2.5 Висновки до розділу

В процесі огляду існуючих рішень було розглянуто інформаційні системи чотирьох вищих навчальних закладів України, за допомогою яких, або на базі яких забезпечується процес голосування за вибіркові дисципліни їх студентами.

За підходом до реалізації розподіл наступний:

- НАУ та КПІ — система реалізована власними ресурсами університету;
- ЛНУ — система реалізована приватною фірмою, на замовлення;
- ЗНУ — в основі системи лежить відкрите програмне забезпечення.

Варто відмітити, що при значних відмінностях у реалізації, усі розглянуті системи пропонують студентам схожий функціонал. Тому досить актуальним є

завдання по його розширенню, створенню більш інтерактивного підходу, відповідно до сучасних стандартів навчання.

Одним із важливих факторів, що впливає на кінцевий вибір студента є ступінь його ознайомленості з тією чи іншою дисципліною. Ймовірність того, що кінцевий вибір припаде на предмет про який студент майже нічого не знає — дуже низька. Тому можливість участі у ввідній лекції, яка є у розроблюваній системі, є хорошим рішенням зазначеної проблеми.

Іншим вагомим інструментом, який може запропонувати розроблювана система — є функціонал по організації студентів. Він допоможе викладачам швидко та ефективно координувати студентів на початку навчання, додаючи посилання на групи з лекційних та практичних занять, у яких вже буде відбуватись подальше спілкування. Студенти при цьому, також отримають певні переваги — вони будуть чітко бачити, які вибіркові дисципліни вони вивчатимуть в поточному семестрі, замість того, щоб шукати їх поміж інших, у загальному розкладі.

Таким чином, зазначений функціонал значно покращить користувацький досвід як для студентів, так і викладачів — зробить його прозорішим та гнучкішим.

Після ретельного аналізу існуючих рішень, а також визначення їх основних недоліків та проблем, можна приступити до проєктування власної інформаційної системи для вибіркових дисциплін, яка буде пропонувати студентам і викладачам покращений та розширений функціонал, описаний вище.

Перед розробкою, було прийнято рішення обрати структуру, в основі якої лежить взаємодія клієнтської та серверної частин. Обидві складові системи будуть базуватися у глобальній мережі Інтернет та взаємодіяти безпосередньо через неї. Це дозволить їй бути легкодоступною на багатьох типах пристроїв, без попереднього завантаження. В той же час це відкриває можливість до швидкої інтеграції у вже існуючі рішення, що також є одним з пріоритетів при розробці.

# 3 РОЗРОБЛЕННЯ ДІАГРАМИ ПРЕЦЕДЕНТІВ

## 3.1 Вимоги до формування інформаційної системи

Інформаційною системою прийнято вважати комплекс обладнання, програмного забезпечення та телекомунікаційних засобів, що забезпечують використання ІТ-ресурсів з метою надання різноманітних інформаційних послуг. У випадку розроблюваної системи — це здійснення професійної діяльності та вирішення поточних завдань, які стоять перед студентами, викладачами, науковцями та співробітниками, а також перед відповідними кафедрами університету.

Важливою вимогою до інформаційної системи є також наявність ІТ-стратегії. Вона сприяє швидкому розвитку і зміцненню конкурентних позицій за рахунок сучасних інформаційних систем, які вирішують широкий комплекс завдань на всіх рівнях управління вищими навчальними закладами.

Обов’язковою складовою частиною ІТ-стратегії є стратегія розвитку ІТ-інфраструктури. Вона забезпечує ефективне впровадження ІТ-процесів у створюваній системі. Інформаційні послуги надаються за встановленими вимогами. Якісна інформаційна система має відповідати великій кількості вимог, проте серед найважливіших — наступні вимоги:

- адекватна вартість;
- доступність;
- адаптивність;
- безпека.

Розглянемо детальніше, що означає кожен з перелічених аспектів.

Адекватна вартість. В умовах високих цін на програмне забезпечення, а також враховуючи те, що більша частина бюджету університету витрачається на підтримку інфраструктури, а не на її формування, вимога до недорогої ІТ-інфраструктури стає особливо важливою.

Доступність. Інформаційна система повинна надавати доступ до інформаційних ресурсів та сервісних служб студентам і працівникам вишів, у будь-який час, в незалежності від місця перебування. Варто зазначити, що термін “доступність” також стосується UI/UX дизайну, адже недосвідчений користувач повинен мати змогу швидко розібратися з системою при першому ж використанні.

Адаптивність. Представляє собою властивість інформаційної системи, що дозволяє не затрачуючи багато часу, вносити зміни з ціллю адаптації до змін зовнішнього і внутрішнього середовища університету. Також мається на увазі можливість збільшення, за необхідності, кількості компонент інформаційної системи. Швидкість зміни освітньої програми часто вимагає миттєвого реагування, викликаючи зміни бізнес-процесів і необхідність створення і впровадження відповідних технологій, які повинні підтримуватися сервісами IT-інфраструктури. Гарним прикладом необхідності миттєвого реагування у кризовій ситуації став карантин 2020 року, під час якого все навчання довелося перевести в онлайн режим. Завдяки освітнім інформаційним системам це вдалося зробити досить швидко, але при цьому якість навчання значно знизилась. Можна зробити висновок, що якнайшвидше реагування на нетипову ситуацію є надзвичайно важливим компонентом успішної інформаційної системи.

Безпека. Дана вимога включає в себе захищеність системи від зовнішніх впливів та уражень, що можуть заважати доступності, цілісності або конфіденційності інформації. Для забезпечення надійності системи нерідко вдаються до резервування комп’ютерів та їх компонентів, як і сегментів мереж.

Підсумовуючи, можна зазначити, що стратегічний підхід до формування інформаційної системи передбачає не тільки постійний аналіз зовнішнього середовища, але й ретельне відстеження внутрішніх IT-процесів. Не варто забувати про всі критерії якісної інформаційної системи при її розробці, особливо звертаючи увагу на те, що даним продуктом будуть користуватися заклади освіти. Тож

від таких факторів, як: безпека, адаптивність, вартість та доступність — буде напряму залежати зручність користування інформаційною системою.

Варто звернути увагу на те, що для ефективного управління освітньою програмою, основними ресурсами і завданнями бізнес-процесів вишів — необхідно використовувати новітні рішення в області IT-технологій, а саме:

- хмарні технології;
- енергоефективне IT-обладнання;
- системи цифрового друку;
- системи зв’язку;
- процеси автоматизації.

Інформатизація закладу повинна бути спрямованою на створення високоякісного та високотехнологічного соціально-орієнтованого електронного на ково-освітнього середовища університету через системний розвиток IT-інфраструктури.

У сучасних умовах саме заклади освіти намагаються поширити інноваційні та передові технології, а також методи надання інформації і організації процесів навчання у світі, де інформаційні технології стали невід’ємною частиною нашого життя, а людина — невіддільною від персоналізованих засобів зв’язку та мережі Інтернет. Замість неперевірених та не завжди якісних матеріалів з мережі, студенту пропонується зручна електронна база освітніх матеріалів, з контекстним пошуком і доступом із будь-якої точки світу.

При експлуатації такої складної інформаційної системи необхідно щодня вирішувати безліч завдань по експлуатації, модернізації, підтримці та забезпеченню безпеки роботи систем. Складність також може полягати в тому, що окремі користувачі погано орієнтуються у інтерфейсах комп’ютера. Тому варто знову наголосити, що UI/UX дизайн системи має бути максимально простим та зрозумілим. Це також дозволить охопити більшу кількість користувачів.

## 3.2 Розроблення власної системи

Розпочати розробку власної системи для вибіркових дисциплін варто з діаграми прецедентів, оскільки вона дозволить чітко окреслити необхідний функціонал, абстрагуючись від конкретної реалізації. Діаграма прецедентів наведена у кресленнику 1.

Варто нагадати, що її основне завдання полягає у візуальному зображенні різноманітних сценаріїв взаємодії між акторами (користувачами) і прецедентами (випадками використання). Таким чином вона описує функціональні аспекти системи (бізнес логіку). Добре продумані і завершені специфікації прецедентів використовуються протягом усього циклу розробки: від проєктування до тестування [13].

У власній реалізації системи для вибіркових дисциплін, користувачі поділені на три можливі типи:

- студент;
- викладач;
- адміністратор.

В залежності від ролі, користувачу надається різний набір можливостей для взаємодії з системою. Для прикладу, якщо адміністратор та викладач мають доступ до редагування інформації, то студент може лише переглядати її.

Також в межах системи існує поняття етапу — воно використовуються для надання користувачам доступу до системи у конкретні проміжки часу. Усього є чотири етапи. Вони мають циклічний характер, тобто по завершенню четвертого етапу розпочинається перший. Розглянемо кожен з них з точки зору різних типів користувачів. Діаграма послідовності наведена у кресленнику 2.

Перший етап:

- студент не має доступу до додатку;
- викладач не має доступу до додатку;

– адміністратор завантажує списки студентів та дисциплін, що доступні їм для вибору. Після завантаження відповідна інформація автоматично додається у базу даних.

Другий етап:

– студент ознайомлюється зі списком вибіркових дисциплін, а також віддає свій голос за предмети, з яких хотів би побачити ввідні лекції (ввідна лекція — це коротка лекція в онлайн форматі, на якій викладач знайомить студентів з предметом, описує програму, відповідає на запитання та намагається зацікавити студентів у виборі саме його дисципліни). Коли предмет набирає п’ять таких голосів — він відправляється на розгляд до адміністратора. Якщо все проходить успішно, то надалі студент бачить інформацію про ввідну лекцію та може її відвідати. Коли студент остаточно визначається з вибірковими дисциплінами - він голосує за них у інформаційній системі для вибіркових дисциплін свого вишу;
– викладач не має доступу до додатку;
– адміністратор зв’язується з потрібними кафедрами та передає списки предметів, з яких студенти хотіли б послухати ввідні лекції. Якщо на кафедрі погоджуються з проведенням ввідної лекції та знаходять вільного викладача — адміністратор додає попередньо узгоджену інформацію в систему, а саме: короткий опис ввідної лекції, викладача, що проводитиме лекцію, дату та час лекції, а також посилання на неї. По завершенню процесу голосування за вибіркові дисципліни в університеті, адміністратор переводить додаток на наступний етап.

