Міністерство освіти і науки України
Одеський національний політехнічний університет
Інститут комп’ютерних систем
Кафедра інформаційних систем

КУРСОВА РОБОТА

з дисципліни «Технології створення програмних продуктів»

за темою

«Менеджер подій»

Частина №1

Виконала:
студентка 3-го курсу
групи АІ-181
Заморіна Е.І.
Перевірив:
Блажко О. А.

Одеса - 2020

Анотація

В курсовій роботі розглядається процес створення програмного продукту «Менеджер подій». Робота виконувалась в команді з декількох учасників: Заморіна Єлизавета, Карпенко Валерія, Юденцев Олексій. Тому в пояснювальній записці у розділах «Проектування» та «Конструювання» детальніше описано лише одну частину з урахуванням планів проведених робіт з розділу «Планування» з описом особливостей конструювання:

– структур даних моделі MVP в системі керування базами даних PostgreSQL;

– програмних модулів в інструментальному середовищі WebStorm з використанням фреймворку React та мови програмування JavaScript.

Результати роботи розміщено на *github*-репозиторії за адресою: https://github.com/dieundead/EventManagerApp

Перелік скорочень

ОС – операційна система

ІС – інформаційна система

БД – база даних

СКБД – система керування базами даних

ПЗ – програмне забезпечення

ПП– програмний продукт

UML – уніфікована мова моделювання

Зміст

1 Вимоги до програмного продукту

1.1 Визначення потреб споживача

1. Ієрархія потреб споживача

Відомо, що в теорії маркетингу потреби людини можуть бути представлені у вигляді ієрархії потреб ідей американського психолога Абрахама Маслоу включають рівні:

- фізіологія (вода, їжа, житло, сон);
- безпека (особиста, здоров'я, стабільність),
- приналежність (спілкування, дружба, любов),
- визнання (повага оточуючих, самооцінка),
- самовираження (вдосконалення, персональний розвиток).

На рисунку 1.1 представлено рівень потреби споживача, який хотілося б задовольнити, використовуючи майбутній програмний продукт.

Рисунок 1.1.1 – Ієрархія потреби споживача

Був обраний рівень «Самовираження», тому що, використовуючи програмний продукт «Менеджер подій», споживач задовольняє таки потреби, як вдосконалення та персональний розвиток.

1.1.2 Деталізація матеріальної потреби

Для деталізації матеріальної потреби можна скористатися ментальними картами (MindMap). При створенні ментальних карт матеріальна потреба розташовується в центрі карти. Асоціативні гілки можна швидко створити, припускаючи, що в загальному вигляді з об'єктом пов'язані три потоки даних / інформації: вхідний, внутрішній, вихідний. Кожен потік - це асоціативна група,

що включає можливі п'ять гілок, що відповідають на п'ять питань: Хто? Що? Де? Коли? Як?

Відповідно до рекомендацій по створенню ментальних карт кожна гілка-асоціація може бути розділена на додаткові асоціативні гілки, які деталізують відповіді на поставлені питання.

Потреба, яка була визначена при аналізі матеріальних проблем споживача, основні та додаткові асоціативні гілки зображені на Рис 1.2

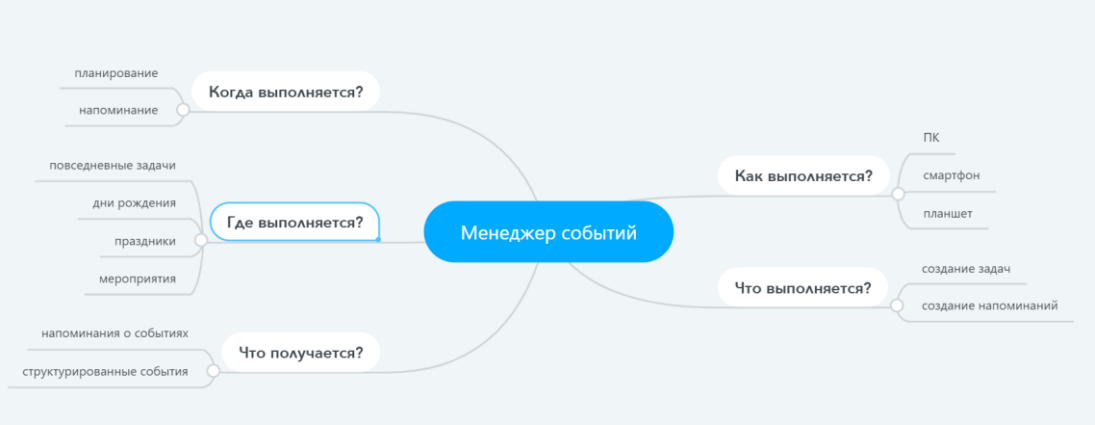


Рисунок 1.1.2 – Деталізація матеріальної потреби

1.2 Бізнес-вимоги до програмного продукту

1.2.1 Опис проблеми споживача

1.2.1.1 Концептуальний опис проблеми споживача

Концептуальний опис проблеми споживача відображений у табл. 1.1

№	Загальний опис проблеми	Метричні показники незадоволеності споживача
1	Неможливо тримати всі справи в голові або скрізь носити свій паперовий щоденник.	Низький рівень доступності групового планування, тобто планування спільних справ сім'ї, співробітників, друзів і т.д.

Таблиця – 1.2.1.1

1.2.1.2 Метричний опис проблеми споживача

Рівень доступності AL (AL – Access Level) можна визначити як

AL = NA / N,

де NA – кількість користувачів, які скористаються можливістю групового планування подій;

N – загальна кількість користувачів.

1.2.2 Мета створення програмного продукту

1.2.2.1 Проблемний аналіз існуючих програмних продуктів

Проблемний аналіз існуючих програмних продуктів відображена у табл.2.1

Таблиця 1.2.2.1

№	Назва продукту	Вартість	Ступінь готовності	Примітка
1	Google Calendar	Безкоштовно	1	Неможливість вибору категорії події
2	Мініплан	Безкоштовно	1	Застарілий інтерфейс
3	Evernote	Безкоштовно	1	Неможливість ділитися подіями

1.2.2.2 Мета створення програмного продукту

Створення менеджера подій – досить актуальна на сьогоднішній день тема. Оскільки у сучасному світі усе досить швидко змінюється, доводиться робити багато планів, які іноді досить складно утримувати в голові.

1.2.3 Назва програмного продукту

Назва «Event manager» відображає мету та гасло програмного продукту.

1.2.3.1 Гасло програмного продукту

Гасло — лаконічна фраза, що впадає в око, добре запам'ятовується та висловлює суть продукту.

На початковому етапі розробки було вигадано гасло, яке найкращим чином відображає мету та суть роботи веб-сервісу та пов’язано з логотипом та назвою.

Гасло - «Час, витрачений на гарне планування, має властивість компенсуватися!»

1.2.3.2 Логотип програмного продукту

Відмінним способом представлення назви програмного продукту є його логотип, що поєднує зорові образи.

На рис 1.2.3.2 можна побачити розроблений командою логотип, який відображає назву веб-сервісу.

Рисунок - 1.2.3.2 - Логотип

1.3 Вимоги користувача до програмного продукту

1.3.1 Історія користувача програмного продукту

Гість може:

- переглядати загальнодоступні події;
- отримувати сповіщення про них.

Зареєстрований користувач може:

- додавати власні події;
- редагувати події;
- видаляти події з календаря;
- сортувати події по їх типу;
- отримувати оповіщення тільки про певні типи подій;

- встановлювати час і дату сповіщень;
- створювати сім`ю;
- ділитися нагадуваннями з учасниками сім'ї;
- розподіляти привілеї в сім'ї.

1.3.2 Діаграма прецедентів програмного продукту

Діаграма прецедентів (Use Case UML-діаграма) включає:

− актори (зацікавлені особи і зовнішні системи зі своїм API);

− прецеденти як основні функції ПП;

− зв'язки між прецедентами і акторами як множиною зацікавлених осіб;

− можливі зв'язки-узагальнення між акторами.

Діаграма прецедентів програмного продукту зображена на рис. 1.3.1

Рисунок - 1.3.2. Діаграма прецедентів програмного продукту

1.3.3 Сценарії використання прецедентів програмного продукту

Для кожного прецедента описаний сценарій використання з урахуванням пунктів:

− назва прецеденту;

− передумови початку виконання прецеденту;

− актори як зацікавлені особи у виконанні прецеденту;

− актор-основна зацікавлена особа як ініціатор початку прецеденту;

− гарантії успіху (що отримають актори у разі успішного завершення прецеденту);

− основний успішний сценарій;

− альтернативні сценарії, прив'язані до кроків основного успішного сценарію.

- Реєстрація

Актори: користувач (неавторизований)

Передумови: відкрита сторінка реєстрації.

Постумови: користувача зареєстровано.

Основний сценарій:

1. Користувач відкриває сторінку реєстрації.
2. Система відображає форму реєстрації.
3. Користувач вводить ім’я та пароль для реєстрації.
4. Користувача зареєстровано.

- Перегляд подій по категоріям

Актори: користувач (авторизований, неавторизований)

Передумови: відкрита сторінка подій.

Постумови: події відсортовані за категоріями.

Основний сценарій:

1. Користувач відкриває сторінку подій.
2. Система відображає список усіх загальнодоступних подій.
3. Користувач натискає на кнопку типу події.
4. Виведені обрані події.

- Авторизація

Актори: користувач (неавторизований)

Передумови: відкрита сторінка авторизації.

Постумови: користувача авторизовано.

Основний сценарій:

1. Користувач відкриває сторінку авторизації.
2. Система відображає форму авторизації.
3. Користувач вводить ім’я та пароль для авторизації.
4. Користувача авторизовано.

- Додавання власних подій

Актори: користувач (авторизований)

Передумови: відкрита сторінка «Додати подію».

Постумови: подія/нагадування додані.

Основний сценарій:

1. Користувач відкриває сторінку «Додати подію».
2. Система відображає форму додавання події.
3. Користувач вводить дату/час, назву і по бажанню опис події у форму.
4. Подію додано.

- Редагування власних подій

Актори: користувач (авторизований)

Передумови: відкрита сторінка «Редагувати подію».

Постумови: подія/нагадування відредаговані.

Основний сценарій:

1. Користувач відкриває сторінку «Редагувати подію».
2. Система відображає форму редагування події.
3. Користувач змінює дату/час, назву і по бажанню опис події у форму.
4. Подію відредаговано.

- Видалення власних подій

Актори: користувач (авторизований)

Передумови: відкрита сторінка «Редагувати подію».

