1. Вступ

Цей документ описує систему **Codebasky** – хмарну платформу для керування задачами та співпраці команди в режимі реального часу. У Codebasky користувачі можуть створювати проєкти, заводити задачі, призначати виконавців, обговорювати деталі в коментарях, прикріплювати файли та отримувати нотифікації про зміни. Система також дає менеджменту базову аналітику продуктивності (наприклад, скільки задач створено, скільки виконано, які зависли).

Проблема, яку вирішує **Codebasky**: у більшості команд робочий контекст розмазаний між різними інструментами – таск-трекер, чат, пошта, гугл-доки, випадкові файли в месенджері. Це ускладнює прозорість, відповідальність і дедлайни. **Codebasky** надає єдиний робочий простір (workspace), де вся ця взаємодія відбувається в одному місці.

Документ також служить основою для проєктування безпеки (керування доступом, права ролей, політики зберігання даних), стійкості до відмов та планування деплойменту платформи у хмарне середовище.

2. Високорівнева архітектура

Codebasky складається з:

- фронтенду (React + TypeScript, SPA),
- бекенду (набір сервісів у стилі SOA),
- інфраструктурних компонентів (база даних, кеш, сховище файлів).

Архітектурна ідея:

- не один моноліт, а кілька окремих сервісів за бізнес-областями;
- всередині кожного сервісу – чітке розділення бізнес-логіки і інфраструктури за принципами Clean Architecture;
- події та повідомлення використовуються там, де не потрібно чекати відповіді одразу (нотифікації, аналітика).

2.1 Бекенд

2.1.1 Сервіси

У бекенді є кілька основних сервісів:

- Identity & Access Service
    - логін, реєстрація, токени доступу(access/refresh);
    - зберігає інформацію про ролі (менеджер, учасник команди, гість) і права доступу до проєктів та задач.
- Project & Task Service
    - створення проєктів;
    - створення, редагування, статуси задач;
    - призначення виконавця, дедлайну, пріоритету;
    - історія змін задач (аудит хто що поміняв).
- Collaboration Service
    - коментарі до задач;
    - згадки користувачів (@user);
    - індикатори активності в реальному часі (хтось зараз дивиться чи редагує задачу).
- Attachment Service
    - завантаження файлів, що стосуються задачі;
    - зберігання метаданих про файл;
    - доступ через тимчасові підписані URL, а не напряму.

2.1.2 Основні шари Clean Architecture

1. **Domain Layer (Доменний шар)**
   1.1. Містить сутності (Task, Project, Comment, NotificationRule тощо), бізнес-правила і політики.
   1.2. Не залежить від інфраструктури (бази даних, мережі, файлового сховища).
   1.3. Визначає інтерфейси (контракти) для збереження та читання даних.

2. **Application Layer (Прикладний шар / Use Cases)**
   2.1. Описує сценарії використання системи: створити задачу, призначити виконавця, додати коментар, надіслати нагадування про дедлайн.
   2.2. Координує роботу доменних об’єктів і викликає інтерфейси доступу до зовнішніх ресурсів.

2.3. Не містить деталей інфраструктури, тільки бізнес-процеси.

3. **Infrastructure Layer (Інфраструктурний шар)**
   3.1. Реалізує доступ до бази даних, кешу, файлового сховища, черги подій, e-mail провайдерів.
   3.2. Надає конкретні імплементації інтерфейсів, які описані у Domain/Application.
   3.3. Цей шар залежить від внутрішніх шарів, а не навпаки (інверсія залежностей).

4. **Web Layer (Транспортний шар сервісу)**
   4.1. Приймає зовнішні запити (HTTP API / WebSocket).
   4.2. Перевіряє вхідні дані, викликає відповідний use case з Application Layer, формує відповідь клієнту.
   4.3. Не містить бізнес-логіки, тільки адаптацію запит / відповідь.

2.2 Фронтенд

Фронтенд **Codebasky** є односторінковим вебзастосунком (Single Page Application, SPA) з компонентною архітектурою на основі React + TypeScript.

Основні елементи:

1. **App**
   Головний компонент застосунку. Відповідає за ініціалізацію клієнтського стану, підключення користувача до свого робочого простору та конфігурацію маршрутизації.