Третій етап:

– студент очікує результатів голосування;
– викладач не має доступу до додатку;
– адміністратор завантажує списки викладачів та оновлені списки студентів, з обраними предметами, які вони вивчатимуть в наступному семестрі.

Четвертий етап:

- студент ознайомлюється зі списком вибіркових дисциплін, які він вивчатиме, а також додається у чати з практичних та лекційних занять, які додали викладачі;
- викладач додає координаційну інформацію про практичні та лекційні заняття, а саме: викладача, мессенджер у якому буде відбуватись його подальше спілкування зі студентами та посилання на чат в цьому месенджері;
- адміністратор очікує початку навчання у наступному семестрі та переводить систему до початкового стану.

## 3.3 Висновки до розділу

У розділі була описана діаграма прецедентів інформаційної системи для вибіркових дисциплін. В результаті виконаної роботи був окреслений та занотований функціонал розроблюваної системи, з абстрагуванням від конкретної реалізації — вона буде описана у наступних розділах. Відповідно до діаграми прецедентів, користувачі поділені на три ролі:

- студент;
- викладач;
- адміністратор.

Процес взаємодії з системою поділений на чотири етапи. Такий розподіл забезпечить користувачам ефективне, зручне та прозоре використання системи. Окрім того, це дозволить уникнути технічних нюансів при написанні коду та взаємодії з базою даних.

Важливо зазначити, що описана вище структура також дозволить швидко та практично безшовно інтегрувати розроблювану інформаційну систему у будь-яку з розглянутих у Розділі 1.

# 4 РОЗРОБЛЕННЯ UI/UX ДИЗАЙНУ

Тепер, коли основний функціонал системи чітко окреслений у Розділі 3, можна приступити до створення UI/UX дизайну. Він надасть можливість не лише візуалізувати розроблювану систему та визначити вимоги до графічного інтерфейсу, але й виявити додатковий функціонал, не врахований на попередньому етапі проєктування інформаційної системи для вибіркових дисциплін.

Ретельно продумані макети/прототипи заощаджують час, зводять до мінімуму виникнення нових проблем під час тривалого періоду розробки [14].

Почати варто з тлумачення самого поняття UI/UX дизайну. Отже воно містить у собі дві складові частини:

- UI дизайн — це все, що стосується графічної складової користувацького інтерфейсу: фото та відео, типографія, зовнішній вигляд елементів керування, анімації та переходи, комплексна стилістика, тощо;
- UX дизайн — це все, що стосується функціональної складової користувацького інтерфейсу: якість графіки та текстового наповнення, архітектура навігації, швидкість роботи з додатком та адаптивність інтерфейсу під різні типи пристроїв.

При розробці UI/UX дизайну інформаційної системи для вибіркових дисциплін пріоритетними цілями були:

- максимально можлива інтеграція додатку у вже існуючу систему голосування за вибіркові дисципліни, на прикладі “Електронного Кампусу” КПІ;
- візуальна та концептуальна подібність до інтерфейсу оновленого “Електронного Кампусу” КПІ.

Такий підхід дозволить забезпечити можливість безшовної та швидкої інтеграції як для користувача (завдяки візуальній та функціональній подібності), так і для інженерів (завдяки відсутності необхідності побудови окремої інфраструктури).

## 4.1 Авторизація

Авторизація є важливим інструментом регулювання доступу до конфіденційної інформації у сучасних системах. Необхідність такого захисту може виникнути з різних причин, проте серед основних:

- сучасні вимоги щодо захисту приватної інформації користувачів;
- необхідінсть надання доступу до різного функціоналу, в залежності від ролі, яку займає конкретний користувач у системі.

Почнемо зі сторінки авторизації, вона є спільною для всіх трьох типів к ристувачів, що наявні у власній реалізації інформаційної системи для вибіркових дисциплін та представлена на рисунку 4.1

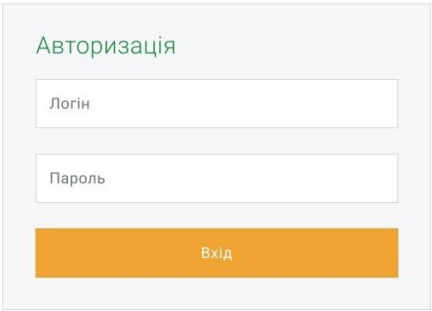



Рисунок 4.1 — Сторінка для авторизації користувачів

Вона є досить стандартною по сучасним міркам, містить в собі базову форму для введення логіну та паролю, а також кнопку для відправки даних на сервер. Можливості авторизуватись через соціальні мережі немає, адже облікові дані створюються централізовано та формуються у офіційні університетські списки. Можливості реєстрації немає по аналогічній причині — вона буде відбуватись автоматично, при завантаженні адміністратором попередньо наданих списків студентів та викладачів.

4.2 Кабінет студента

Далі розглянемо користувацький інтерфейс особистого кабінету студента. Як зазначалось раніше, студенти отримують доступ до системи на другому етапі. Також необхідно зауважити, що логін та пароль для входу в особистий кабінет повинен надати куратор групи або кафедра. Одразу після авторизації, студент потрапляє в особистий кабінет, зображений на рисунку 4.2.

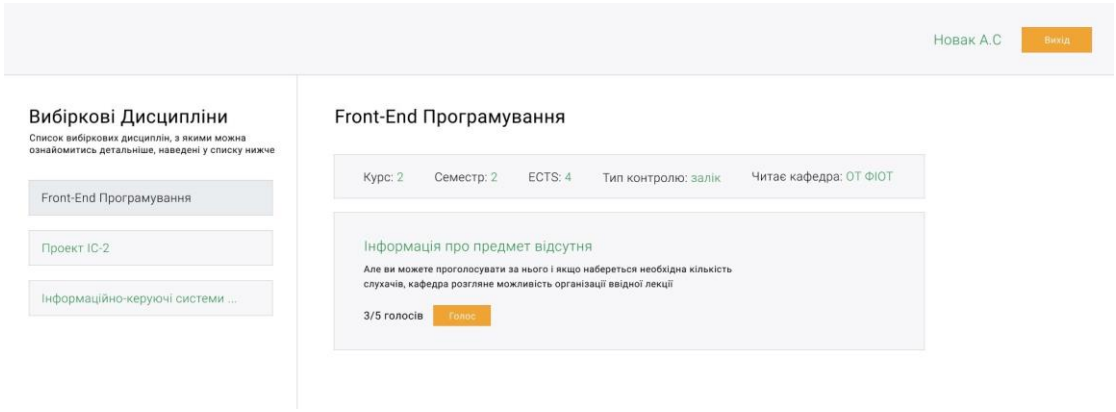



Рисунок 4.2 — Особистий кабінет студента

У верхній частині вікна можна побачити верхню панель (англ. header) сайту, з ідентифікаційною інформацією — фамілією та прізвищем студента. Поряд також розташована навігаційна кнопка для виходу з системи. Завдяки бічній панелі (англ. sidebar) з вибірковими дисциплінами, у лівій частині вікна, студент може ознайомитись зі списком дисциплін доступних йому для вибору, а при натисканні на одну з них він побачить у правій частині вікна лаконічний опис з детальною інформацією про:

- назву дисципліни;
- курс, на якому вивчатиметься дисципліна;
- семестр вивчення дисципліни;
- тип кінцевого контролю;

– кафедру, що викладає дану дисципліну.

У студента є можливість проголосувати за проведення ввідної лекції з цікавої йому дисципліни. Варто нагадати, що поняття ввідної лекції було детально описано у Розділі 3. Після натискання на кнопку “Голос” лічильник повинен оновитись, а при досягненні цілі в п’ять голосів - заявка на проведення ввідної лекції автоматично відправиться на розгляд адміністратору, про що студента повідомить відповідна інформаційна панель, зображена у верхній частині рисунку 4.3.

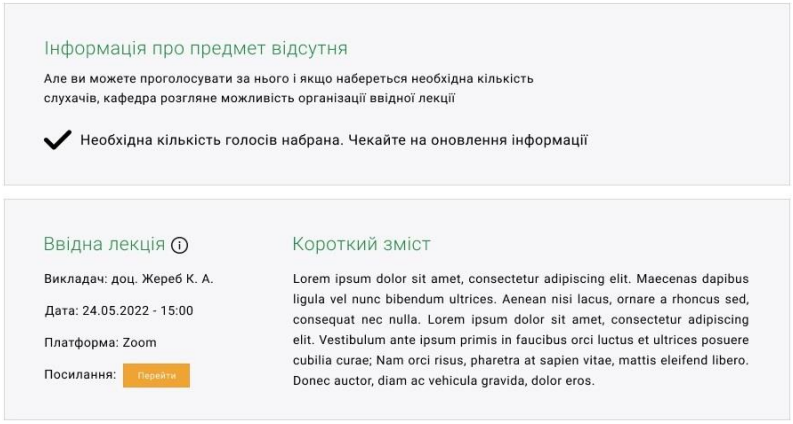



Рисунок 4.3 — Інформаційні панелі стосовно ввідних лекцій

Якщо адміністратору вдалося узгодити з відповідною кафедрою деталі ввідної лекції - він додає їх у систему, а студент одразу бачить їх на оновленій інформаційній панелі, зображеній у нижній частині рисунку 4.3. Тут можна ознайомитись з наступною інформацією:

– датою та часом проведення ввідної лекції;
– коротким змістом ввідної лекції;
– викладачем, що проводитиме ввідну лекцію;
– платформою, у якій буде проведена ввідна лекція.

Після натискання на кнопку “Перейти”, студент буде автоматично перенаправлений на відповідну онлайн-зустріч.

Якщо студент авторизується у системі на третьому етапі — він побачить коротке інформаційне повідомлення про те, що відбувається підрахунок голосів, тож варто зачекати. На рисунку 4.4 зображено інформаційне повідомлення про підрахунок голосів.