Постумови: подія/нагадування видалені.

Основний сценарій:

1. Користувач відкриває сторінку «Редагувати подію».
2. Система відображає кнопку видалення події.
3. Користувач натискає кнопку «Видалити подію».
4. Подію видалено.

- Встановлення часу і дати нагадувань

Актори: користувач (авторизований)

Передумови: відкрита сторінка «Редагувати подію».

Постумови: встановлені час та дата нагадування про подію.

Основний сценарій:

1. Користувач відкриває сторінку «Редагувати подію».
2. Система відображає форму редагування події.
3. Користувач встановлює дату/час нагадування.
4. Час і дата нагадування встановлені.

- Створення родини

Актори: користувач (авторизований)

Передумови: відкрита сторінка «Створення родини».

Постумови: родина створена.

Основний сценарій:

1. Користувач відкриває сторінку «Створення родини».
2. Система відображає форму створення та налаштування родини.
3. Користувач обирає зареєстрованого користувача та додає його у родину.
4. Родину створено.

- Поділ подіями

  Актори: користувач (авторизований)

  Передумови: відкрита сторінка «Поділитися подією».

  Постумови: успішний поділ подією.

  Основний сценарій:

1. Користувач відкриває сторінку «Поділитися подією».
2. Система відображає форму поділу подією.
3. Користувач обирає зареєстрованого користувача та відправляє йому свою подію.
4. Подія відправлена.

- Розподіл привілей у родині

  Актори: користувач (авторизований)

  Передумови: відкрита сторінка «Налаштування родини».

  Постумови: привілеї розподілені.

  Основний сценарій:

1. Користувач відкриває сторінку «Налаштування родини».
2. Система відображає форму налаштування родини.
3. Користувач обирає зареєстрованого користувача та події, до яких він матиме доступ.
4. Родину створено.

1.4 Функціональні вимоги до програмного продукту

1.4.1. Багаторівнева класифікація функціональних вимог

Таблиця – 1.4.1. Багаторівнева класифікація функціональних вимог

Ідентифікатор функції	Назва функції
FR1	Авторизація користувача
FR1.1	Створення запиту у користувача на отримання його параметрів ідентифікації та аутентифікації

FR1.2	Передача від користувача його параметрів ідентифікації та аутентифікації
FR1.3	Пошук інформації у базі даних користувачів
FR1.4	Створення сесії для користувача
FR2	Створення події
FR2.1	Додавання події до бази даних
FR3	Видалення події
FR3.1	Видалення події з бази даних
FR4	Редагування події
FR4.1	Редагування події у базі даних
FR5	Встановлення часу і дати нагадувань
FR5.1	Створення запиту у користувача на отримання дати та часу нагадування
FR5.2	Передача від користувача параметрів дати та часу нагадування
FR5.3	Додавання нагадування до існуючої події у базі даних
FR6	Перегляд подій по категоріям
FR6.1	Користувач обирає категорії подій, які він хоче подивитись
FR6.2	З бази даних відображаються обрані події
FR7	Створення сім’ї
FR7.1	Додавання сім’ї до бази даних

FR8	Поділ подіями
FR8.1	Створення запиту для того щоб поділитися подією
FR8.2	Копіювання інформації про подію з бази даних
FR8.3	Відправка інформації про подію
FR9	Розподіл привілеїв
FR9.1	Створення запиту у користувача на отримання форми налаштування сім'ї
FR9.2	Передача від користувача його параметрів налаштування привілей у сім'ї

1.4.2 Функціональний аналіз існуючих програмних продуктів

Таблиця – 4.2. Функціональний аналіз існуючих програмних продуктів

Ідентифікатор функції	Google Calendar	Миниплан
FR1	+	+
FR2	+	+
FR3	+	+
FR4	+	+
FR5	+	-
FR6	-	+
FR7	-	-

1.5 Нефункціональні вимоги до програмного продукту

1.5.1 Опис зовнішніх інтерфейсів

1.5.1.1 Опис інтерфейса користувача

1.5.1.1.1 Опис INPUT-інтерфейса користувача

Таблиця – 1.5.1.1.1. Опис INPUT-інтерфейса користувача

Ідентифікатор функції	Засіб INPUT-потоку	Особливості використання
FR1.1	стандартна комп`ютерна клавіатура	
FR4.1	стандартна комп`ютерна клавіатура	
FR4.1	2/3-кнопочний маніпулятор типу "миша"	Використання лівої кнопки миші для завершення процесу вводу даних
FR5.1	стандартна комп`ютерна клавіатура	
FR5.1	2/3-кнопочний маніпулятор типу "миша"	Використання лівої кнопки миші для завершення процесу вводу даних
FR6.1	2/3-кнопочний маніпулятор типу "миша"	Використання лівої кнопки миші для вибору виведення даних

1.5.1.1.2 Опис OUTPUT-інтерфейса користувача

Для функції «FR 1.1» OUTPUT-інтерфейса користувача.

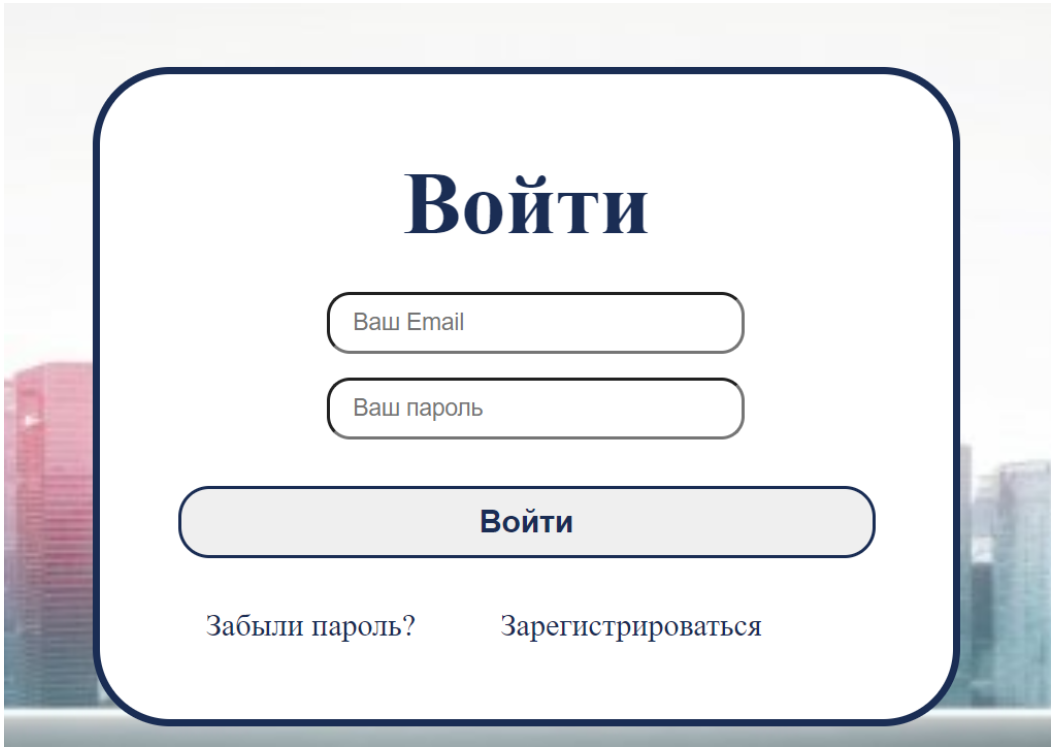


Рисунок 1.5.1.1.2.1 – Дизайн функції FR1

Для функції «FR 5.1» OUTPUT-інтерфейса користувача.

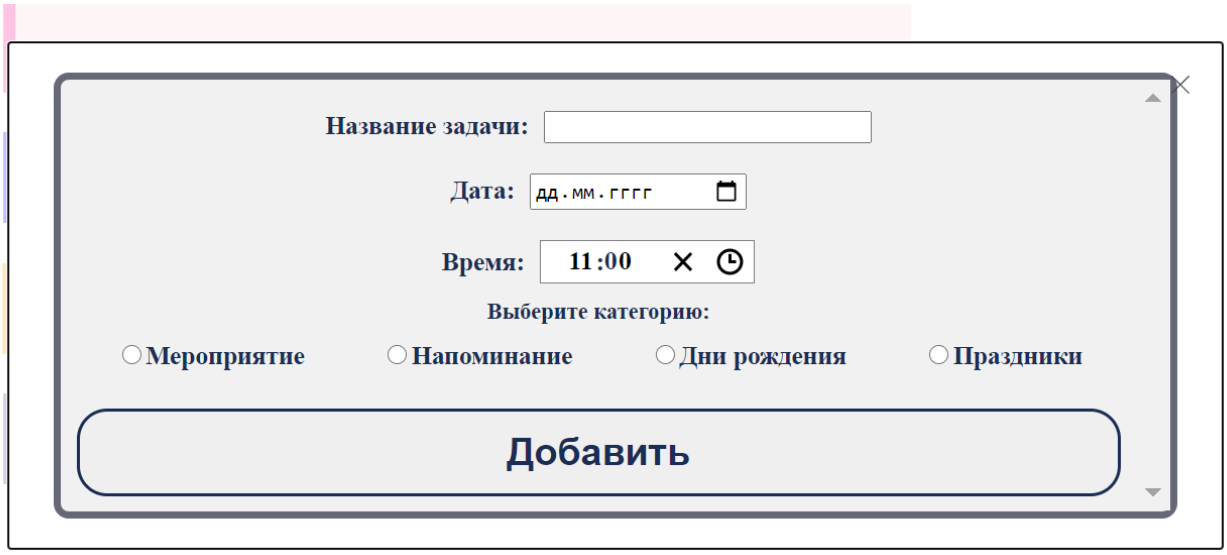


Рисунок 1.5.1.1.2.2 – Дизайн функції FR5.1

Для функції «FR6.1» OUTPUT-інтерфейса користувача.