2. **Pages**
   Вищого рівня екрани, які відображають основні області продукту (наприклад, список проєктів, дошка задач, деталі задачі, аналітична панель). Кожна сторінка збирає дані з бекенду і передає їх у дочірні компоненти для відмалювання.
3. **Components**
   Багаторазові UI-елементи (картка задачі, список коментарів, модальні вікна, форми зміни статусу задачі, контрол для прикріплення файлу). Вони не знають про глобальний стан застосунку і отримують усе необхідне через пропси.
   Router (React Router)
   Забезпечує навігацію між сторінками без повного перезавантаження браузера. Відповідає за формування URL-структури (проєкт, задача, аналітика) і рендер відповідної сторінки.
4. **Global State / Context**
   Шар керування глобальним станом (наприклад, Context API / Redux-подібний стор). Зберігає поточного користувача, його роль і дозволи, обраний проєкт/воркспейс, а також кеш основних даних (наприклад, перелік задач). Це дозволяє уникати дублювання запитів та узгоджено показувати права доступу у всіх компонентах.
5. **Services**
   Клієнтські сервіси, через які фронтенд спілкується з бекендом.
   - HTTP/REST клієнт: отримання й оновлення задач, проєктів, коментарів, вкладень.
   - WebSocket клієнт: підписка на події в реальному часі (оновлення задачі, новий коментар, зміна виконавця) без ручного перезавантаження сторінки. Таким чином фронтенд отримує як запит-відповідь дані, так і push-оновлення.

3. Функціональні вимоги

3.1 Каталог функціональних вимог

Нижче наведено таблицю функціональних вимог до системи.

ID	Назва	Опис
Керування користувачами та доступом		
FR01	Реєстрація та вхід користувача	Користувач може створити обліковий запис, увійти та вийти. Після входу отримує токен доступу. Доступ до інших функцій можливий лише після автентифікації.
FR02	Робочі простори	Система підтримує кілька робочих просторів (workspaces). Кожен простір має власні проєкти, задачі, учасників. Менеджер простору може запрошувати інших користувачів.
FR03	Ролі та права доступу	Ролі (наприклад, менеджер, учасник команди, гість) визначають дозволи користувача: створення проєктів, редагування задач, зміна статусів або тільки перегляд.
Управління проєктами і задачами		
FR04	Створення та редагування проєкту	Користувач з відповідними правами може створити новий проєкт у робочому просторі, змінити його назву/опис або заархівувати проєкт.
FR05	Створення задачі	У межах проєкту можна створити задачу з полями: заголовок, опис, статус, пріоритет, дедлайн, виконавець. Задачу можна редагувати або видаляти (залежно від прав).

FR06	Статуси задачі	Можна змінювати статус задачі (напр. “To Do / In Progress / Done”). Переміщення задачі по статусах відображається на дошці проєкту для всієї команди.
FR07	Призначення відповідальних і дедлайнів	До задачі можна призначити відповідального користувача та дедлайн. Зміни у виконавці або даті завершення мають бути видимі іншим учасникам проєкту.
FR08	Історія змін задачі	Система зберігає хто і що змінював у задачі (статус, дедлайн, опис, виконавця). Ця історія доступна для перегляду, щоб відстежити послідовність правок.
Співпраця та обмін інформацією		
FR09	Коментарі до задачі	Користувачі можуть залишати коментарі, ставити питання, давати уточнення. Підтримується згадка іншого користувача через @username.
FR10	Оновлення в реальному часі	Зміни задачі (оновлення статусу, новий коментар, зміна виконавця) доставляються іншим учасникам майже миттєво без перезавантаження сторінки.
FR11	Вкладення файлів	До задачі можна додати вкладення (зображення, документ тощо). Система зберігає файл та показує його як частину задачі. Доступ до вкладень обмежується тими ж правами, що й до задачі.
Сповіщення та аналітика		
FR12	Сповіщення про зміни	Користувач отримує сповіщення, якщо його призначили відповідальним, якщо його згадали в коментарі або якщо наближається дедлайн. Сповіщення можуть відображатись у застосунку і (опційно) надсилатись email.

FR13	Огляд прогресу команди	Користувач із роллю менеджера може переглядати зведену інформацію: скільки задач відкрито, скільки закрито за період, які задачі прострочені.
Пошук і навігація		
FR14	Пошук і фільтр задач	Можна шукати та фільтрувати задачі за статусом, виконавцем, дедлайном, пріоритетом або текстом у назві/описі, щоб швидко знайти потрібні задачі у великому проєкті.

3.2 Актори

Актор – це суб’єкт, що взаємодіє із системою Codebasky. Нижче наведено перелік основних акторів і їхні ролі в системі.

3.2.1 Менеджер команди

Менеджер команди має розширені права в межах робочого простору (workspace). Його роль – організовувати роботу інших учасників і контролювати виконання задач. Основні обов’язки менеджера:

- створення та налаштування робочого простору (додавання/видалення учасників);
- створення і редагування проєктів;
- створення задач і призначення відповідальних;
- встановлення дедлайнів і пріоритетів;
- зміна статусів задач;
- перегляд і аналіз прогресу команди (прострочені задачі, виконані задачі, навантаження);
- отримання нотифікацій про критичні оновлення (наприклад, пропущені дедлайни).

Менеджер відповідає за координацію роботи команди та прозорість стану задач у проєкті.