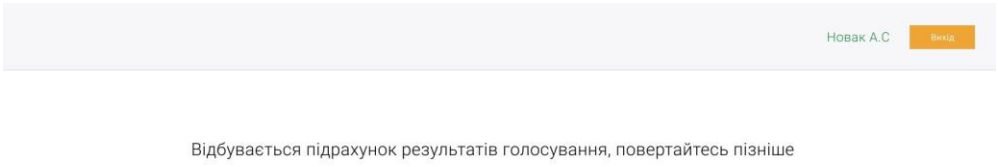



Рисунок 4.4 — Інформаційне повідомлення про підрахунок голосів

На четвертому етапі студент може ознайомитись з результатами голосування, тобто переліком вибіркових дисциплін, які він вивчатиме у наступному семестрі. Одразу після публікації результатів будуть відображатись лише назви предметів, але по мірі того, як викладачі додаватимуть координаційну інформацію — студент матиме можливість ознайомитись з нею та приступити до подальшого навчання. На рисунку 4.5. зображена інформаційна панель про відсутність координаційних даних.

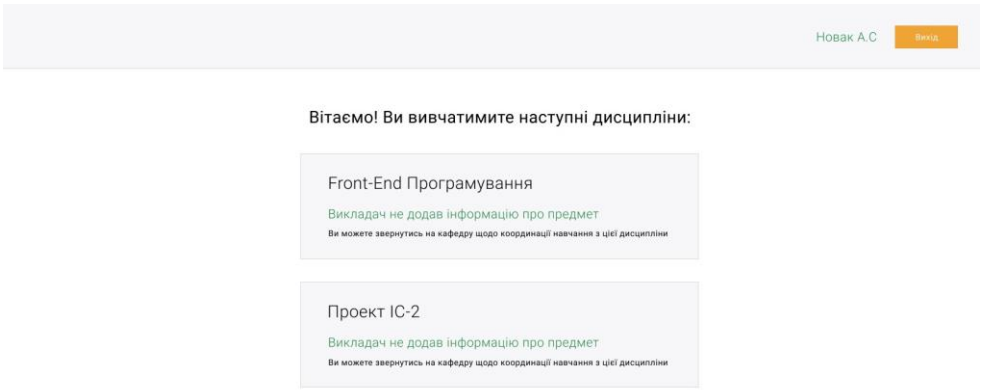



Рисунок 4.5 — Інформаційна панель про відсутність координаційних даних

Після того, як викладач додає координаційну інформацію, вона відображається в кабінеті студента, на інформаційній панелі з відповідної дисципліни та складається з:

- ім’я та прізвища викладача;
- назви месенджеру, у якому буде відбуватись подальше спілкування між викладачем та студентами;
- прямого посилання на чат.

Інформація додається як для лекційних, так і для практичних занять. Натисканням на кнопку “Перейти”, студент може потрапити у потрібний чат. На рисунку 4.6 зображено інформаційні панелі з заповненими координаційними даними.

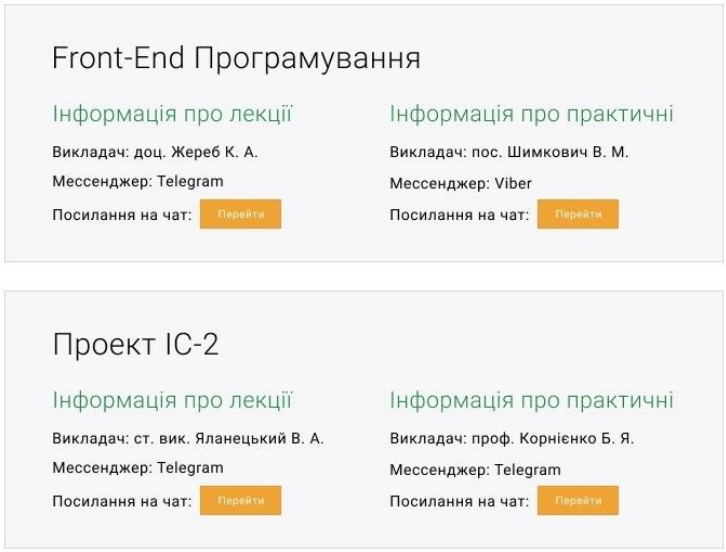



Рисунок 4.6 — Інформаційні панелі з заповненими координаційними даними

## 4.3 Кабінет викладача

Наступним кроком розглянемо інтерфейс особистого кабінету викладача в розроблюваній інформаційній системі для вибіркових дисциплін. Викладач

отримує доступ до системи на останньому, четвертому, етапі. Логін та пароль для входу надаються на відповідній кафедрі. Особистий кабінет викладача, одразу після авторизації зображений на рисунку 4.7.

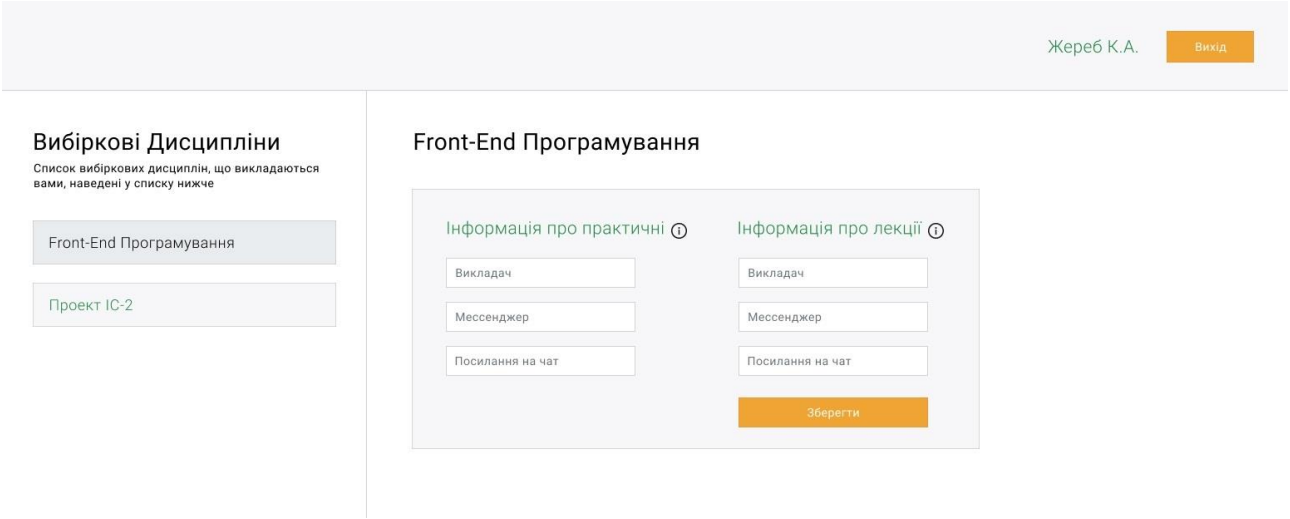



Рисунок 4.7 — Особистий кабінет викладача

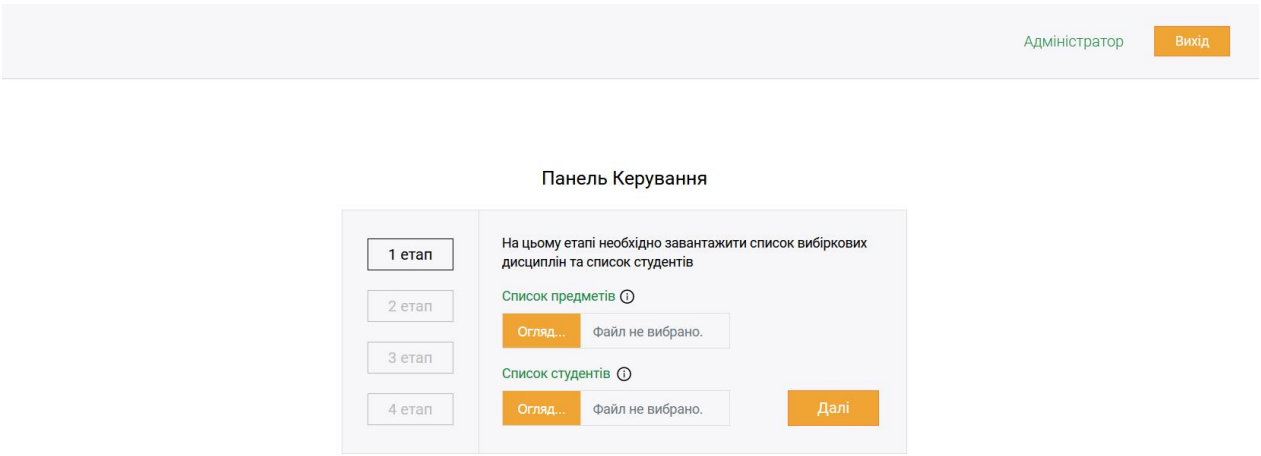
У верхній частині вікна розташована верхня панель (англ. header) з ідентифікаційною інформацією — прізвищем та ім’ям викладача. Поряд також розташована навігаційна кнопка для виходу з системи. На бічній панелі (англ. sidebar), у лівій частині вікна, відображені дисципліни за якими закріплений викладач. У правій частині вікна розташована форма, у якій можна заповнити дані про лекції та практичні:

– імена та прізвища викладачів;
– назву месенджерів, у яких буде відбуватись подальше спілкування між викладачем та студентами;
– прямі посилання на чати.

Для збереження інформації необхідно натиснути кнопку “Зберегти”.

## 4.4 Кабінет адміністратора

В останню чергу буде розглянутий найскладніший з точки зору інтерфейсу та функціоналу кабінет — це кабінет адміністратора. Одразу варто зазначити, що його зовнішній вигляд значно відрізняється від розглянутих раніше рішень через кардинально інші вимоги до функціоналу. Він зображений на рисунку 4.8.