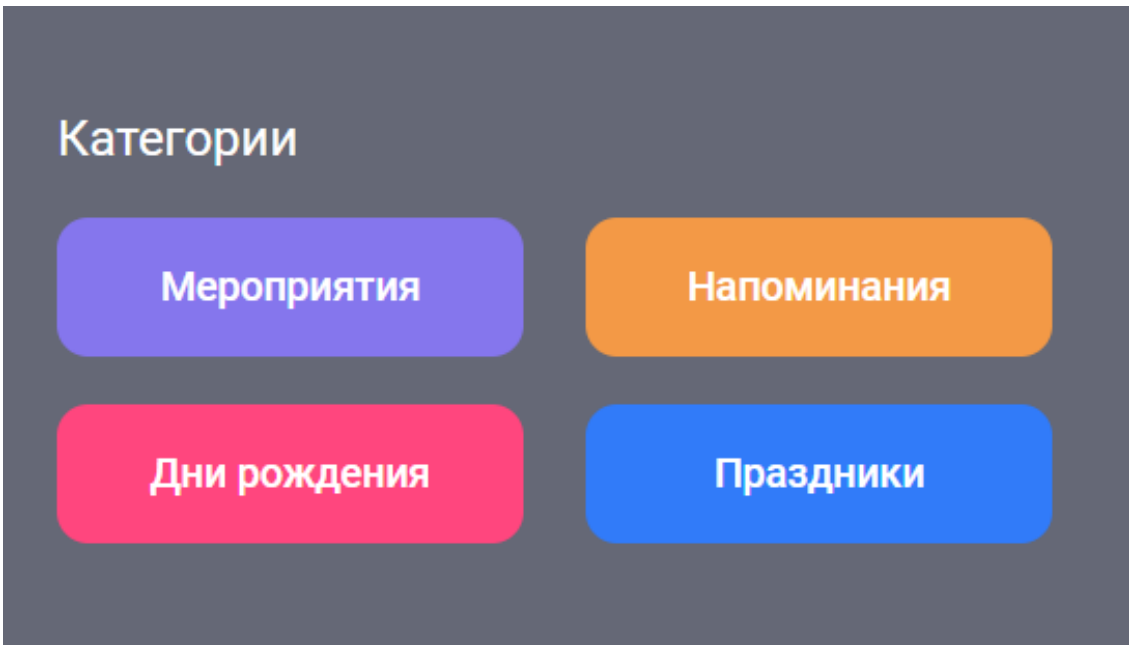


Рисунок 1.5.1.1.2.3 – Дизайн функції FR6.1

1.5.1.2 Опис інтерфейсу із зовнішніми пристроями

FR1.1 – Desktop, Notebook, смартфон

FR4.1 – Desktop, Notebook, смартфон

FR5.1 – Desktop, Notebook, смартфон

FR6.1 – Desktop, Notebook, смартфон

1.5.1.3 Опис програмних інтерфейсів

Даний ПП працює на будь-яких операційних системах, певна версія програмного забезпечення не потрібна, бо ПП адаптований під усі.

Взаємодія ПП із зовнішніми ПП не передбачається.

1.5.1.4 Опис інтерфейсів передачі інформації

Інтерфейси передачі інформації для реалізації більшості функцій ПП:

Провідні інтерфейси:

- Ethernet

Безпровідні інтерфейси:

- Wi-Fi

1.5.1.5 Опис атрибутів продуктивності

Ідентифікатор функції	Максимальний час реакції ПП на дії користувача, секунди
FR1.1	4
FR4.1	3
FR5.1	3
FR6.1	3

2 Планування процесу розробки програмного продукту

2.1 Планування ітерацій розробки програмного продукту

З метою забезпечення для вимог таких рекомендацій IEEE-стандарту, як необхідність, корисність при експлуатації, здійсненність функціональних вимог до ПП, були визначені функціональні пріоритети, які будуть використані при плануванні ітерацій розробки ПП.

2.2 Концептуальний опис архітектури програмного продукту

Концептуальний опис архітектури програмного продукту зображено на рис. 2.2

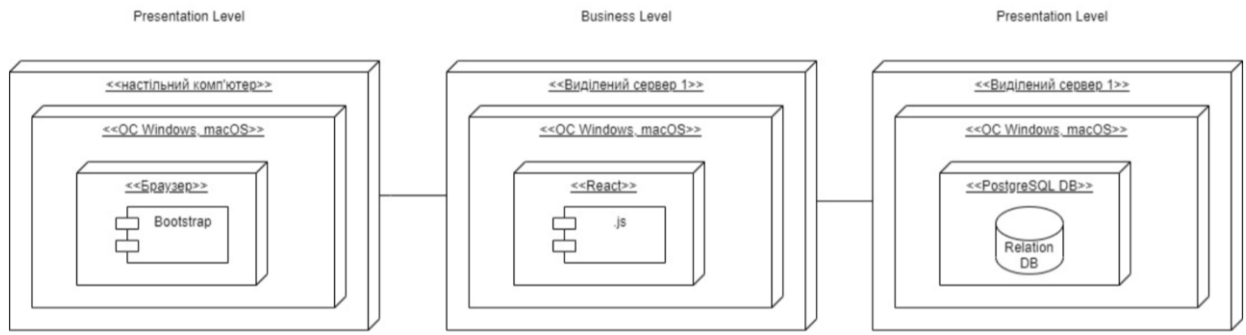


Рисунок – 2.2. Концептуальний опис архітектури програмного продукту

2.3 План розробки програмного продукту

З метою забезпечення для вимог таких рекомендацій IEEE-стандарту, як необхідність, корисність при експлуатації, здійсненність функціональних вимог до ПП, було визначено функціональні пріоритети, які будуть використані при плануванні ітерацій розробки ПП.

При створенні пріоритетів необхідно врахувати:

− сценарні залежності між прецедентами, до яких належать функції, на основі аналізу пунктів передумов початку роботи прецедентів, вказаних в описі сценаріїв роботи прецедентів;

− вплив роботи прецеденту, до якого належить функція, на досягнення

мети ПП, наприклад у відсотках, на основі аналізу пунктів гарантій успіху, вказаних в описі сценаріїв роботи прецедентів.

Сценарні залежності будуть перетворені у відповідні функціональні залежності. Вплив роботи прецеденту буде поширено на всі підлеглі функції ієрархії. При визначенні пріоритетів рекомендується використовувати наступні позначки:

− M (Must) – функція повинна бути реалізованою у перших ітераціях за будь-яких обставин;

− S (Should) – функція повинна бути реалізованої у перших ітераціях, якщо це взагалі можливо;

− C (Could) – функція може бути реалізованої, якщо це не вплине негативно на строки розробки;

− W (Want) – функція може бути реалізованої у наступних ітераціях.

Опису представлено в таблиці 2.3

Таблиця 2.3 – Опис функціональних пріоритетів

Ідентифікатор функції	Функціональні залежності	Вплив на досягнення мети, %	Пріоритет функції
FR1		0	M
FR2		10	M
FR3	FR2	0	S
FR4	FR2	0	S
FR5	FR2	10	C
FR6		30	M
FR7		20	W
FR8		20	W
FR9	FR7	10	W

2.3.1 Оцінка трудомісткості розробки програмного продукту

Визначення нескорегованого показника UUCP (Unadjusted Use Case

Points)

1. Визначення вагових показників акторів A

Всі актори діляться на три типи: прості, середні і складні.

Простий актор представляє зовнішню систему з чітко визначеним Програмним інтерфейсом.

Середній актор представляє або зовнішню систему, що взаємодіє з ПП за допомогою мережевих протоколів, або особистість, що користується текстовим інтерфейсом (наприклад, алфавітно-цифровим терміналом).

Складний актор представляє особистість, що користується графічним інтерфейсом. Загальна кількість акторів кожного типу помножується на відповідний ваговий коефіцієнт, потім обчислюється загальний ваговій показник (табл.2.3.1.1)

Таблиця - 2.3.1.1.Вагові коефіцієнти акторів

Тип актора	Ваговий коефіцієнт	Кількість
Простий	1	0
Середній	2	0
Складний	3	3

Визначення UUCP

A = 9

UC = 75

UUCP=A+UC = 84

Технічна складність проекту (TCF - Technical Complexity Factor) обчислюється з урахуванням показників технічної складності.

Кожному показнику присвоюється значення STi в діапазоні від 0 до 5:

1) 0 означає відсутність значимості показника для даного проекту;
2) 5 - високу значимість.

Показники технічної складності проекту TCF представлено в таблиці 2.3.1.2

Таблиця – 2.3.1.2. Показники технічної складності проекту

Показник	Опис	Вага	ST
T1	Розподілена система	2	2
T2	Висока продуктивність (пропускна здатність)	1	3
T3	Робота кінцевих користувачів в режимі он-лайн	1	1
T4	Складна обробка даних	-1	2
T5	Повторне використання коду	1	2
T6	Простота установки	0,5	0
T7	Простота використання	0,5	1
T8	Переносимість	2	2
T9	Простота внесення змін	1	2
T10	Паралелізм	1	2
T11	Спеціальні вимоги до безпеки	1	1
T12	Безпосередній доступ до системи з боку зовнішніх користувачів	1	2

T13	Спеціальні вимоги до навчання користувачів	1	1

Значення TCF

TCF = 0,6 + (0,01 * (STi * Вага_i)) = 0,805

Рівень кваліфікації розробників (EF - Environmental Factor) обчислюється з урахуванням наступних показників (табл. 2.3.1.3).

Таблиця – 2.3.1.3. Показники рівня кваліфікації розробників

Показник	Опис	Вага	SF
F1	Знайомство з технологією	1,5	1
F2	Досвід розробки додатків	0,5	3
F3	Досвід застосування об’єктно-орієнтованого підходу	1	3
F4	Наявність ведучого аналітика	0,5	1
F5	Мотивація	1	5
F6	Стабільність вимог	2	2
F7	Часткова зайнятість	-1	1

F8	Складні мови програмування	-1	1

Обчислення значення EF:

EF = 1,4 + (- 0,03 * (SFi * Вага_i)) = 0,94

Остаточне значення UCP (Use Case Points)

UCP = UUCP * TCF * EF = 63,56

2.3.2 Визначення дерева робіт з розробки програмного продукту

При створенні дерева робіт (Work BreakDown Structure- WBS) використовується дерево функцій, яке було створено раніше.

Кожна функція 1-го рівня ієрархії перетворюється в Work Package (WP)

Кожна функція 2-го рівня ієрархії перетворюється в Work Task (WT).

Для кожної задачі визначаються підзадачі - Work SubTask (WST) з урахуванням базових процесів розробки програмних модулів: проектування, конструювання, модульне тестування, збірка та системне тестування.