3.2.2 Учасник команди

Учасник команди виконує роботу в межах проєктів і задач, до яких йому надано доступ. Основні обов’язки учасника:

- перегляд призначених йому задач;
- оновлення стану задачі;
- участь в обговореннях через коментарі;
- прикріплення файлів/доказів виконання;
- реагування на дедлайни і сповіщення;
- згадування інших учасників через @username для уточнення деталей.

Учасник команди не обов’язково може змінювати чужих виконавців або редагувати весь проєкт — його права визначаються роллю, яку йому надав менеджер.

3.2.3 Гість

Гість має доступ лише до вибраних проєктів або окремих задач, які йому відкрив менеджер. Роль гостя зазвичай використовується для зовнішніх людей (клієнт, стейкхолдер, підрядник), яким треба бачити статус роботи, але не втручатися в сам процес.

Основні можливості гостя:

- перегляд задач (назва, опис, статус, дедлайн);
- перегляд коментарів і вкладень, якщо йому надали доступ;
- отримання сповіщень, у яких його згадали.

Гість зазвичай не може:

- створювати нові проєкти;
- перепризначати задачі;
- змінювати статуси задач;
- видаляти або редагувати чужі коментарі.

3.3 Випадки використання (Use Cases)

3.3.1 Логін / Реєстрація

Ініціатор: Будь-який користувач (Менеджер, Учасник команди, Гість).

Опис:
Актор входить у систему або створює новий обліковий запис, щоб отримати доступ до робочого простору (workspace) і пов’язаних проєктів та задач.

Передумови:

1. Користувач має інтернет-з’єднання.
2. У разі реєстрації — вказано e-mail.

Основний потік:

1. Користувач вводить e-mail і пароль.
2. Система перевіряє правильність даних.
3. У разі успіху система надає токен доступу та відкриває робочий простір користувача.

Альтернативні потоки:

1. А-1: Користувач уперше в системі, система створює новий аккаунт і додає користувача або в новий workspace, або в workspace за інвайт-посиланням.

Виняткові потоки:

1. Е-1: Введено неправильні дані (невірний пароль / нема такого користувача), тоді система показує помилку та пропонує повторити спробу.
2. Е-2: Обліковий запис деактивовано адміністратором робочого простору, тоді доступ заборонено.

3.3.2 Створення проєкту

Ініціатор: Менеджер команди.

Опис:
Дозволяє менеджеру створити новий проєкт у своєму робочому просторі, щоб організувати задачі команди.

Передумови:

1. Менеджер автентифікований у системі.
2. Менеджер має роль із правом створювати проєкти в цьому workspace.

Основний потік:

1. Менеджер натискає “Створити проєкт”.
2. Вводить назву та короткий опис проєкту.
3. (Необов’язково) додає учасників команди до проєкту.
   Підтверджує створення.
4. Система створює проєкт і показує його в списку доступних проєктів.

Альтернативні потоки:

1. А-1: Менеджер одразу додає попередньо визначені колонки статусів задач (наприклад “To Do / In Progress / Done”), замість дефолтних.

Виняткові потоки:

1. Е-1: У менеджера немає прав на створення проєкту в цьому workspace, тоді система відмовляє і показує повідомлення про недостатні права.

2. Е-2: Обов’язкові поля (наприклад назва проєкту) не заповнено, тоді система не дає підтвердити створення.

3.3.3 Створення задачі і призначення виконавця

Ініціатор: Менеджер команди або Учасник команди (якщо роль дозволяє створювати задачі).

Опис:
Дозволяє додати нову задачу в конкретний проєкт, одразу вказати відповідального та дедлайн.

Передумови:

1. Користувач автентифікований.
2. Користувач має доступ до вибраного проєкту.

Основний потік:

1. Користувач натискає “Створити задачу”.
2. Вказує заголовок задачі та опис.
3. Встановлює статус початкової стадії (наприклад “To Do”).
4. Призначає відповідального користувача (виконавця).
5. Встановлює дедлайн / пріоритет (за потреби).
6. Підтверджує створення.
7. Система додає задачу до проєкту та робить її видимою іншим учасникам.

Альтернативні потоки:

1. А-1: Задача створюється без призначеного виконавця (наприклад як беклог). Виконавець буде доданий пізніше.

Виняткові потоки:

1. E-1: Користувач намагається призначити виконавцем людину, якої немає в цьому проєкті, тоді система не дозволяє і показує помилку доступу.

2. E-2: Дедлайн у минулому, тоді система не приймає значення і просить вказати коректну дату.

3.3.4 Оновлення задачі та обговорення

Ініціатор: Учасник команди.

Опис:
Дозволяє учаснику команди оновити стан задачі (наприклад змінити статус на “In Progress” або “Done”), додати коментар і прикріпити файли, щоб дати контекст або результат роботи.

Передумови:

1. Користувач авторизований.
2. Користувач має право редагувати цю задачу (він виконавець задачі або має роль, що дозволяє змінювати статуси/залишати коментарі).