Рисунку 4.8 — Особистий кабінет адміністратора

У верхній частині вікна розташована верхня панель (англ. header) з інформаційною позначкою про те, що це аккаунт адміністратора. Поряд також розташована навігаційна кнопка для виходу з системи. У центральній частині вікна розташована панель керування інформаційною системою. В свою чергу, у її лівій частині відображені чотири етапи. Поточний етап виділений чорним кольором, а наступні позначені сірим. Пройдені ж етапи, позначені зеленим кольором з відміткою. У правій частині панелі керування розташовані короткі інструкції, відповідні елементи взаємодії з системою, а також кнопка переведення системи на наступний етап.

Для завантаження списків студентів та предметів на першому етапі, адміністратор може скористатись відповідними полями для введення, обравши файли у форматі CSV на своєму пристрої.

CSV файли — це текстове представлення таблиць, а отже вони можуть бути створені у будь-якому з популярних редакторів (Excel, Google Sheets, тощо). Вони мають надзвичайно просту структуру — це список даних, розділених комами. Завдяки цьому вони є зручними для імпортування та експортування великих об’ємів інформації [15].

Структура таблиці студентів (як і викладачів) наступна:

- id (ідентифікатор), для прикладу “f1b567f390q6”;
- name (прізвище та ім’я), для прикладу “Нестеренко В.А.”;
- role (роль), одне з трьох наступних значень англійською мовою: “student” / “lecturer” / “admin” (студент / викладач / адміністратор);
- subjects (пов’язані предмети), для прикладу “001;002;003;”.

Індекс предмету береться зі списку предметів та пишеться однією строкою, через крапку з комою. В кінці також ставиться крапка з комою.

Структура таблиці предметів наступна:

- id (ідентифікатор), для прикладу “001”;
- name (назва), для прикладу “Бази даних”;
- votes (кількість голосів), для прикладу “3”;
- controlType (тип контролю), для прикладу “Екзамен”;
- department (кафедра), для прикладу “ОТ ФІОТ”;
- ects (кількість кредитів), для прикладу “4”;
- semester (семестр), для прикладу “6”.

На другому етапі адміністратор поступово отримує запити на проведення ввідних лекцій від студентів. Такі предмети автоматично відображаються у списку, зображеному на рисунку 4.10.


Рисунок 4.9 — Форма для введення даних щодо ввідної лекції

При натисканні на один з предметів, з'являється форма для введення даних щодо ввідної лекції, попередньо узгоджених з відповідною кафедрою, яка зображена на рисунку 4.9. Для збереження інформації необхідно натиснути кнопку "Зберегти".

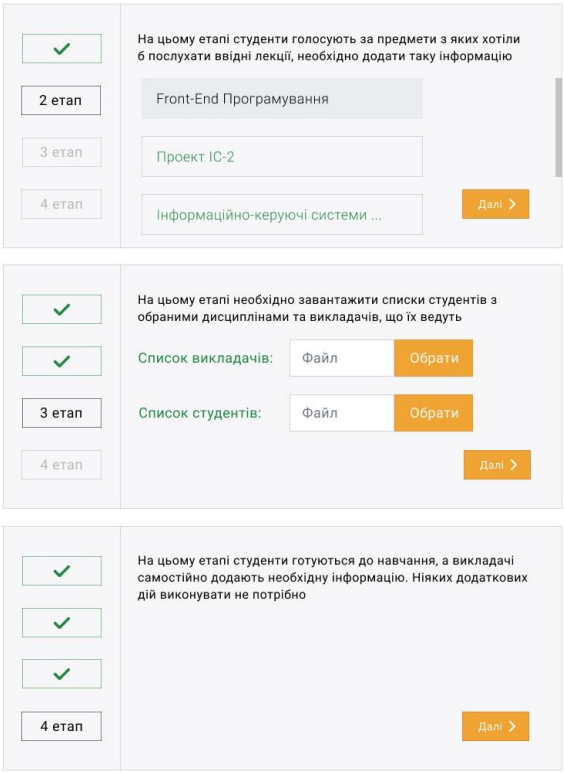



Рисунок 4.10 — Панель керування

На третьому етапі адміністратор завантажує списки викладачів, та оновлені списки студентів, вже з дисциплінами які вони вивчатимуть, які зображені на рисунку 4.9. Дії аналогічні до першого етапу.

На четвертому (останньому) етапі, адміністратору не потрібно виконувати ніяких дій, про що повідомляє текст у верхній частині інформаційної панелі — рисунок 4.9. При натисканні на кнопку “Далі”, система автоматично повернеться на самий початок.

## 4.5 Висновки до розділу

У даному розділі був детально розглянутий та описаний UI/UX дизайн інформаційної системи для вибіркових дисциплін. Робота велась з урахуванням потреб різних користувачів: студентів, викладачів та адміністраторів.

Вдалось сегрегувати роботу сервіса для різних типів взаємодії. В залежності від типу користувача, інтерфейс та контроллери відрізняються, щоб максималізувати користь та зручність роботи.

Сервіс виконаний у єдиній графічній стилістиці з “Електронним Кампусом” КПІ, оскільки саме ця система була обрана для демонстрації розширеного функціоналу розроблюваної інформаційної системи. Це повинно створити ефект “безшовності” при переході між ресурсами. У підсумку можна сказати, що завдання було виконане успішно. Як і планувалось, інтерфейс вийшов дуже схожим на “Електронний Кампус” КПІ — це палітра з білого, сірого, зеленого та помаранчевого кольорів, а також строгі прямокутні форми. Також вдалось імплементувати контроллери, що відповідають “Електронного Кампусу” КПІ.

Функціонал аналогічним чином повторює “Електронний Кампус КПІ”, з мінімалістичним підходом до інтерактивних елементів.

# 5 ВИБІР ОКРЕМИХ ТЕХНОЛОГІЙ

## 5.1 React

React — це JavaScript бібліотека для побудови користувацьких інтерфейсів. Розроблена компанією Facebook та випущена у 2013 році, вона є однією з найвпливовіших UI бібліотек сучасності. Найголовніша особливість бібліотеки — це запропонований нею процес мислення при створенні додатків [16].

React використовується для побудови компонентів (англ. components) — функцій, що репрезентують логічні фрагменти інтерфейсу з можливістю повторного використання. Ці функції власне й повертають візуальні елементи у формі JSX розмітки — спеціального синтаксису, що є суміщщю JavaScript і HTML. Якщо необхідно передати ті чи інші дані всередину компоненти, їх просто передають у вигляді параметрів функції (англ. props). Пізніше до них можна звернутись прямо у JSX розмітці або у тілі функції. Якщо ці параметри змінюються, React реагує на зміну та перемальовує інтерфейс. Компонента може не тільки отримувати дані зовні, але й містити їх всередині себе, реагуючи на їх зміну аналогічним чином.

Але одна з основних причин, за якою ховається популярність цієї бібліотеки в її величезній екосистемі. Наприклад, сам по собі React не “турбується” про анімації, навігацію, або глобальне управління даними. Натомість існує величезна кількість допоміжних бібліотек, які пропонують цей додатковий функціонал і переважна більшість з них розроблена саме спільнотою React розробників. Зважаючи на всі переваги описані вище, а також на особистий досвід роботи з цією бібліотекою, React був обраний основним інструментом для реалізації Front-End частини інформаційної системи для вибіркових дисциплін.

## 5.2 Redux

Redux — це бібліотека, яка пропагує використання “єдиного джерела правди” для всіх даних в JavaScript додатку. Вона була розроблена Facebook та наразі є однією з найпопулярніших у екосистемі React. З її допомогою можна створювати програми, які працюють послідовно, запускаються у різних середовищах (клієнтське, серверне) та легко тестуються [17]. Вона також має деякі вбудовані інструменти для спрощення процесу розробки, наприклад зневаджувач, що працює в реальному часі.

Сучасні веб застосунки представляються як розгалужені дерева компонентів які постійно керують даними, що в купі називаються станом (англ. state). І коли цей стан децентралізований, це може викликати серйозні проблеми з розумінням та тестуванням. Redux, на противагу, допомагає розробникам централізовано керувати великими об’ємами даних у застосунках та надає можливість розширювати цю інфраструктуру в майбутньому.

Бібліотека покладається на єдиний незмінний (англ. immutable) JavaScript об’єкт, щоб зберігати усю інформацію. Щоб змінити його, потрібно відправити (англ. dispatch) дію (англ. action), яка містить в собі назву та корисне навантаження у вигляді нових даних. Але оскільки згаданий раніше стан імутабельний, ця дія повинна пройти через редуктор (англ. reducer) — спеціальну функцію яка обробить дію та поверне повністю новий об’єкт, яким і буде замінений поточний стан. На виході ми отримуємо односторонній потік даних, що є передбачуваним та легким у тестуванні.

Завдяки доведеній ефективності описаного патерну, а також високому рівню інтеграції з React, ця бібліотека була обрана для глобального керування даними у розроблюваній системі.

## 5.3 Tailwind

Tailwind — це бібліотека з колекцією невеликих допоміжних CSS класів для швидкої та послідовної побудови користувацького інтерфейсу. Вона працює шляхом сканування усіх файлів проєкту на наявність імен класів, генеруючи відповідні стилі, а потім записуючи їх у статичний CSS файл. Це робить її швидкою, гнучкою і надійною — фактично з нульовим часом виконання [18].

На відміну від таких популярних рішень як Bootstrap або Material UI, що пропонують високорівневі компоненти (кнопки, форми, випадаючі меню, тощо), Tailwind пропонує більш функціональний підхід — невеликі стилізовані класи, які можуть комбінуватись для побудови тих самих високорівневих компонентів. Такий підхід є більш гнучким та надає розробникам значно більше свободи, водночас економлячи певну кількість часу в порівнянні з класичним CSS.

Але Tailwind це не лише про стилі, бібліотека також пропонує комплексні рішення зі створення адаптивних інтерфейсів за допомогою гнучкої системи сіток, керування станом елементів інтерфейсу за допомогою псевдоселекторів, а також оптимізації продуктивності шляхом очистки невикористовуваних фрагментів коду.

Оскільки UI/UX дизайн системи розроблявся під специфічні візуальні вимоги — таке рішення буде найбільш оптимальним для використання при подальшій програмній реалізації.