WBS представлено на рисунку 2.3.2.1

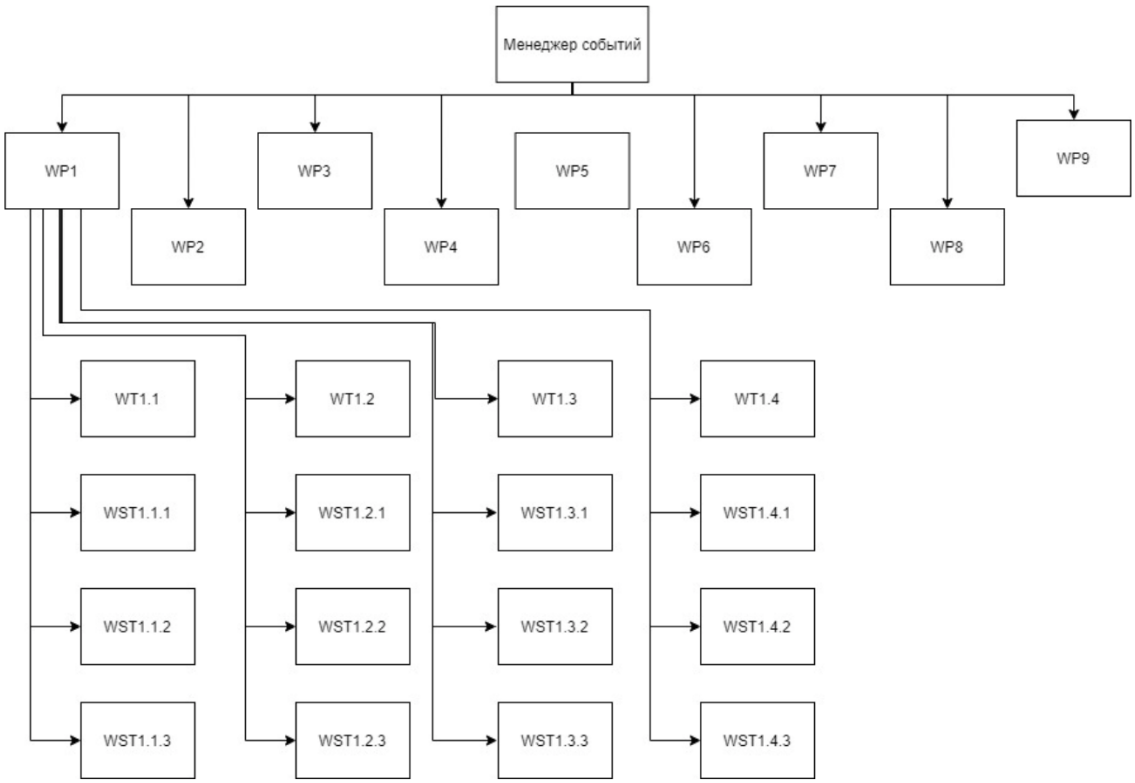


Рисунок – 2.3.2.1 - Дерево робіт

2.3.3 Графік робіт з розробки програмного продукту

2.3.3.1 Таблиця з графіком робіт

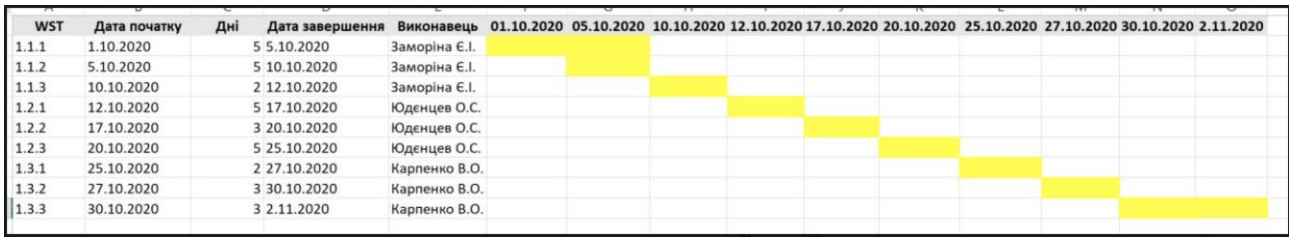
Для кожної підзадачі визначається виконавець, що фіксується у вигляді таблиці, яка представлена в таблиці 2.3.3.1

Таблиця – 2.3.3.1

WST	Дата початку	Дні	Дата завершення	Виконувач
WST1.1.1	1.10.2020	5	5.10.2020	Заморіна Є.І.
WST1.1.2	5.10.2020	5	10.10.2020	Заморіна Є.І.
WST1.1.3	10.10.2020	2	12.10.2020	Заморіна Є.І.
WST1.2.1	12.10.2020	5	17.10.2020	Юдєнцев О.С.
WST1.2.2	17.10.2020	3	20.10.2020	Юдєнцев О.С.
WST1.2.3	20.10.2020	5	25.10.2020	Юдєнцев О.С.
WST1.3.1	25.10.2020	2	27.10.2020	Карпенко В.О.
WST1.3.2	27.10.2020	3	30.10.2020	Карпенко В.О.
WST1.3.3	30.10.2020	3	2.11.2020	Карпенко В.О.

2.3.3.2 Діаграма Ганта

Діаграму Ганта наведено на рисунку 2.3.3.2.



WST	Дата початку	Дні	Дата завершення	Виконавець	01.10.2020	05.10.2020	10.10.2020	12.10.2020	17.10.2020	20.10.2020	25.10.2020	27.10.2020	30.10.2020	2.11.2020
1.1.1	1.10.2020	5	5.10.2020	Заморіна Є.І.										
1.1.2	5.10.2020	5	10.10.2020	Заморіна Є.І.										
1.1.3	10.10.2020	2	12.10.2020	Заморіна Є.І.										
1.2.1	12.10.2020	5	17.10.2020	Юдєнцев О.С.										
1.2.2	17.10.2020	3	20.10.2020	Юдєнцев О.С.										
1.2.3	20.10.2020	5	25.10.2020	Юдєнцев О.С.										
1.3.1	25.10.2020	2	27.10.2020	Карпенко В.О.										
1.3.2	27.10.2020	3	30.10.2020	Карпенко В.О.										
1.3.3	30.10.2020	3	2.11.2020	Карпенко В.О.										

Рисунок 2.3.3.2 – Діаграма Ганта

3. Проектування програмного продукту

3.1 Концептуальне та логічне проектування структур даних програмного продукту

3.1.1 Концептуальне проектування на основі UML-діаграми концептуальних класів

Використовуючи кроки основного успішного та альтернативного сценаріїв роботи прецедентів ПП, було спроектовано UML-діаграми концептуальних класів.

Моделювання проведено з урахуванням наступної послідовності кроків.

1. Визначити імена класів.
2. Визначити імена атрибутів класів.
3. Визначити зв'язку між класами (узагальнення, іменовані асоціації, агреговані асоціації).
4. Для зв'язків-асоціацій визначити назви, кратність і можливість агрегації.
5. Створити UML-діаграму класів в будь-якому графічному редакторі.

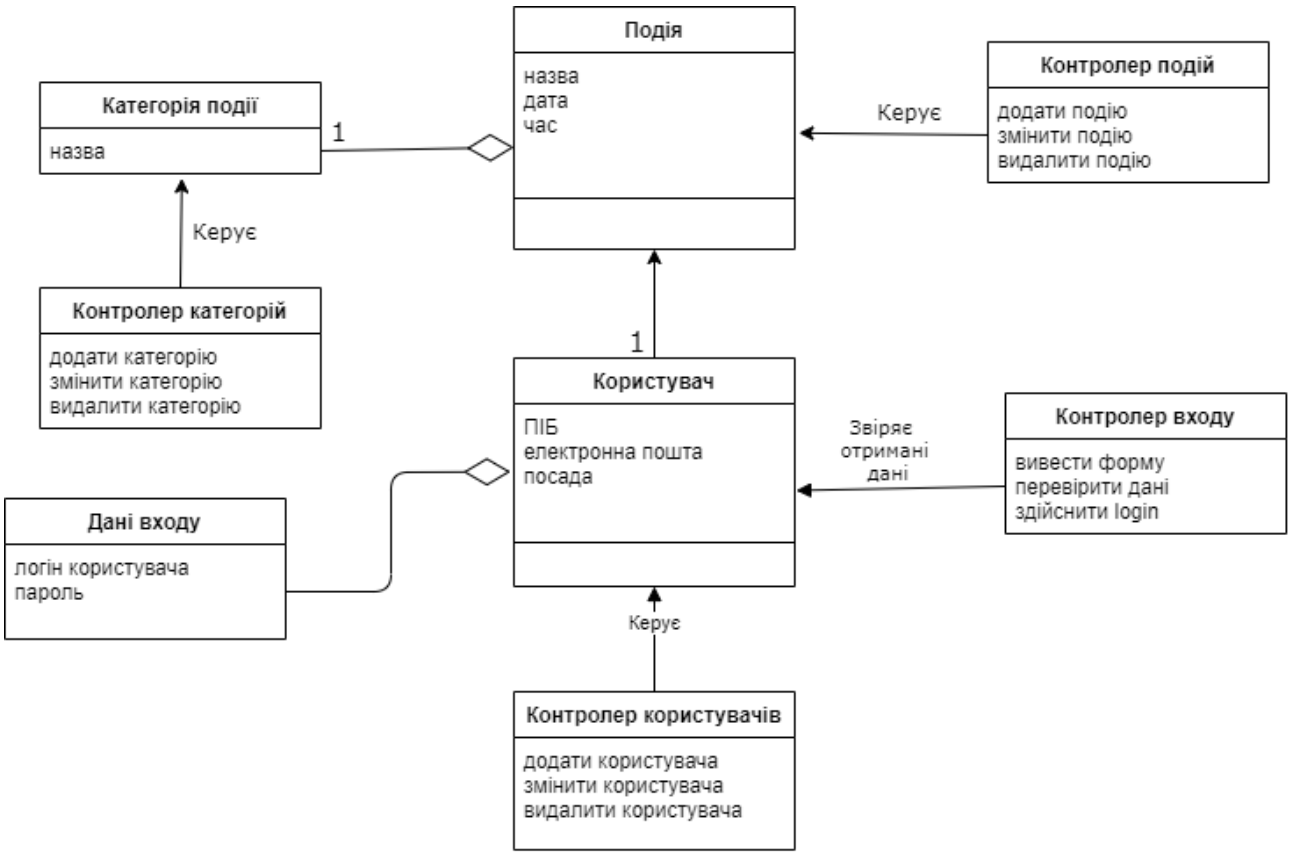


Рисунок - 3.1.1. UML-діаграма класів

3.1.2 Логічне проектування структур даних

UML-діаграма концептуальних класів була перетворена в опис структур даних з використанням моделі, яка була обрана в концептуальному описі архітектури ПП,