Основний потік:

1. Користувач відкриває задачу.
2. За потреби оновлює статус задачі (наприклад “In Progress” -> “Done”).
3. Додає коментар (уточнення, запитання, звіт про виконання).
4. Прикріплює файл (скріншот, документ тощо).
5. Система зберігає зміни та одразу робить їх доступними іншим учасникам проєкту.
6. Інші учасники бачать оновлення майже в реальному часі (без перезавантаження сторінки).

Альтернативні потоки:

1. A-1: Користувач залишає тільки коментар, не змінюючи статус.
2. A-2: Користувач змінює статус, але не додає коментар.

Виняткові потоки:

1. E-1: Роль користувача не дозволяє змінювати статус задачі, тоді система не застосовує зміну статусу, показує повідомлення про недостатні права.

2. E-2: Завантажений файл перевищує обмеження розміру або не дозволений тип, тоді система блокує файл і просить інший варіант.
3. E-3: Інший користувач встиг змінити задачу раніше (конфлікт версій), тоді система показує, що є новіша версія задачі, і пропонує перезавантажити дані.

3.3.5 Перегляд прогресу команди

Ініціатор: Менеджер команди.

Опис:
Менеджер переглядає зведену інформацію по проєкту (виконані задачі, прострочені задачі, навантаження на учасників), щоб оцінити стан роботи команди та визначити ризики.

Передумови:

1. Менеджер автентифікований у системі.
2. Менеджер має доступ до відповідного проєкту у своєму робочому просторі.

Основний потік:

1. Менеджер відкриває панель аналітики / огляду проєкту.
2. Система показує перелік відкритих задач і їхній поточний статус.
3. Система окремо позначає задачі з простроченим дедлайном.
4. Система показує базові метрики, наприклад:
    - скільки задач завершено за період;
    - скільки задач зараз “In Progress”;
    - хто має найбільше активних задач.
5. Менеджер переглядає деталі проблемних задач і вирішує, кому їх перепризначити або що ескалювати усно/на зустрічі.

Альтернативні потоки:

1. A-1: Менеджер фільтрує дані за конкретним виконавцем, щоб подивитися тільки задачі певної людини.
   A-2: Менеджер відфільтровує тільки прострочені задачі, щоб сфокусуватись на критичних ризиках.

Виняткові потоки:

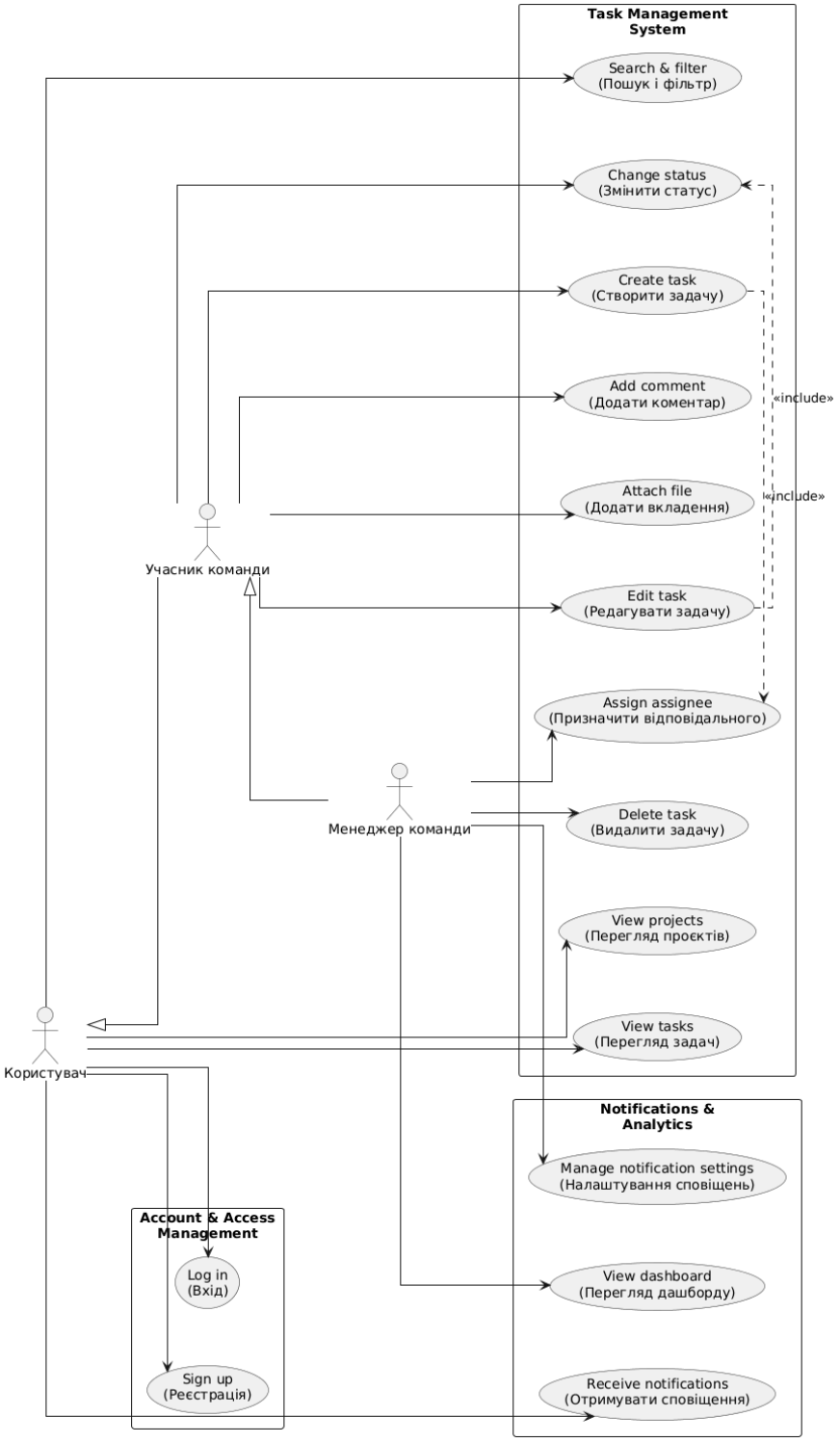
1. E-1: Менеджер не має прав доступу до цього проєкту, тоді система не показує метрики і виводить повідомлення про відмову в доступі.