## 5.4 Firebase

Firebase — це набір інструментів для побудови додатків та управління інфраструктурою, що є надбудовою над Google Cloud Platform. Сервіс надає комплекти для розробки програмного забезпечення фактично під кожну з існуючих платформ, а також пропонує можливості для його подальшого масштабування

практично без необхідності знання бекенд розробки [19]. Такий підхід називається бекенд-як-сервіс (англ. backend-as-a-service).

Firebase пропонує наступні рішення:

- хмарні повідомлення — інфраструктура для відправки та отримання повідомлень між сервером та кінцевими пристроями;
- хостинг — інфраструктура для розміщення сайтів в мережі Інтернет, по захищеному протоколу доступу;
- хмарне сховище — інфраструктура для зберігання даних на серверах Google;
- аутентифікація — інфраструктура для реєстрації та авторизації користувачів, керування доступом до інформації;
- хмарні функції — інфраструктура для виконання коду на серверах Google;
- моніторинг — інфраструктура для моніторингу різноманітних показників застосунку.

Для розробки власної інформаційної системи для вибіркових дисциплін, було вирішено застосувати окремі рішення від Firebase.

## 5.5 Cloud Firestore

Cloud Firestore — це хмарне сховище даних від Firebase, яке дозволяє синхронізувати інформацію між усіма користувачами як в режимі реального часу, так і мануально.

Це рішення включає в себе зручні та надійні клієнтські бібліотеки, підтримку оффлайн режиму в разі втрати зв’язку з хмарою, а також комплексний набір засобів безпеки та захисту, а завдяки використанню NoSQL підходу, розробник може структурувати дані у необхідній йому формі [20]. Cloud Firestore також пропонує безшовну інтеграцію з іншими продуктами Firebase та Google Cloud, включаючи хмарні функції.

Як і усі рішення від Firebase описані нижче, дане рішення було обране по кільком ключовим критеріям:

- обмеженість часу на реалізацію бекенд частини застосунку;
- комплексність запропонованих рішень;
- надійність інфраструктури, на якій базуються сервіси.

### 5.6 Firebase Cloud Functions

Firebase Cloud Functions — це сервіс для хмарних обчислень від Firebase, що надає у використання розробникам свою серверну інфраструктуру. JavaScript або TypeScript код зберігається у хмарі Google і працює в керованому середовищі [21]. Він може бути корисним для автоматизації певних рішень, таких як розсилання інформаційних емейлів користувачам, або просто для зменшення навантаження на кінцеві пристрої при наявності важких обчислень. При цьому немає необхідності керувати окремими серверами та масштабувати їх. Рішення також включає в себе готові бібліотеки для взаємодії з серверним API від Google.

### 5.7 Firebase Authentication

Firebase Authentication — це компактне та водночас функціональне рішення для реєстрації та авторизації від Firebase. Багатьом системам необхідно мати підтвердження індивідуальності користувачів для надання персоналізованого досвіду та захисту їх інформації, тому даний сервіс пропонує різноманітні можливості для цього: через емейл, по номеру телефону або через існуючі акаунти в соціальних мережах. Firebase Authentication легко інтегрується з іншими сервісами Firebase і використовує галузеві стандарти, такі як OAuth 2.0 і OpenID Connect, тому його можна легко інтегрувати з існуючим бекендом [22].

## 5.8 Firebase Hosting

Firebase Hosting — це статичний хостинг-провайдер від Firebase, орієнтований в першу чергу на веб-розробників. Завдяки розгалуженій мережі власних SDN серверів у різних часових поясах, на яких і кешуються дані розміщених сайтів, сервіс може гарантувати достатньо швидкий та безперебійний доступ. Окрім того, Firebase Hosting видає SSH сертифікат для кожного опублікованого сайту, в підтвердження надійності та захищеності передачі даних. Важлива перевага в порівнянні з конкурентами полягає також у інтегрованості сервісу в екосистему Firebase і керування розгортанням сайту прямо з консолі. За допомогою однієї команди можна швидко розгортати веб-програми та передавати контент у глобальну мережу CDN [23].

## 5.9 Висновки до розділу

У даному розділі були детально розглянуті технології, підібрані для комплексної реалізації інформаційної системи для вибіркових дисциплін.

Для фронтенд частини були обрані: React — побудова архітектури додатку; Redux — керування глобальним станом; Tailwind — стилізація компонентів;

Для бекенд частини були обрані окремі рішення від Firebase: Cloud Firestore — хмарне сховище даних; Cloud Functions — хмарні функції; Authentication — реєстрація та авторизація користувачів; Hosting — публікація веб застосунків у мережі Інтернет;

Такий набір технологій дозволить побудувати тестову версію системи прийнятно якісно та у дуже стислі терміни, з можливістю перевикористання коду у майбутньому, при її доопрацюванні. Також важливо, що вона буде мати повноцінну архітектуру — складатись з фронтенду та бекенду.

# 6 РОЗРОБЛЕННЯ СХЕМИ БАЗИ ДАНИХ

Розроблення схеми бази даних є важливою частиною проектування будь-якої системи, оскільки вона допомагає розробникам уявити, яким саме чином повинна бути побудована модель взаємодії з даними. Її наявність дає чітке уявлення про те, які таблиці та поля повинна містити база даних розроблюваної системи [24]. Схема бази даних інформаційної системи для вибіркових дисциплін представлена у кресленнику 3. Розглянемо кожну з таблиць та зв'язки між ними детальніше.

Таблиця користувачів містить наступні поля:

- ідентифікатор;
- ім'я;
- роль;
- предмети.

Це універсальна таблиця для студентів, викладачів та адміністратора. В ній зберігатиметься авторизаційна та ідентифікаційна інформація учасників системи, а також список предметів, з яким так чи інакше пов'язаний конкретний учасник інформаційної системи.

Таблиця предметів містить наступні поля:

- ідентифікатор;
- назва;
- кількість голосів;
- базова інформація;
- інформація про ввідну лекцію;
- навчальна інформація.

У ній зберігатимуться вичерпні дані про дисципліни, що спочатку обираються, а потім вивчаються / викладаються. Більше того, в залежності від етапу

інформація про той чи інший предмет може доповнюватись. Дана таблиця пов’язана з таблицею користувачів зв’язком “один до багатьох”.

Таблиця з базовою інформацією містить наступні поля:

– ідентифікатор;
– тип контролю;
– кафедра;
– кількість кредитів;
– семестр.

Це уточнювальна таблиця, призначена для зберігання формальної інформації про предмет, відповідно офіційної університетської документації. Пов’язана з таблицею предметів зв’язком “один до одного”.

Таблиця з інформацією про ввідну лекцію містить наступні поля:

– ідентифікатор;
– опис;
– викладач;
– дата та час;
– платформа;
– посилання на відеозустріч.

Це також уточнювальна таблиця, з даними про ввідну лекцію, які додаються в разі досягнення згоди між адміністратором та кафедрою. В іншому випадку вона може бути замінена на значення null. Пов'язана зв’язком “один до одного” з таблицею предметів.

Таблиця з інформацією про навчання містить наступні поля:

– інформація по лекціям;
– інформація по практичним;

Третя по рахунку уточнювальна таблиця для зберігання даних безпосередньо про лекційні та практичні заняття з предмету. У разі відсутності даних може

бути замінена на значення null. Пов'язана зв’язком “один до одного” з таблицею предметів.

Таблиця з детальною інформацією про навчання містить наступні поля:

- ідентифікатор;
- викладач;
- мессенджер;
- посилання на чат.

Це таблиця, основною ціллю якої є масштабування попередньої, оскільки і лекції і практичні містять ідентичну координаційну інформацію. Пов’язана зв’язком “багато до багатьох” з попередньою таблицею відповідно.

Таблиця з інформацією про систему містить наступні поля:

- ідентифікатор;
- етап.

Окрема таблиця, не пов’язана з усіма іншими, єдиною ціллю якої є зберігання інформації про поточний етап системи.

## 6.1 Висновки до розділу

У даному розділі було розглянуто схему бази даних інформаційної системи для вибіркових дисциплін. Вона містить шість пов’язаних між собою таблиць та одну окрему — з інформацією про етап додатку. Між таблицями переважає зв’язок “один до одного”, за виключенням зв’язку між таблицями користувачів та предметів, він “один до багатьох”.

Загальна структура вийшла в міру декомпозованою та досить прозорою, не дивлячись на обширний функціонал веб застосунку. Практична реалізацію бази даних виглядатиме ще компактнішою за рахунок використання NoSQL рішення від Firebase — Cloud Firestore.

# 7 РОЗРОБЛЕННЯ СТРУКТУРНОЇ СХЕМИ

Завершити проєктування власної системи для вибіркових дисциплін варто зі структурної схеми. Вона дозволить систематизувати усю зібрану інформацію та відобразити взаємодію між окремими технологіями, обраними у минулих розділах. Структурна схема наведена у кресленнику 4.

Варто зазначити, що вона надає уявлення про структуру програмного забезпечення на верхньому рівні, що переважно включає взаємодію різних компонентів системи між собою. Дана схема може бути корисною як під час розробки, так і при розширенні системи або модифікації вже існуючого функціоналу [25].

На ній зображено ключові архітектурні складові інформаційної системи для вибіркових дисциплін, а саме:

- фронтенд частину, яка відповідає за відображення даних та їх мінімальну обробку;
- бекенд частину, яка відповідає за ресурсовитратні дії з обробки даних та операції з базою даних;

Розглянемо кожну з частин більш детально.

## 7.1 Фронтенд частина застосунку

Фронтенд частина реалізована за принципом односторінкового застосунку (англ. single page application), який розміщується в глобальній мережі Інтернет за певним посиланням, аналогічно до звичайного сайту. Його основна перевага перед традиційним, старішим підходом в тому, що замість підвантаження окремих сторінок або групи сторінок, використовується єдина HTML сторінка та Javascript код, який перемальовує її за необхідності. Це дозволяє значно пришвидшити роботу сайту для кінцевих користувачів, а також знизити навантаження на сервер. Усе це здійснюється за посередництвом асинхронних AJAX запитів,

які дозволяють підвантажувати дані та інструкції в JSON форматі, без перезавантаження сайту.