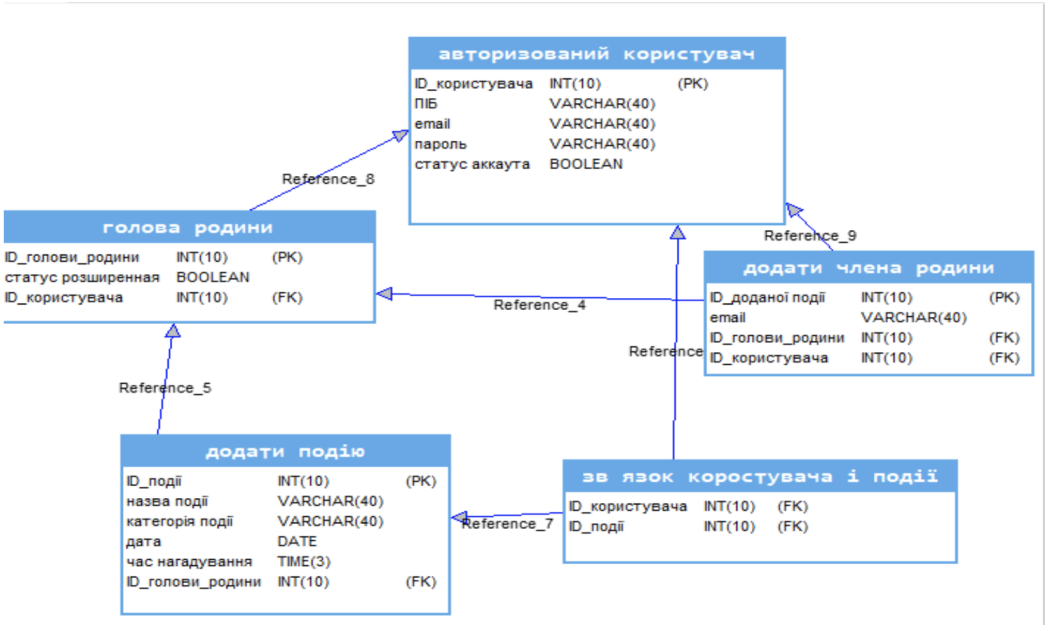


Рисунок – 3.1.2. Схема БД

3.2 Проектування програмних класів

1. На основі UML-діаграми концептуальних класів були спроектовані програмні класи:

– англійські або транслітерацію україномовних назви класів та їх атрибутів;

– абстрактні класи, їх класи-нащадки та інші класи;

– зв`язки між класами (наслідування, іменована асоціація, агрегатна асоціація, або агрегація, композитна асоціація або композиція) та їх кратності;

– атрибути класів с типами даних (цілий, дійсний, логічний, перелічуваний, символьний з урахуванням розміру), та типом видимості (публічний, захищений, приватний);

− методи-конструктори ініціалізації екземплярів об`єктів класу, set-методи та get-методи для доступу до атрибутів класу.

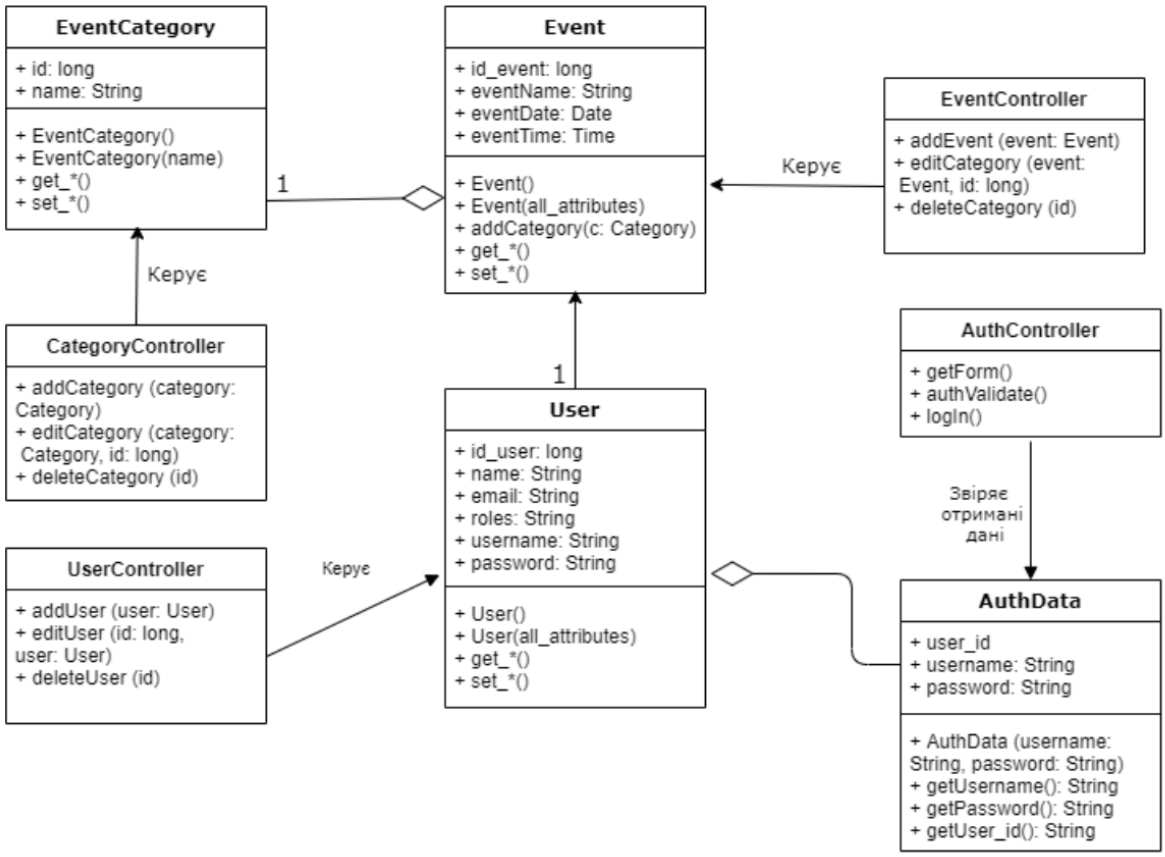


Рисунок – 3.2. UML-діаграма програмних класів

3.3 Проектування алгоритмів роботи методів програмних класів

Процес розробки алгоритмів:

1. З усіх методів виділили ті, що мають операції доступу до БД або містять керуючи умови (if-then-else, while).

2. Опишіть алгоритм роботи у вигляді UML-діаграми активності:

− кожний опис алгоритму представили в текстовому файлі на мові PlantUML, використовуючи веб-редактор.

@startuml

```
start
repeat
repeat
:Создание запроса;
repeat while(Запрос не подтвержден)
:Проверка email;
backward:Информация пользователя об ошибке;
repeat while(Ошибка данных!)
:Передача от пользователя параметров аутентификации в БД;
:Сохранение параметров аутентификации в БД;
note right
CREATE user WITH ENCRYPTED PASSWORD
GRANT CREATE ON events_db TO user
CREATE TABLE user(
id_user BEGINT PRIMARY KEY;
username VARCHAR();
name VARCHAR)
end note
:Информирование пользователя об успешной регистрации;
stop
@enduml
```

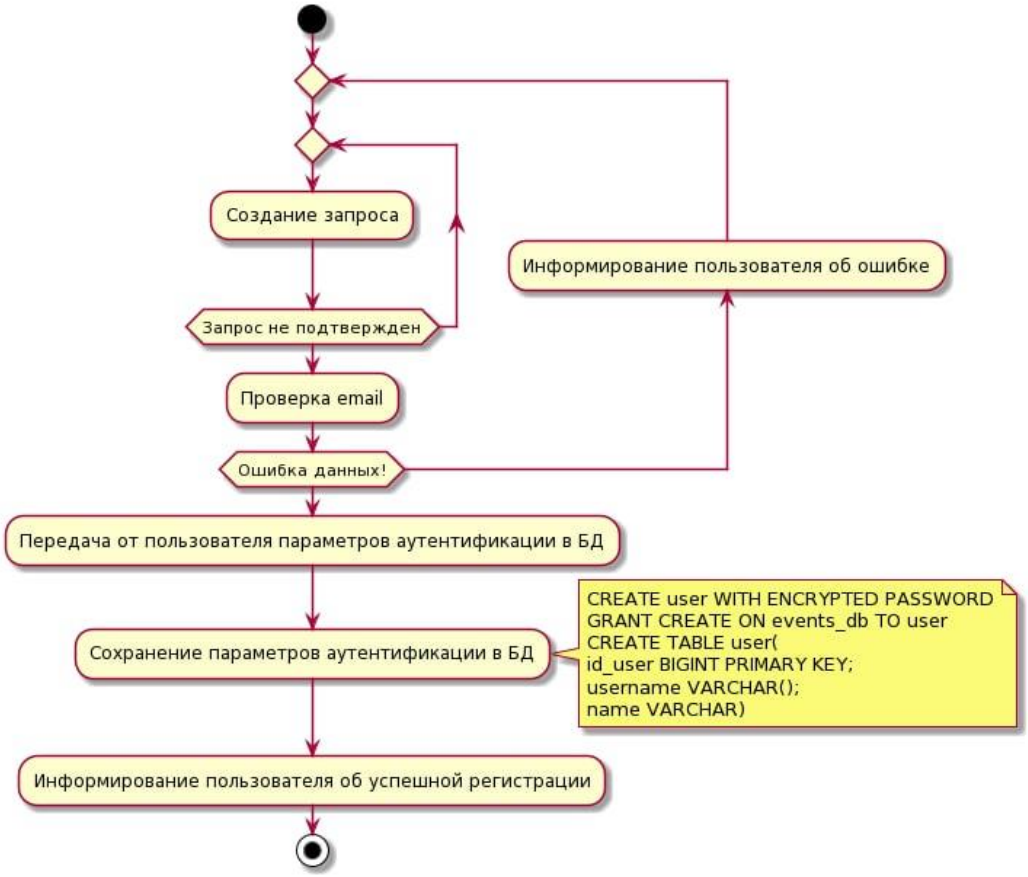


Рисунок – 3.3.1. Алгоритм реєстрації користувача

3.4 Проектування тестових наборів методів програмних класів

Назва функції	№ тесту	Вхідні дані	Очікуваний результат тесту
User.registerUser (name, email, password, checkpassword, role)	1	Коректний список вхідних аргументів	Збереження користувача до БД
	2	name:<String>, len >100	Помилка вхідних даних
	3	email:<String>, len >100 email !=(a-Z)(1 upper), 0-9(1), спецсимволы любые, 8-16	Помилка вхідних даних
	4	password:<String>, len > 80	Помилка вхідних даних
	5	Checkpassword: <String>, Checkpassword!= password	Помилка вхідних даних

4 Конструювання програмного продукту

4. 1 Особливості конструювання структур даних

4.1.1 Особливості інсталяції та роботи з СУБД

Розглянуто створення бази даних для роботи веб-системи подій з використанням СУБД PostgreSQL.