2. E-2: У проєкті немає активних задач, тоді система показує порожній стан (“Немає відкритих задач”).

3.4 Use-case діаграми

3.4.1 UML діаграма

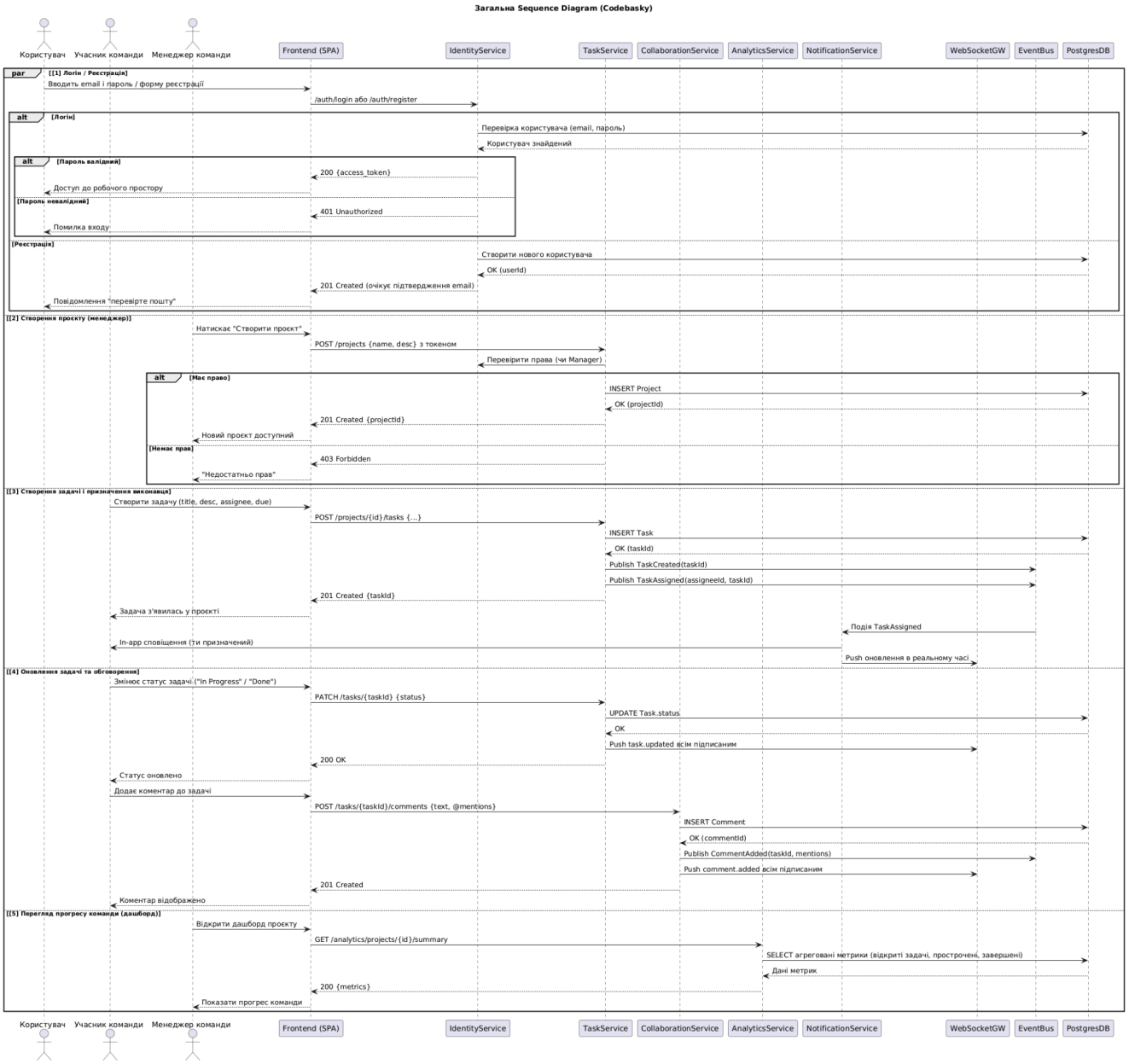
UML діаграма створена за допомогою платформи PlantUML, також за посиланням наведено вихідний код діаграми у репозиторії GitHub.