Односторінковий застосунок складається з наступних частин:

- презентаційного шару, де відбувається безпосереднє відображення даних за допомогою HTML, CSS та Javascript;
- шару бізнес логіки, де відбувається зберігання та модифікація даних за допомогою Javascript;
- шару доступу до даних, де відбувається взаємодія з веб сервером та офлайн сховищем застосунку, також за допомогою Javascript.

У розроблюваній системі за керування односторінковим застосунком відповідає Javascript-бібліотека React, з її ж допомогою ми будуємо усю архітектуру системи, а також через неї підключаємо UI бібліотеку Tailwind та бібліотеку керування станом Redux.

## 7.2 Бекенд частина застосунку

Для реалізації бекенд частини було використано Firebase. Це бекенд-як-сервіс (англ. backend-as-a-service) рішення від Google. Чому було обрано саме це рішення поміж схожих, або альтернативних, було обґрунтовано у Розділі 5 — вибір окремих технологій. В тому ж розділі був детально описаний сам сервіс та інструменти, які він пропонує розробникам, а зараз розглянемо безпосередньо структуру, яку буде експлуатувати розроблюваний веб застосунок.

Структура бекенд частини застосунку складається з наступних частин:

- хостинг, завдяки якому додаток доступний для користувачів за певною веб адресою;
- хмарні функції, в яких відбуваються блокуючі або просто ресурсозатратні операції;
- хмарне сховище, який є типовою noSQL базою даних.

Усі ці сервіси підтримуються серверними станціями Google, що розташовані в різних часових поясах по всьому світу, для забезпечення швидкості обміну даними.

## 7.3 Висновки до розділу

У розділі була описана структурна схема інформаційної системи для вибіркових дисциплін. Вона складається з фронтенд частини та бекенд частини, що постійно взаємодіють між собою.

У фронтенд частині додатку: React (побудова архітектури додатку), Redux (керування глобальним станом), Tailwind (стилізація компонентів).

У бекенд частині додатку: Cloud Firestore (хмарне сховище даних), Cloud Functions (хмарні функції), Authentication (реєстрація та авторизація користувачів), Hosting (публікація веб застосунків у мережі Інтернет).

Описана структура повинна забезпечити стабільну та швидку роботу веб додатку, а також відкрити певні можливості для масштабування системи в майбутньому за необхідності.

На основі зібраної інформації можна відтворити типовий потік даних. Він виглядає наступним чином:

- користувач вводить певні дані у браузері;
- фронтенд отримує їх, попередньо обробляє та робить з ними запит на бекенд;
- бекенд отримує запит, аналізує дані та звертається до бази даних;
- база даних повертає дані, які бекенд знову обробляє та надсилає назад у якості відповіді;
- фронтенд отримує відповідь з даними та опрацьовує їх;
- користувач бачить оновлені дані у браузері.

# 8 ПРОГРАМНА РЕАЛІЗАЦІЯ

На даному етапі завершується проєктування інформаційної системи для вибіркових дисциплін та починається її програмна реалізація. Програмна реалізація полягає у впровадженні процесів та процедур, необхідних для переходу системи від стадії планування до стадії виробництва [26]. На перший погляд можна подумати, що вона завершується одразу після публікації першої робочої версії застосунку, але це не так. Насправді цей етап значно триваліший та багатогранніший і передбачає як мінімум наступні моменти:

– адаптація інфраструктури відповідно до кількості користувачів;
– виявлення та виправлення багів, що виникають в процесі експлуатації;
– майбутні розширення функціоналу системи.

## 8.1 Фронтенд частина застосунку

Зазвичай розробка користувацького інтерфейсу починається з базових компонентів. У даному випадку, зважаючи на UI/UX дизайн — це кнопки та поля для вводу тексту / вибору дати й часу / завантаження файлів. Оскільки принцип реалізації усіх перерахованих елементів приблизно однаковий, розглянемо компоненту кнопки для прикладу, які зображені на рисунку 8.1.

```
function Button({label, onClick, isLoading, isDisabled, isSubmit, styles}: ButtonProps) {

    return (
        <button type={isSubmit ? 'submit' : 'button'} onClick={onClick} disabled={isDisabled || isLoading}
                className={`justify-center items-center w-full h-full ease-in duration-200
                border border-[#F08833] text-white text-base font-normal bg-[#F0A433] flex
                focus:outline focus:outline-3 focus:outline-[#92C7FF] ${styles}
                ${isDisabled && 'text-[#BABABA] border-[#E9ECEF] bg-[#E9ECEF]'}`}>
            {isLoading ? <img className='w-5 h-5 pt-[1.2px]' src={Spinner}/> : label}
        </button>
    )
}
```

Рисунок 8.1 — Компонента кнопки

Варто нагадати, що компонента — це звичайна функція, яка повертає JSX розмітку. Як можна бачити на рисунку, вона має характерну назву Button. Далі, в дужках, передаються необхідні для роботи вхідні параметри:

– label (назва кнопки);
– onClick (колбек функція, що викликається по натисканню);
– isLoading (прапорець для контролю за станом завантаження даних);
– isDisabled (прапорець для блокування кнопки);
– isSubmit (прапорець для визначення типу кнопки);
– styles (додаткові стилі).

Вони можуть використовуватись як в тілі функції, так і серед JSX розмітки. В блоці повернення значення (англ. return) використаний стандартний HTML елемент <button>, до якого застосовані Tailwind класи через атрибут className. Тут також присутні різноманітні умовні перевірки з використанням props, описаних вище. Внизу файлу компонента експортується, завдяки чому може багатократно використовуватись у інших місцях коду.

Тепер, коли в нашому розпорядженні є базові компоненти, ми можемо створювати складніші компоненти, такі як панель-заголовок сайту, який зображений на рисунку 8.2.

```
function Header({user}: HeaderProps) {

    const {isErrorOccurred} = useAppSelector( selector: state => state.app)

    return (
        <div className='items-center justify-end w-screen h-20
        pr-9 bg-[#F7F7F9] border-b border-b-[#D1D4D7] flex gap-8'>
            {isErrorOccurred && <ErrorScreen/>}
            <ToastContainer position="bottom-right"/>
            <p className='text-base font-light text-[#208843]'>{user || '–'}</p>
            <div className='h-8 w-20'>
                <Button onClick={() => auth.signOut()} label='Вихід' styles='text-sm'/>
            </div>
        </div>
    )
}
```

Рисунок 8.2 — Компонента панелі-заголовку сайту

Процес їх створення аналогічний попередньому, проте кількість логіки для роботи з даними в тілі компоненти зростає пропорційно до масштабу. Так, всередині компоненти Header ми вже можемо бачити хук (англ. hook) для роботи з глобальним станом додатку. Це спеціальна функція, що використовується для прямого доступу до даних, без їх попередньої передачі із компоненти в компоненту. В даному випадку вона необхідна для відображення інформаційного повідомлення в разі помилки зчитування даних про користувача. Також ми бачимо, що тут перевикористовується компонента Button, створена раніше.

З таких масштабних компонентів як Header, в свою чергу, складаються цілі сторінки, які зображені на рисунок 8.3.

```
function AdminAccount() {

    const {stage, authorizedUserData} = useAppSelector( selector: state => state.app)

    return (
        <>
            <Header user={authorizedUserData?.name}/>
            <div className='flex flex-row w-screen h-[calc(100vh-5rem)]'>
                <div className='w-full flex flex-col items-center justify-center'>
                    <p className='text-xl pb-5'>Панель Керування</p>
                    <AdminPanel currentStage={stage}/>
                </div>
            </div>
        </>
    )
}
```

Рисунок 8.3 — Компонента сторінки адміністратора

Невід’ємною частиною роботи з даними на фронтенді є форми. Вони використовуються для зворотнього зв’язку з користувачем. Розглянемо сторінку з формою для авторизації — Рисунок 8.4. Використовуючи стандартний HTML елемент <form>, обгортаємо поля для введення даних, а в кінці додаємо кнопку відправки форми.

Оскільки React пропагує використання патерну Flux (повсюдний контроль даними), то ми використовуємо спеціалізовану бібліотеку, яка й візьме на себе цю роботу. Створюємо змінні зі значеннями по замовчуванню, а потім пов'язуємо їх з відповідними <input> елементами у блоці повернення. На подію відправки форми (англ. handleSubmit) дані автоматично прийдуть зібраними у єдиний об'єкт, а нам залишиться лише провести бажані маніпуляції з ним.

```
function LoginPage() {

    const {register, handleSubmit, formState: {errors}} = useForm({
        defaultValues: {login: '', password: ''}
    });

    return (
        <div className='h-screen w-screen flex justify-center items-center'>
            <h1 className='flex text-xl font-light text-[#1A8A42] pt-5'>Авторизація</h1>
            <form onSubmit={handleSubmit((data) => handleLogin(data.login, data.password))}
                  className='flex flex-col gap-5 pt-5 pb-7'>
                <div className='flex w-80 h-10'>
                    <Input validationProps={register('login', {required: true})}
                           placeholder='Логін' styles={`${errors.login && 'border-red-300'}`}/>
                </div>
                <div className='flex w-80 h-10'>
                    <Input validationProps={register('password', {required: true})} isPassword={true}
                           placeholder='Пароль' styles={`${errors.password && 'border-red-300'}`}/>
                </div>
                <div className='flex w-80 h-10'>
                    <Button isLoading={isDataLoading} isSubmit={true} label='Вхід'/>
                </div>
            </form>
        </div>
    )
}
```

Рисунок 8.4 — Компонента сторінки авторизації

Наступним етапом буде формування маршрутизації у додатку. Наразі найпопулярнішою бібліотекою для цих цілей є React Router. Вона необхідна для того, щоб користувач міг вільно перемикатись між сторінками без їх постійного перезавантаження. React Router використовує спеціальну структуру для виклику тих чи інших компонентів з метою відображення необхідної інформації.