Для тестування програмних модулів використовується СУБД PostgreSQL версії 13.0. Для роботи веб-сервісу використана хмарна СУБД Firebase.
Для підключення Firebase у проект встановлено Firebase JavaScript SDK командою `npm init`. Оскільки Firebase розроблений Google, передбачається наявність Google-акаунтів у розробників програмного продукту для роботи з базою даних.

4.1.2 Особливості створення структур даних

До створеної бази даних були створені тригери та функції для маніпулювання даними, занесена демонстраційна інформація. Також були розроблені запити для демонстрації можливостей створеної бази даних. Архітектура бази даних передбачає підключення її до комп'ютерного додатку чи веб-сайту через використання користувачів. Така система дозволяє обмежувати привілеї для захисту даних.

База даних складається з наступних таблиць: Події, Користувачі.

Таблиця «Події» містить інформацію про події. У ній присутні такі поля, як унікальний ідентифікатор події, назва події, її дата та час нагадування.

Таблиця «Користувачі» містить інформацію про усіх користувачів системи. У ній присутні такі поля, як унікальний ідентифікатор анкети (первинний ключ), ім'я, пароль та ім'я користувача (пошта) для входу у систему.

Розглянемо SQL-запити для створення таблиць:

1. Таблиця "Події"

```
CREATE TABLE events(
id_event BIGINT PRIMARY KEY,
```

```
name VARCHAR,
date DATE,
time TIME,
category VARCHAR)
```

2. Таблиця “Користувачі”

```
CREATE TABLE user(
id_user BIGINT PRIMARY KEY,
email VARCHAR,
name VARCHAR)
```

Для зберігання первинного ключа події використовується тип даних BIGINT, бо операції порівняння та пошуку швидше за все працюють з цифровими типами.

4.2 Особливості конструювання структур даних

Для реалізації структур, описаних у розділі вище, було обрано створення веб-застосунку, де і буде реалізовано структури даних за допомогою програмних класів.

4.2.1 Особливості роботи з інтегрованим середовищем розробки

Інформаційна система «Менеджер подій» представляє собою клієнт-серверне застосування.

Розробка системи виконується в IDE JetBrains WebStorm. WebStorm знаходить застосування всіх можливостей сучасної JavaScript-екосистеми. Включає в себе розумне автодоповнення коду, перевірку помилок на льоту, швидку навігацію по коду і рефакторинг для JavaScript, TypeScript, мов стилів, а також для популярних фреймворків.

4.2.2 Особливості створення програмної структури з урахуванням спеціалізованого Фреймворку

Для реалізації системи була обрана мова об’єктно-орієнтованого програмування JavaScript із використанням платформи Node.js, фреймворку React та бібліотеки Bootstrap.

Bootstrap під’єднано до проекту наступним кодом:

```
      <link                                                    rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
      <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>
```

4.2.3 Особливості створення програмних класів

Наведемо як приклад клас створення сторінки календаря на сайті. Використовуємо фреймворк React.

```
import React, { Component } from 'react';
import './calendar-page.scss';
import NavigationSidebar from "../../components/navigation-sidebar";
import TasksList from "../../components/tasks-list";
import SearchPanel from "../../components/search-panel";
import CalendarComponent from "../../components/calendar";
import CategoriesComponent from "../../components/categories";
import Modal from 'react-modal';
import AddModal from "../../components/add-modal";
Modal.setAppElement('#root');
class CalendarPage extends Component {
constructor() {
super();
this.state = {
items: [],
term: '',
date: new Date(),
day: '',
activeCategory:'',
addModalIsOpen: false,
calendarDate: ''
};
this.toggleAddModal = this.toggleAddModal.bind(this);
}
componentDidMount() {
this.setState({
items: [
{
title: "Закончить наконец-то курсовую",
time: "09:00",
category: "events",
date: new Date(1589403600000)
}
```

4.2.4 Особливості розробки алгоритмів методів програмних класів або процедур/функцій

У програмному середовищі були розроблені алгоритми методів програмних класів. Як приклад, наведемо алгоритм методу сортування за категорією

```
class CategoriesComponent extends Component {
  constructor() {
    super();
    this.state = {
        activeCategory: ''
    };
     this.onCategoryChange = this.onCategoryChange.bind(this);
  }
  onCategoryChange(value){
    if (this.state.activeCategory !== value) {
      this.setState({activeCategory : value});
      this.props.onCategoryChange(value);
    } else {
      this.setState({activeCategory : ''});
      this.props.onCategoryChange('');
    }
  }
  render() {
    return (
        <div className="categories">
```

```
        <h6 className="category-title">Категории</h6>
        <div className="categories-wrapper" >
         <button className={this.state.activeCategory==='events' ? "activeCategory category-btn purple-btn" : "category-btn purple-btn"}
                 onClick={() => this.onCategoryChange('events')}
>Мероприятия</button>
         <button className={this.state.activeCategory==="reminders" ? "activeCategory category-btn orange-btn" : "category-btn orange-btn"}
                onClick={() => this.onCategoryChange('reminders')}
>Напоминания</button>
         <button className={this.state.activeCategory==="birthdays" ? "activeCategory category-btn pink-btn" : "category-btn pink-btn"}
                 onClick={() => this.onCategoryChange('birthdays')}>Дни рождения</button>
         <button className={this.state.activeCategory==="holidays" ? "activeCategory category-btn blue-btn" : "category-btn blue-btn"}
                 onClick={() => this.onCategoryChange('holidays')}>Праздники</button>
        </div>
       </div>
    );
  }
}
export default CategoriesComponent;
```

4.2.5 Особливості використання спеціалізованих бібліотек та API

Для спрощення розробки програмного продукту було використано готове API календаря – React Calendar. Оскільки API має відкритий вихідний код, .календар було кастомізовано та налагоджено відповідно до потреб програмного продукту.

3. Тестування програмних модулів

4.3.1 Тестування методу loginUser()

User.loginUser **()** являє собою метод, що перевіряє введені користувачем під час авторизації та проводить їхню валідацію. Метод виглядає наступним образом:

```
$(document).ready(function() {
  $('#third_form').submit(function(e) {
    e.preventDefault();
    var email = $('#email').val();
    var password = $('#password').val();
    var confirmPassword = $("#txtConfirmPassword").val();

    $(".error").remove();

    if (email.length< 1) {
      $('#email').after('<span class="error">Заполните это поле</span>');
    } else {
      var regEx = /^[A-Z0-9][A-Z0-9._%+-]{0,63}@(?:[A-Z0-9-
]{1,63}.){1,125}[A-Z]{2,63}$/;
      var validEmail = regEx.test(email);
      if (!validEmail) {
        $('#email').after('<span class="error">Введите зарегистрированный
email</span>');
      }
    }
 if (password != confirmPassword)
        $("#divCheckPasswordMatch").html("Пароль неверный !");
    else
        $("#divCheckPasswordMatch").html("Пароли совпадают.");
    }
  });
});
```

№	Тестові набори	Очікувані результати
1	email = lizazamorina2001@gmail.com, password = 2014Ov08	Вхід до системи
2	email = testemail@gmail.com, password = testpass	Помилка: Користувач не знайдений!