3.4.2 Загальна sequence діаграма

Sequence діаграма створена за допомогою платформи PlantUML, також за посиланням наведено вихідний код діаграми у репозиторії GitHub.

Діаграма відображає реалізацію основних use-case сценаріїв системи (пункти 3.3.1–3.3.5). Кожен блок відповідає окремому випадку використання.



3.4.3 Конкурентні потоки (граничні сценарії)

У межах граничних сценаріїв конкурентної взаємодії система Codebasky реалізує комбінацію шаблонів проєктування Active Object, Monitor Object та Producer–Consumer, що забезпечують узгоджене, безпечне та ефективне паралельне виконання операцій.

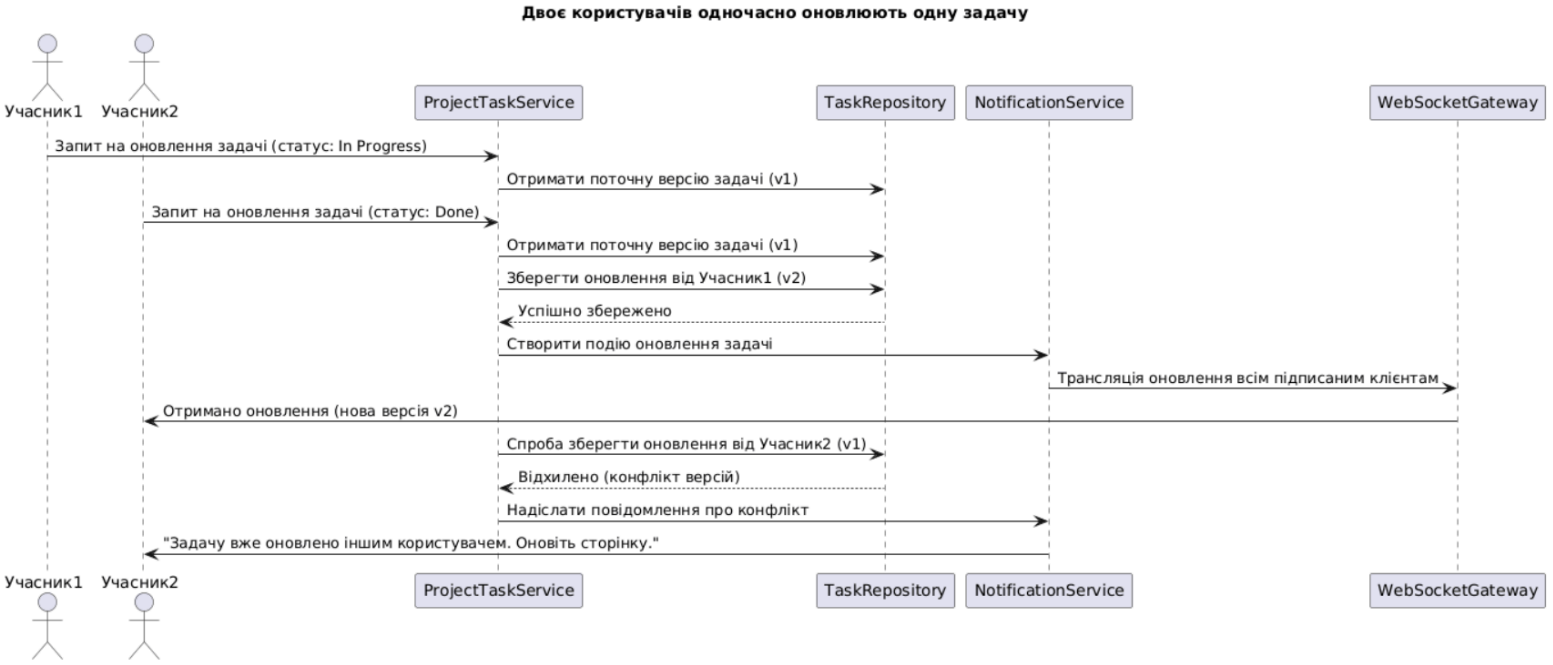
Патерн Active Object застосовується у сервісах ProjectTaskService та CollaborationService, які здійснюють асинхронну обробку подій, пов’язаних із створенням, оновленням і коментуванням задач. Це дозволяє відокремити ініціацію запиту від його виконання та підвищити стійкість системи до пікових навантажень.