Розглянемо систему маршрутизації, що використовується в нашому додатку для обробки різних ролей користувачів, які зображені на рисунку 8.5.

```
useEffect( effect: () => {
    if (authorizedUserData !== null) {
        navigate(authorizedUserData.role);
    } else {
        navigate('/');
    }
}, deps: [authorizedUserData])

return (
    <Routes>
        <Route path='/' element={<LoginPage/>}/>
        {(id && role === 'admin') && <Route path='/admin' element={<AdminAccount/>}/>}
        {(id && role === 'student') && <Route path='/student' element={<StudentAccount/>}/>}
        {(id && role === 'lecturer') && <Route path='/lecturer' element={<LecturerAccount/>}/>}
    </Routes>
)
}
```

Рисунок 8.5 — Система маршрутизації додатку

На найвищому рівні розташована обгортка <Routes>, всередині якої є чотири маршрути <Route>. У кожного з них є певні умови для відображення:

– ідентифікатор авторизованого користувача;

– роль авторизованого користувача.

Якщо останній параметр співпадає зі шляхом у пошуковому рядку браузера, то React Router автоматично відмальовує особистий кабінет для студента, викладача чи адміністратора відповідно. Також у цій системі бере участь useEffect — спеціальна функція, яка підписується на будь-які оновлення даних про авторизованого користувача та автоматично перенаправляє його на потрібний шлях у браузері.

Завершальним етапом є підключення Redux, бібліотеки для глобального керування даними. Для цього використовується спеціальна конструкція, що складається з thunk функцій, функцій редукторів та функцій конфігурації сховища. Розглянемо їх по порядку.

Thunk функція зображена на рисунку 8.6. ЇЇ основна ціль — виконання асинхронного запиту до бази даних з подальшою передачею отриманої відповіді у функцію редуктор. В даному випадку ми робимо запит на оновлення документу з інформацією про етап застосунку у базі даних. У разі позитивної відповіді повертаємо ці дані, а у разі негативної відповіді - повертаємо помилку.

```
export const updateAppStage = createAsyncThunk<AppStage, AppStage, { rejectValue: string }>(
    'app/updateAppStage',
    async (appStage, {rejectWithValue}) => {
        try {
            const appRef = doc(db,  path: "app",  pathSegments: "info");
            await updateDoc(appRef,  data: {stage: appStage});
            return appStage
        } catch (err: any) {
            return rejectWithValue(err.message)
        }
    }
)
```

Рисунок 8.6 — Thunk функція

Функція редуктор зображена на рисунку 8.7. ЇЇ основна ціль — оновлення конкретної частини даних, що містяться в локальному сховищі.

```
const appSlice = createSlice({
    name: 'app',
    initialState: initialState,
    reducers: {},
    extraReducers: (builder) => {
        builder
            // Update App Stage
            .addCase(updateAppStage.pending, (state) => {
                state.isDataLoading = true
            })
            .addCase(updateAppStage.fulfilled, (state, action) => {
                state.stage = action.payload
                toast.success('Перехід успішний')
                state.isDataLoading = false
            })
```

Рисунок 8.7 — Функція редуктор

У даному випадку ми оновлюємо інформацію про етап застосунку за допомогою параметрів, отриманих з Thunk функції. Але це відбувається лише при успішному отриманні таких даних. В разі отримання помилки ми відображаємо користувачу інформаційне повідомлення, а під час виконання самої Thunk функції — показуємо користувачу спінер.

Файл конфігурації хмарного сховища Firebase зображений на рисунку 8.8.

```
const rootReducer = combineReducers({app: appSlice})

const store = configureStore({
    reducer: rootReducer,
})

export default store;
```

Рисунок 8.8 — Файл конфігурації сховища

Отже для створення єдиної системи, необхідно спочатку об’єднати всі редуктори за допомогою функції combineReducers. Таких редукторів може бути декілька, під операції з різними частинами глобального стану. В результаті ми отримуємо головний, кореневий редуктор. Його вже потрібно передати у функцію configureStore, для ініціалізації глобального сховища даних, а в кінці отриманий об’єкт експортувати для використання у коді.

## 8.2 Бекенд частина застосунку

Оскільки для бекенд частини застосунку використовується бекенд-як-сервіс (англ. backend-as-a-service) рішення від Firebase, його необхідно підключити до додатку через відповідний комплект для розробки програмного забезпечення (англ. software development kit). SDK — це набір інструментів для створення програмного забезпечення для певної платформи [27].

Файл з підключенням Firebase SDK зображений на рисунку 8.9.

```
const firebaseConfig = {
    apiKey: "<API_KEY>",
    authDomain: "<PROJECT_ID>.firebaseapp.com",
    databaseURL: "https://<DATABASE_NAME>.firebaseio.com",
    projectId: "<PROJECT_ID>",
    storageBucket: "<BUCKET>.appspot.com",
    messagingSenderId: "<SENDER_ID>",
};

const app = initializeApp(firebaseConfig);
const db = getFirestore(app);

export default db;
```

Рисунок 8.9 — Файл підключення Firebase SDK

Об’єкт конфігурації, який можна отримати у розділі “Налаштування”, в особистому кабінеті Firebase, складається з персоналізованих даних:

- API ключ;
- ідентифікатор проєкту;
- посилання на базу даних;
- та інші.

За допомогою даного об’єкту ми ініціалізуємо сутність Firebase у додатку та отримуємо доступ до усіх сервісів Firebase в подальшому.

Для реалізації попередньо спроєктованої схеми бази даних (рисунок 8.10) необхідно створити базові колекції (app, subjects та users) через елементи керування у графічному інтерфейсі.

Колекції — це фактично таблиці у яких і будуть зберігатись відповідні сутності. Якщо колекції subjects та users будуть заповнюватись даними у потрібному форматі динамічно, то для колекції app нам необхідно створити статичний документ info з полем stage. Він буде оновлюватись в разі переходу системи на наступний етап.

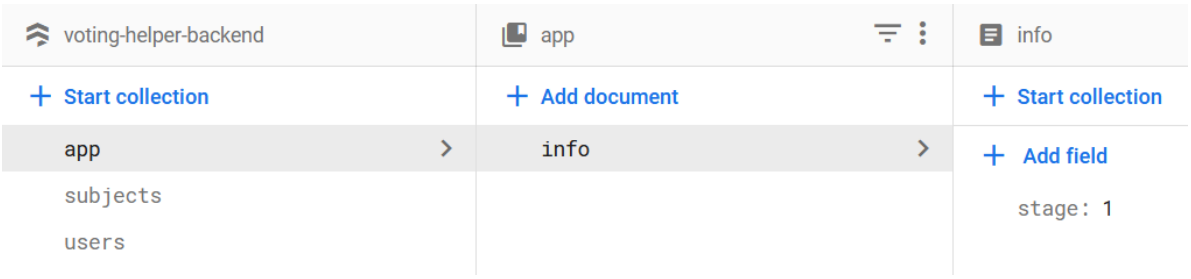



Рисунок 8.10 — Графічний інтерфейс бази даних

Раніше зазначалось, що сервіс Firebase Cloud Functions використовується для виконання ресурсозатратних або блокуючих операцій. Цей опис ідеально підходить під процес завантаження списків, що виконується адміністратором. Код хмарної функції наведений у Додатку А.

Дана функція спрацьовує на сервері автоматично, по завантаженню нового файлу в Firebase Storage — спеціалізоване сховище файлів. Як пам'ятаємо, за відправку заповненої форми зі списками відповідає адміністратор. Всередині хмарної функції ми маємо доступ до завантаженого файла, завдяки чому можемо дізнатись його назву та перевірити у якій папці, в залежності від змісту, він знаходиться:

- subjects;
- lecturers;
- students;
- students_updated.

Це необхідно для того, щоб після зчитування файлу та отримання інформації потрапити у потрібну гілку коду та здійснити фінальну обробку даних та їх коректне завантаження у базу даних.

Також під час завантаження списків студентів та викладачів має відбутись їх автоматична реєстрація в системі. Цю операцію можна автоматизувати та виконувати в хмарі, як і парсинг файлів. Отже після потрапляння у гілку коду, що відповідає за реєстрацію користувачів — викликається функція registerUsers.

Усе, що відбувається надалі — це перебір кожного з користувачів в отриманому списку та його реєстрація за допомогою запропонованого Firebase метода по логіну та паролю.

Якщо реєстрація пройшла успішно — нові користувачі будуть відображені у таблиці, в особистому кабінеті Firebase.

## 8.3 Тестування застосунку

По завершенню розробки релізної версії інформаційної системи для вибіркових дисциплін, можна приступити до її тестування.

Тестування окремих частин коду проводиться для того, щоб впевнитись у коректній та передбачуваній роботі застосунку, при його використанні кінцевими користувачами. Наявність тестів підвищує надійність системи, зменшує кількість потенційних помилок та полегшує її підтримку та подальше вдосконалення. Простими словами — це спосіб перевірити, чи код виконує саме те, що задумали розробники.

При цьому, тестування також містить потенційні недоліки [28]:

- написання тестів займає багато часу;
- за певних сценаріїв, виконання тестів може коштувати реальних грошей;
- у разі неякісного написання тестів, вони можуть видавати помилкові результати та вводити розробників в оману.

Існує велика кількість підходів до тестування, тому вибір того чи іншого варіанту залежить переважно від особливостей архітектури системи та конкретних вимог до її тестування. У даному випадку буде використовуватись модульне тестування (англ. unit testing), за допомогою React Testing Library.

React Testing Library — це бібліотека, що містить набір пакетів з комплексними рішеннями для тестування найпопулярніших фреймворків та бібліотек

(React, Vue, Angular). ЇЇ основна концепція — тести повинні бути максимально подібними до реальної взаємодії користувача з системою.

```
const dummyData = 'Вихід'

test( title: 'Button component renders & callback fires', config: () => {
  const handleClick = jest.fn()
  const {getByText} = render(<Button onClick={handleClick}>{dummyData}</Button>)

  expect(getByText(dummyData)).toBeInTheDocument()
  userEvent.click(getByText(dummyData))
  expect(handleClick).toHaveBeenCalledTimes( expected: 1)
})
```

Рисунок 8.11 — Тест компоненти Button

У тесті на рисунку 8.11 ми перевіряємо, чи компонента Button коректно відмальовується на сторінці, а також чи спрацьовує задана callback-функція при натисканні на неї користувачем.