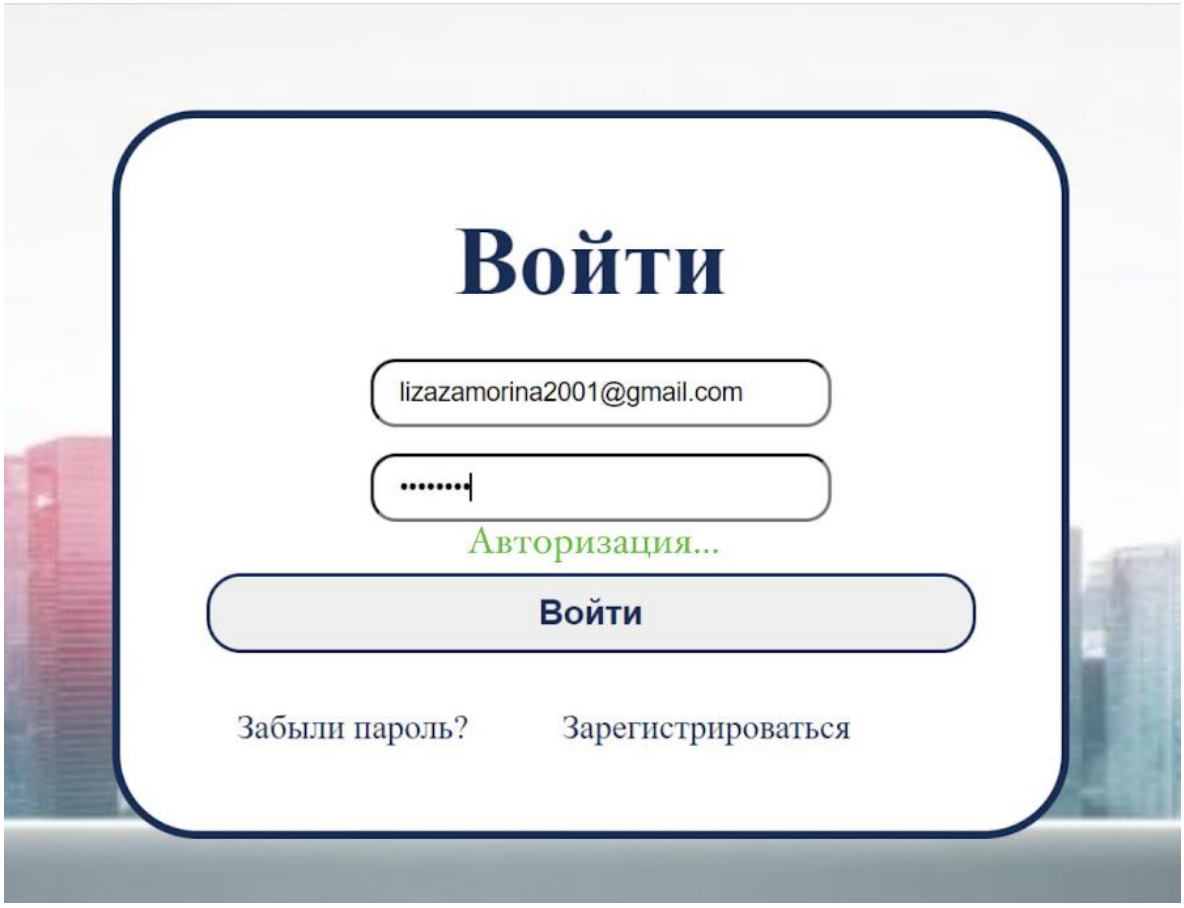


Рис. 4.3.2.1 – Процес авторизації користувача

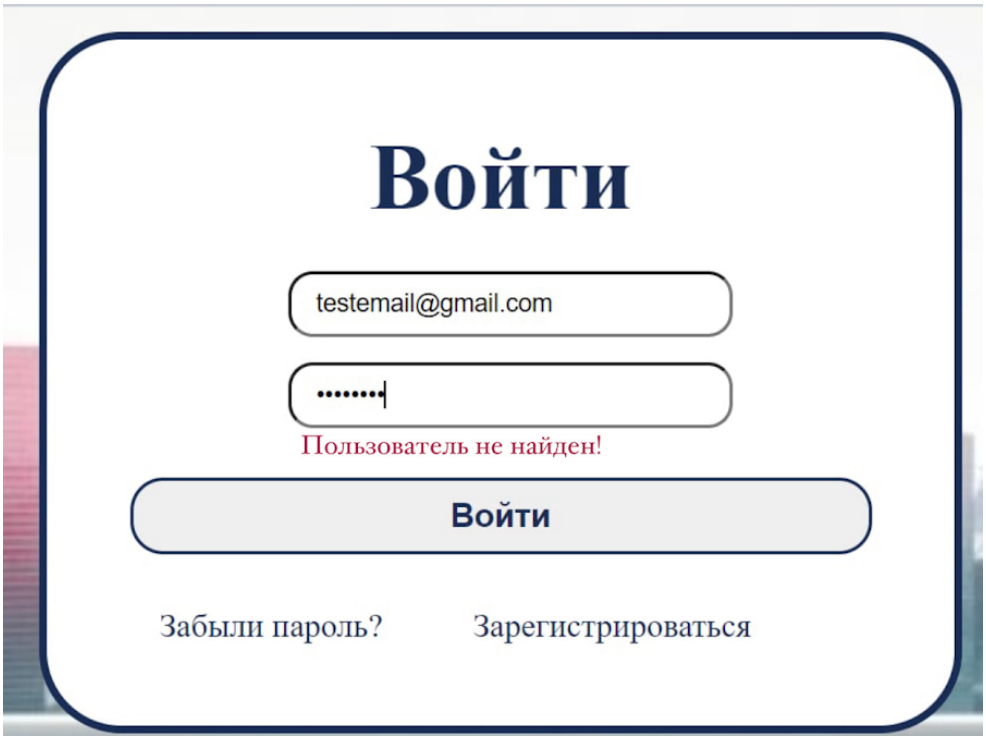


Рис. 4.3.2.2 - Помилка: користувач не знайдений

5 Розгортання та валідація програмного продукту пояснювальної записки курсової роботи

5. 1 Інструкція з встановлення програмного продукту

Розроблене програмне забезпечення підтримується усіма веб-браузерами, усіма версіями як на ОС Windows, Mac ОС , так и на Linux ОС.

Здійснювати дії на веб-сервісі та користуватися ним користувач може за допомогою маніпулятора «миша» та клавіатури. За допомогою маніпулятора «миша» користувач може нажати на кнопку/текст, а за допомогою клавіатури – вводити дані у різні поля та форми.

5.2 Інструкція з використання програмного продукту

На рис. 5.2.1 зображена головна сторінка сайту, з якої можна здійснити будь-яку дію – вхід або перегляд нагадувань.

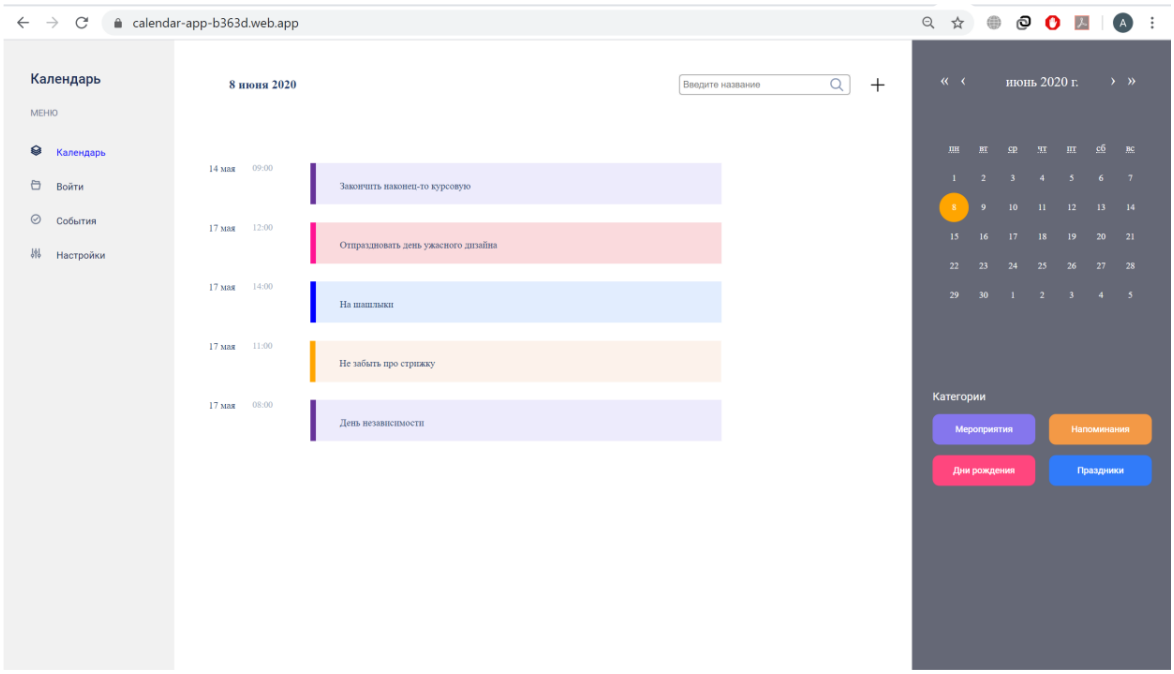


Рис. 5.2.1 – Головна сторінка

На сторінці авторизації можна здійснити вхід, або, за необхідності, реєстрацію або відновлення паролю.

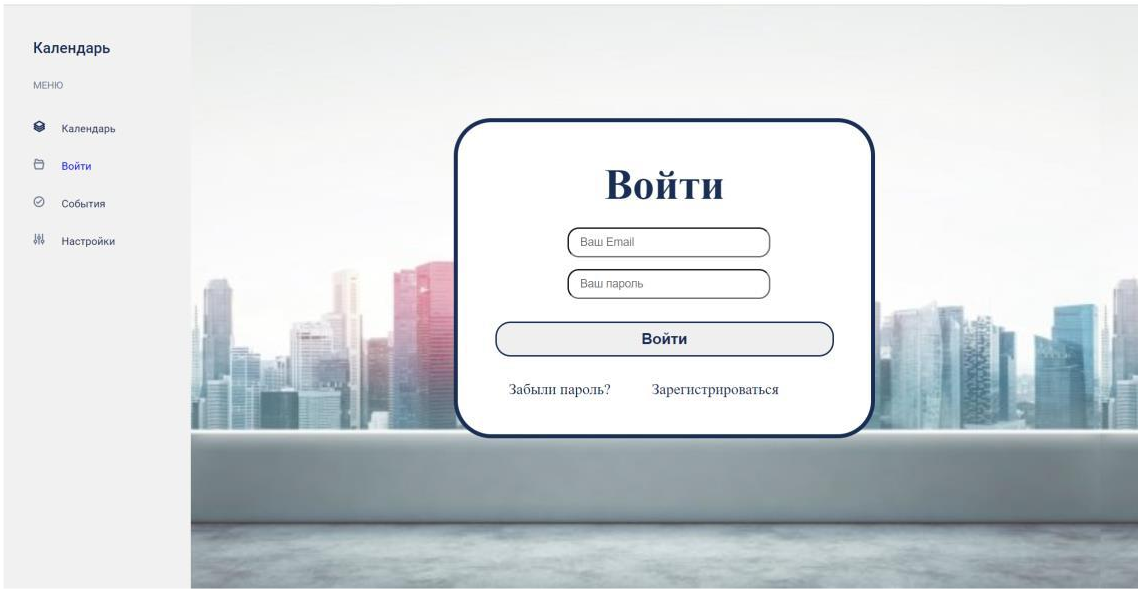


Рис. 5.2.2 – Сторінка авторизації

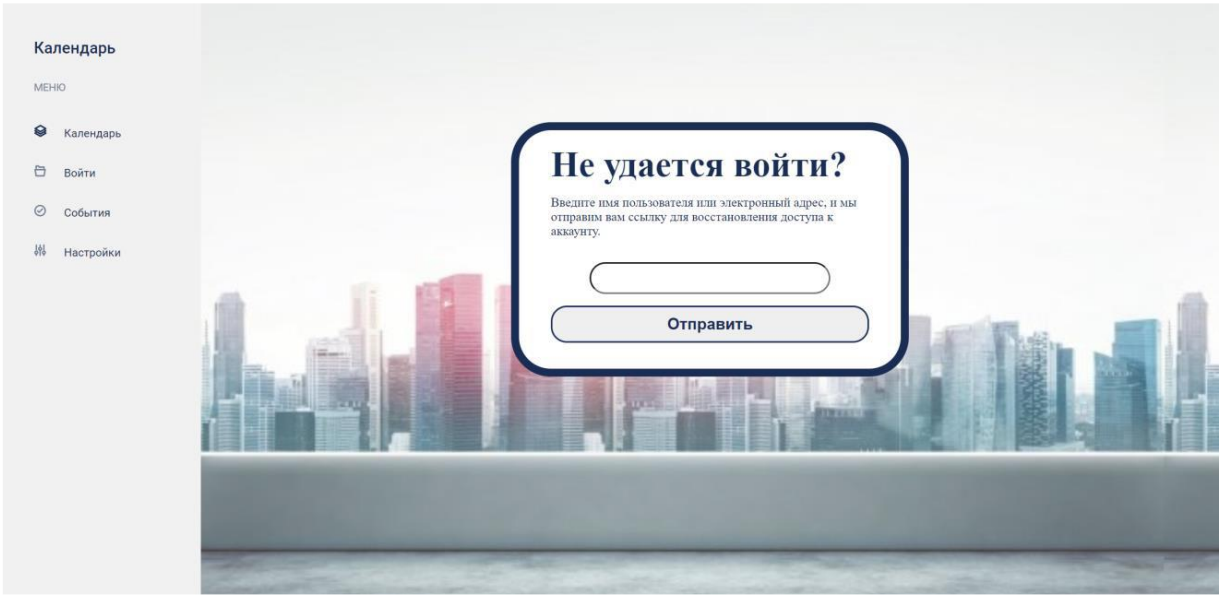


Рис. 5.2.3.- Сторінка відновлення паролю

Зі сторінки реєстрації також можливо перейти назад до сторінки входу.