Патерн Monitor Object реалізовано у компонентах IdentityAccessService та TaskRepository, що гарантують синхронізований доступ до спільних ресурсів — облікових записів користувачів, прав доступу та записів задач. Таким чином забезпечується цілісність даних у ситуаціях одночасного звернення кількох користувачів.

Патерн Producer–Consumer використовується у сервісах NotificationService та WebSocketGateway, які формують черги подій, виконують асинхронну доставку повідомлень і транслюють оновлення у реальному часі. Це дозволяє системі реагувати на події без затримок і зберігати стабільність при високій кількості одночасних клієнтських з’єднань.

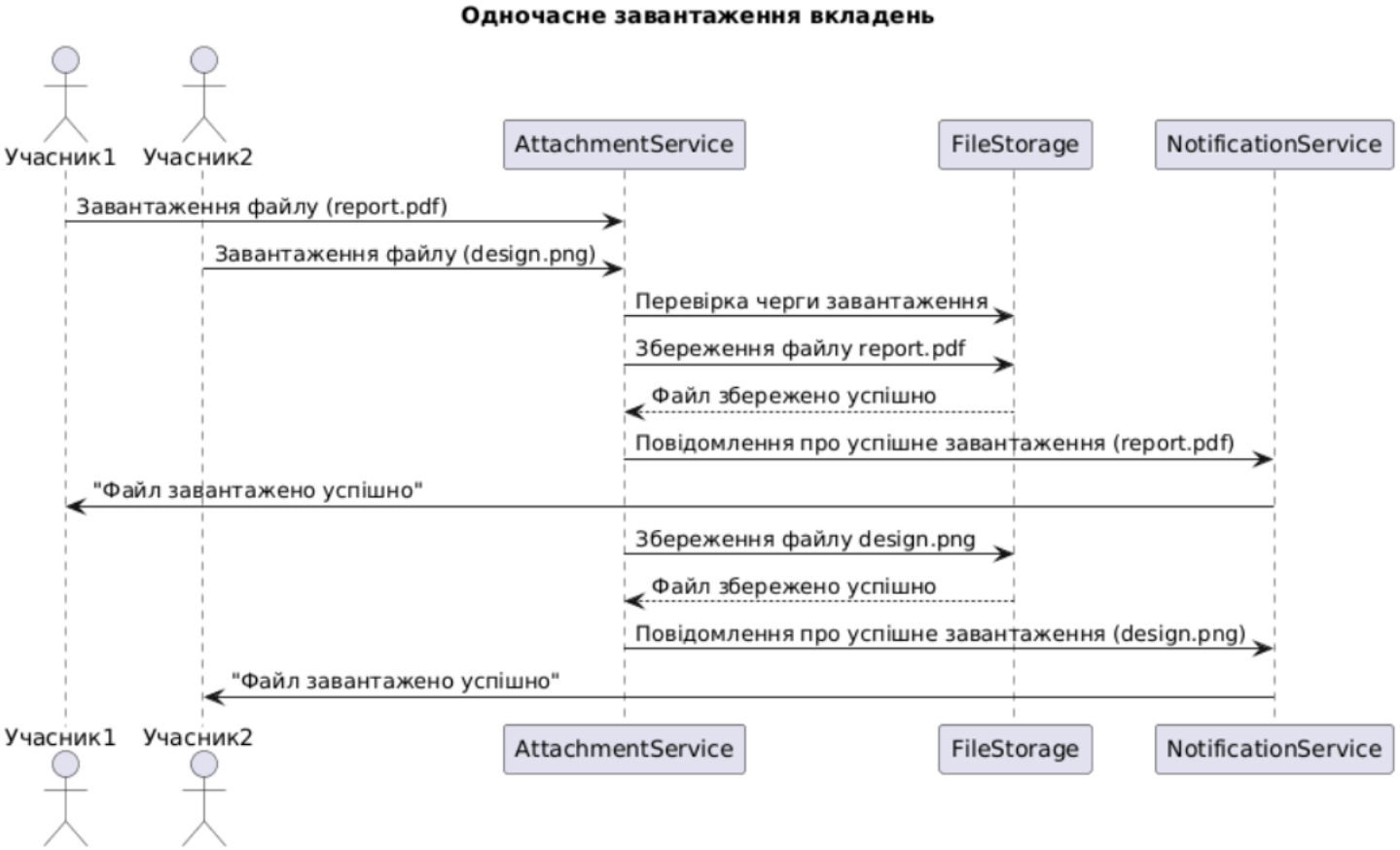
Сценарій 1. Одночасне оновлення задачі двома користувачами

У даному сценарії два учасники команди одночасно змінюють статус однієї й тієї ж задачі.
Система фіксує перший успішно оброблений запит як актуальний, а другому користувачеві повертає повідомлення про конфлікт версій та пропонує оновити сторінку для отримання останнього стану.
UML діаграма створена за допомогою платформи PlantUML, також за посиланням наведено вихідний код діаграми у репозиторії GitHub.