## 8.4 Висновки до розділу

В розділі представлено загальний опис реалізації інформаційної системи для вибіркових дисциплін. У описі фронтенд частині для цього використовувались окремі різнорівневі компоненти, на прикладі яких і демонструвався алгоритм дій. Оскільки сучасна фронтенд розробка ведеться переважно за патернами — то написання решти коду є концептуально ідентичним.

У описі бекенд частини були детально розглянуті хмарні функції для парсингу файлів та реєстрації користувачів, кінцевою метою яких є зменшення навантаження на клієнтську частину додатку. А також описані конфігураційний файл для підключення Firebase SDK та реалізована структура бази даних за допомогою Cloud Firestore.

# ВИСНОВКИ

В результаті виконання індивідуального дослідницького проєкту було виконано основну мету — розширення функціоналу існуючих систем для електронного голосування українських вишів за рахунок створення спеціалізованої інформаційної системи.

Результатом виконання є робоча модель інформаційної системи для вибіркових дисциплін на основі системи голосування “Електронного Кампусу” КПІ.

У роботі було розглянуто та проаналізовано існуючі рішення в області електронних освітніх систем на прикладі інформаційних систем для голосування чотирьох популярних українських вишів:

- Київський політехнічний інститут;
- Львівський національний університет;
- Національний авіаційний університет;
- Запорізький національний університет.

Також в ході проєктування було розроблено наступну документацію:

- діаграма прецедентів;
- макет з UI/UX дизайном;
- схема бази даних;
- структурна схема.

Уся зазначена документація змістовно відображає архітектуру системи, взаємодію програмних елементів, будову та можливий функціонал.

Процес створення фронтенд та бекенд частин інформаційної системи для вибіркових дисциплін описано у ключових деталях у відповідному розділі дослідницької роботи. Також наведений приклад тестування коду. Серед використаних технологій: React, React Testing Library, Redux, Tailwind та Firebase. Доступ до розробленої інформаційної системи можна отримати з будь-якого пристрою, що має веб-браузер та підключення до мережі Інтернет.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Сервіс автоматизованого запису на вибіркові дисципліни. URL: http://ekmair.ukma.edu.ua/bitstream/handle/123456789/11110/Horbo-rukov_Oletskyi_Sevis_avtomatyzovanoho_zapysu.pdf (дата звернення: 02.05.2022).
2. Вибіркові дисципліни. Матеріал з Wikiситет. URL: http://wiki.nuwm.edu.ua/index.php (дата звернення: 04.05.2022).
3. Дистанційне навчання у закладах вищої освіти: ризики та перспективи розвитку. URL: https://nubip.edu.ua/node/100736 (дата звернення: 05.05.2022).
4. Інформатизація освіти: стан та перспективи впровадження. URL: https://lib.iitta.gov.ua/710965/1/dyg-2018-009-block-7-16.pdf (дата звернення: 05.05.2022).
5. Інформатизація освіти як основа розвитку інформаційного суспільства. URL: http://dspace.tnpu.edu.ua/bitstream/123456789/15382/1/39_Yordan_Yordan. pdf (дата звернення: 09.05.2022).
6. Дистанційне навчання: реалії і перспективи. URL: https://visnyk.naps.gov.ua/index.php/journal/article/view/72 (дата звернення: 09.05.2022).
7. Вибір без вибору: моніторинг вибіркових курсів у державних вишах. URL: https://cedos.org.ua/researches/vybir-bez-vyboru-monitorynh-vybirkovykh-kursiv-u-derzhavnykh-vyshakh (дата звернення: 13.05.2022).
8. Процес вибору вибіркових дисциплін. КПІ. URL: https://telegra.ph/choose-your-fighter-02-28 (дата звернення: 14.05.2022).
9. Покрокова Інструкція для роботи здобувача вищої освіти в автоматизованій системі “Формування індивідуальної освітньої траєкторії”. НАУ. URL: https://nau.edu.ua/download/trajectory/manual_FIET_stud.pdf (дата звернення: 15.05.2022).

10. Процес реєстрації на вибіркові дисциплін. ЛНУ. URL: https://law.lnu.edu.ua/wp-content/uploads/2021/02/Reiestratsiia-na-vybirkovi-dystsypliny.pdf (дата звернення: 17.05.2022).

11. MoodleDocs — About Moodle. URL: https://docs.moode.org/400/en/About _Moodle (дата звернення: 17.05.2022).

12. Інструкція для здобувачів освіти. ЗНУ. URL: https://moodle.znu.edu.ua/mod/page/view.php?id=49165 (дата звернення: 19.05.2022).

13. Use case diagram. URL: https://www.quality-assurance-group.com/use-case-diagrams (дата звернення: 20.05.2022).

14. Importance of UI/UX design in software development. URL: https://medium.com/@tobiegbude/importance-of-ui-ux-design-in-software-development-c21b342c2f 12 (дата звернення: 23.05.2022).

15. What is a CSV file and how do I open it? URL: https://www.how-togeek.com/348960/what-is-a-csv-file-and-how-do-i-open-it (дата звернення: 23.05.2022).

16. React. Документація. URL: https://reactjs.org (дата звернення: 27.05.2022).

17. Redux. Документація. URL: https://redux.js.org (дата звернення: 8.05.2022).

18. Tailwind CSS. Документація. URL: https://tailwindcss.com (дата звернення: 29.05.2022).

19. Firebase. Документація. URL: https://firebase.google.com/products-build (дата звернення: 02.06.2022).

20. Cloud Firestore. Документація. URL: https://firebase.google.com/docs/firestore (дата звернення: 03.06.2022).

21. Cloud Functions for Firebase. Документація. URL: https://firebase.google.com/docs/functions (дата звернення: 03.06.2022).

22. Firebase Authentication. Документація. URL: https://firebase.google.com/docs/auth (дата звернення: 03.06.2022).

23. Firebase Hosting. Документація. URL: https://firebase.google.com/docs/hosting (дата звернення: 05.06.2022).

24. What is a database schema? URL: https://careerkarma.com/blog/database-schema (дата звернення: 05.06.2022).

25. What is an architecture diagram, and why do you need one? URL: https://cacoo.com/blog/what-is-an-architecture-diagram-and-why-do-you-need-one (дата звернення: 08.06.2022).

26. What is software implementation? URL: https://blogs.opentext.com/what-is-software-implementation (дата звернення: 10.06.2022).

27. SDK vs API: what’s the difference? URL: https://www.ibm.com/cloud/blog/sdk-vs-api (дата звернення: 14.06.2022).

28. How to test React components: the complete guide. URL: https://www.freecodecamp.org/news/testing-react-hooks (дата звернення: 17.06.2022).

# ДОДАТОК А

Код хмарної функції для парсингу списків та реєстрації користувачів

```
import {ObjectMetadata} from "firebase-functions/lib/providers/storage";
import {DownloadResponse} from "@google-cloud/storage";
import {Storage} from '@google-cloud/storage';
import {ParseResult} from "papaparse";
import {UserRole} from "../../src/types";

const functions = require("firebase-functions");
const admin = require('firebase-admin');
const papa = require("papaparse");

admin.initializeApp();

interface RawSubjectData {
 id: string,
 name: string,
 votes: number,
 controlType: string,
 department: string,
 ects: number,
 semester: number
}

interface RawUserData {
 login: string;
 password: string;
 name: string;
 role: UserRole;
 subjects: string;
}

exports.handleCSVfiles = functions.storage.object().onFinalize(
 async (object: ObjectMetadata) => {

 const registerUsers = async (data: RawUserData[]) => {
   data.forEach((user) => {
     admin.auth().createUser({
       email: `${user.login}@gmail.com`,
       password: user.password
     });
   })
 };
```

```
const parseUsers = async (
 data: RawUserData[], actionType: 'set' | 'update'
 ) => {
  const parsedUsers = data.map(
      (item) => ({
        id: item.login,
        name: item.name,
        role: item.role,
        subjects: item.subjects
            .slice(0, -1)
            .split(';')
      }))
  if (actionType === 'set') {
    parsedUsers.forEach((item: any) => {
      admin.firestore()
          .collection('users')
          .doc(item.id)
          .set(item)
    })
  } else if (actionType === 'update') {
    parsedUsers.forEach((item: any) => {
      admin.firestore()
          .collection('users')
          .doc(item.id)
          .update({subjects: [...item.subjects]})
    })
  }
}

const parseSubjects = async (data: RawSubjectData[]) => {
  const parsedSubjects = data.map(
      (item) => ({
        id: item.id.slice(0, -1),
        name: item.name,
        votes: item.votes,
        basicInfo: {
          controlType: item.controlType,
          department: item.department,
          ects: item.ects,
          semester: item.semester
        },
        studyingInfo: null,
        introLectureInfo: null
      })
  )
  parsedSubjects.forEach((item: any) => {
```

```
        admin.firestore()
            .collection('subjects')
            .doc(item.id)
            .set(item)
      })
    }

    const storage = new Storage();
    const bucketPath = 'gs://voting-helper-backend.appspot.com';
    const bucket = storage.bucket(bucketPath);
    const filePath = object.name;
    const file = bucket.file(filePath!);

    file.download()
        .then((data: DownloadResponse) => {
          papa.parse(data.toString(),
              {
                header: true, dynamicTyping: true,
                complete:
                  async (response: ParseResult<RawUserData | RawSubjectData>) => {
                  const folderName = filePath?.split('/')[0];
                  if (folderName === 'lecturers' || folderName === 'students') {
                    await parseUsers(response.data as RawUserData[], 'set')
                    await registerUsers(response.data as RawUserData[])
                  } else if (folderName === 'students_updated') {
                    await parseUsers(response.data as RawUserData[], 'update')
                  } else {
                    await parseSubjects(response.data as RawSubjectData[])
                  }
                }
              }
          );
        });
});
```