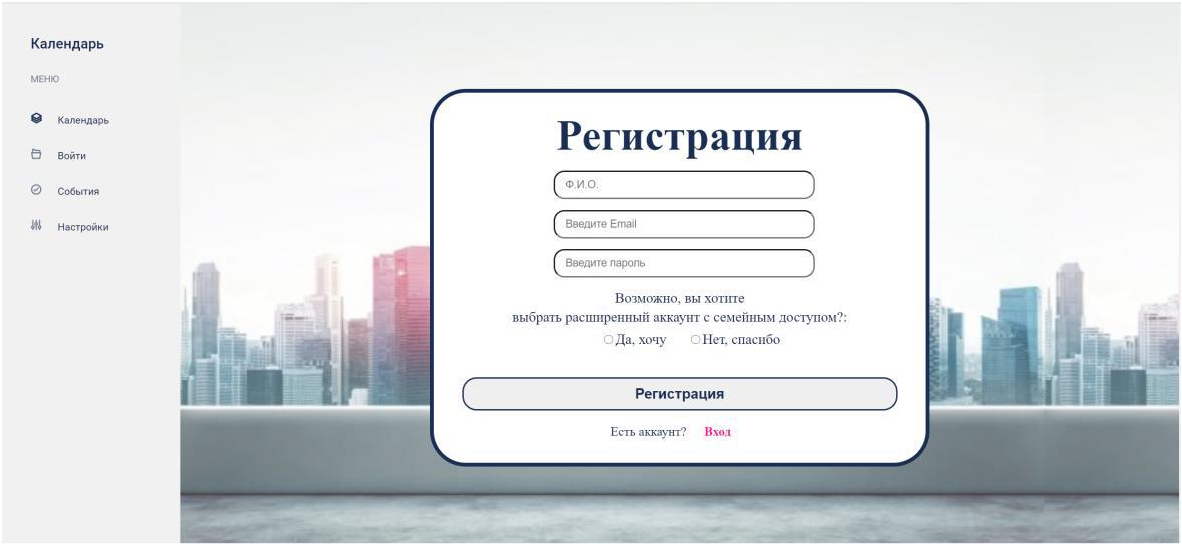


Рис. 5.2.4 – Сторінка реєстрації

На головній сторінці міститься список всіх нагадувань. Для швидкого пошуку за ними передбачено поле «Пошук» у верхньому правому кутку сторінки.

Для вибору нагадувань у конкретний день цей день потрібно вибрати на календарі у правому кутку сторінки. Сьогоднішній день підсвічується жовтим кольором, обраний – синім кольором.

Спробуємо переглянути усі події:

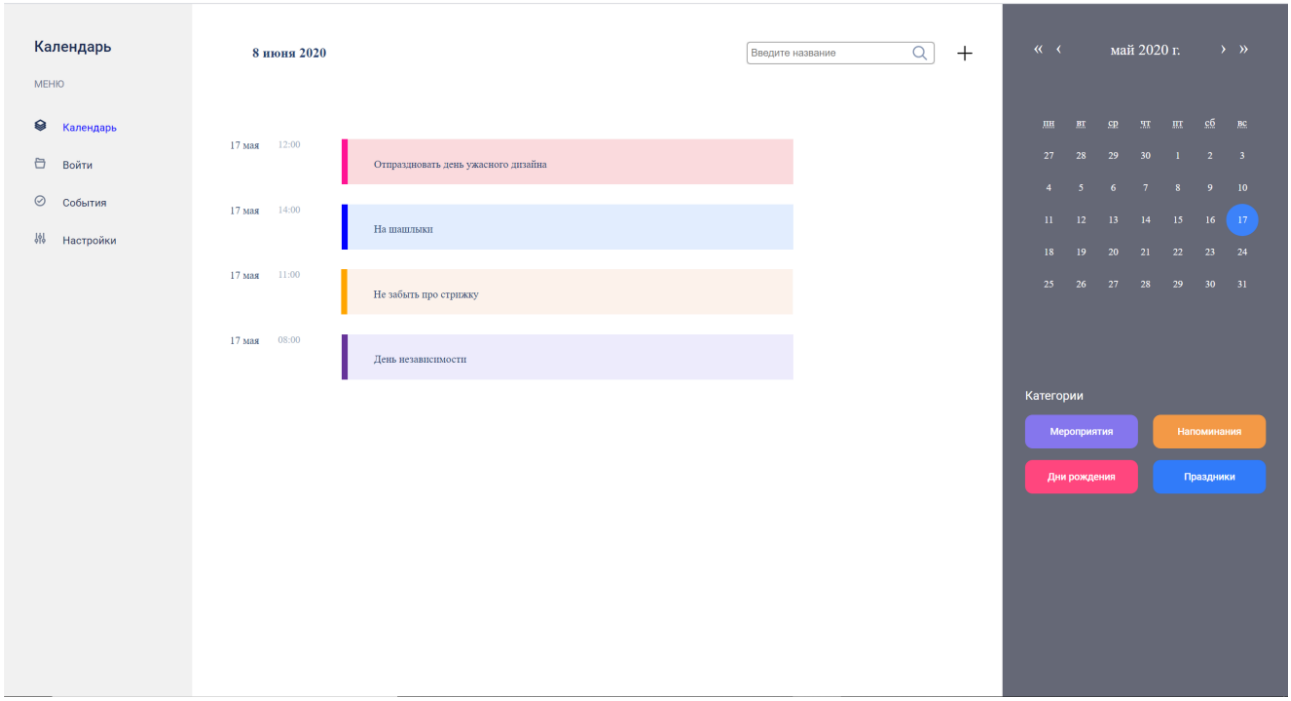


Рис. 5.2.5 – Список подій

Для того, шоб обрати події тільки за одною категорією («Заходи», «Нагадування», «Дня народження», «Свята»), потрібно натиснути відповідну

кнопку в розділі «Категорії» під календарем. Спробуємо обрати «Дні народження»:

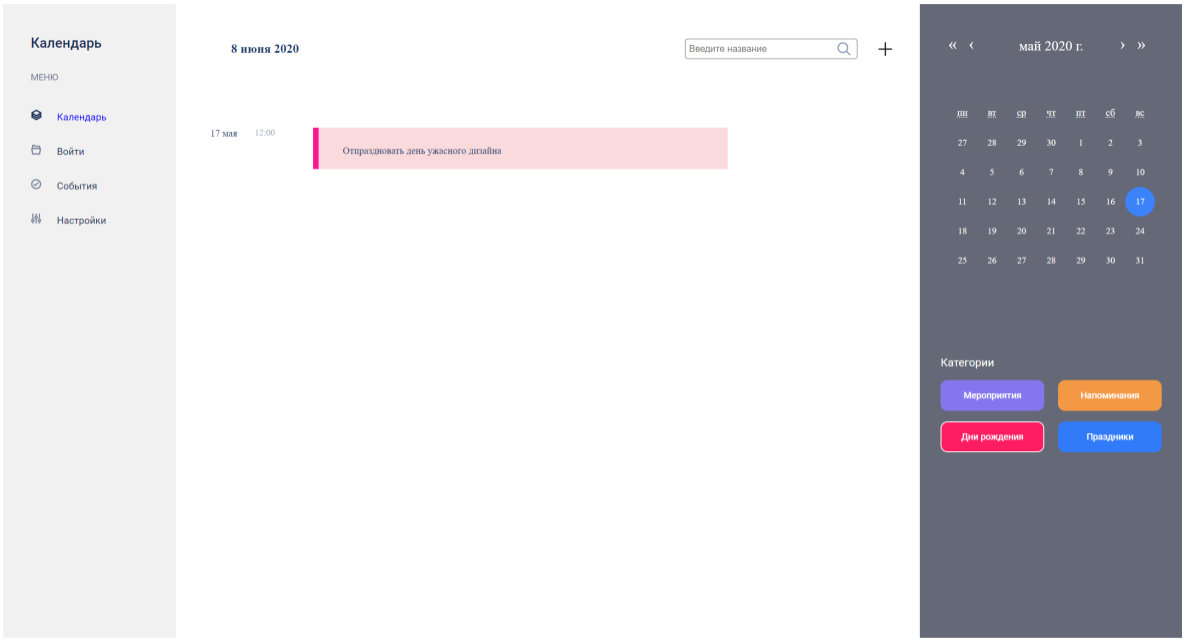


Рис. 5.2.6 – Дні народження

Для додавання власної події потрібно натиснути знак «+», заповнити необхідні поля та натиснути «Додати»:

Рис. 5.2.7 – Додавання події

5.3 Результати валідації програмного продукту

Метою програмного продукту було створення календаря подій, який спростив би планування подій за допомогою нагадувань.

В даний період часу проект знаходиться на ранній стадії свого розвитку, але внаслідок буде розвинений більше. У розробленому ПП можливо додавати власні події, але поки не реалізовано поділ подіями та груповий перегляд.

Розробка подібних інформаційних систем може значно допогти людям у плануванні свого повсякденного життя. Вона повинна вплинути на збільшення розпланованого часу у користувачів та структурувати їх повсякденні діла.

Висновки

В результаті створення програмного продукту була досягнута наступна мета його споживача: «Створення календаря подій, який спростив би планування подій за допомогою нагадувань».

Програмний продукт «Менеджер подій» задовольняє такі потреби споживача:

1) Додавання події;
2) Перегляд всіх доданих подій;
3) Категоризування подій (перегляд за категоріями);
4) Редагування подій.

В процесі створення програмного продукту виникли такі труднощі:

1) деякі організаційні труднощі роботи у команді;
2) брак часу;
3) відсутність досвіду у front-end-розробці.

Через вищеописані труднощі, а також через обмежений час на створення програмного продукту, залишилися нереалізованими такі прецеденти або їх окремі кроки роботи:

1) поділ подями;
2) груповий перегляд подій.

Зазначені недоробки планується реалізувати в майбутніх курсових роботах з урахуванням тем дисциплін наступних семестрів.