Сценарій 2. Одночасне завантаження вкладень

Двоє користувачів одночасно завантажують великі файли до однієї задачі. Сервіс AttachmentService реалізує чергу обробки запитів, гарантуючи коректність запису та унікальність ідентифікаторів файлів. У результаті обидва вкладення зберігаються без втрати даних.
UML діаграма створена за допомогою платформи PlantUML, також за посиланням наведено вихідний код діаграми у репозиторії GitHub.

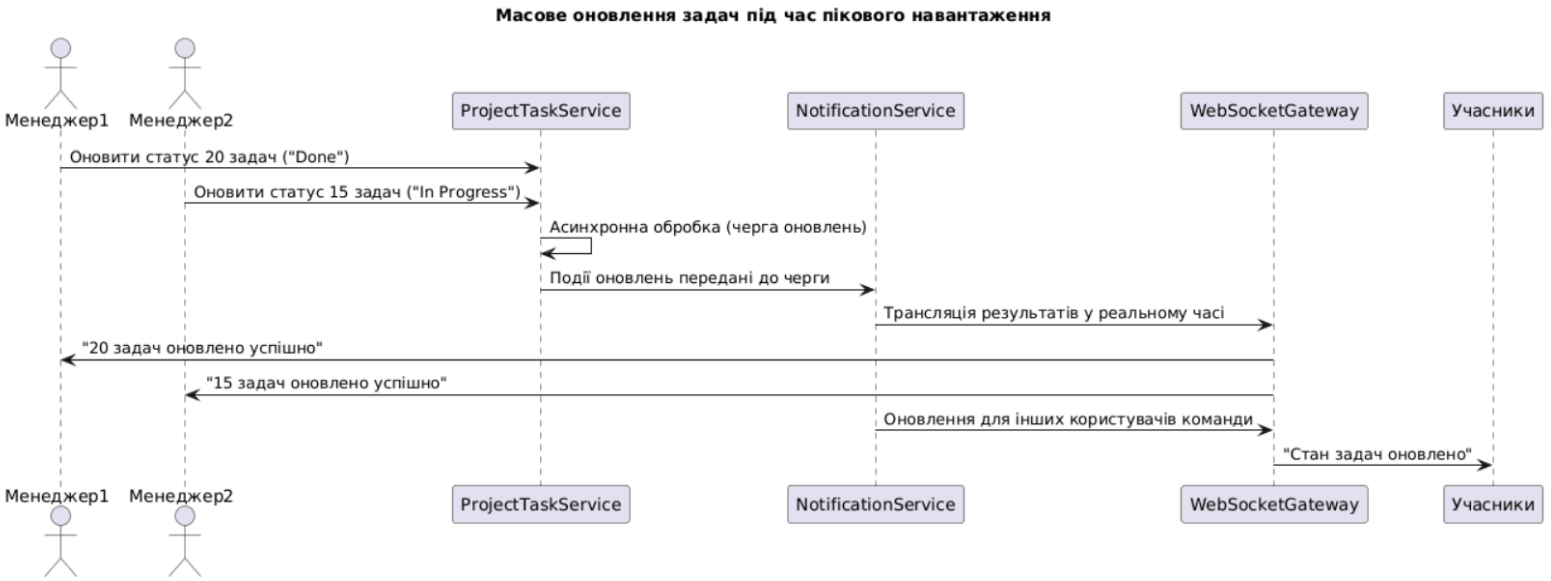


Сценарій 3. Одночасне редагування коментарів

Двоє користувачів взаємодіють із секцією коментарів однієї задачі: перший редагує наявний запис, другий — додає новий.
Сервіс CollaborationService синхронізує зміни, а WebSocketGateway транслює актуальний стан усім учасникам у реальному часі, забезпечуючи узгодженість даних.
UML діаграма створена за допомогою платформи PlantUML, також за посиланням наведено вихідний код діаграми у репозиторії GitHub.

Сценарій 4. Масове оновлення задач під час пікового навантаження

Кілька менеджерів одночасно змінюють статуси великої кількості задач. Сервіс ProjectTaskService використовує асинхронні воркери для паралельної обробки, тоді як NotificationService формує черги подій і поступово розсилає результати через WebSocketGateway. Це забезпечує стабільність системи без втрати даних.
UML діаграма створена за допомогою платформи PlantUML, також за посиланням наведено вихідний код діаграми у репозиторії GitHub.



Сценарій 5. Одночасне видалення проєкту та створення задачі

Менеджер видаляє проєкт у той самий момент, коли учасник намагається створити в ньому нову задачу.
Сервіс ProjectService блокує доступ до проєкту на час видалення, запобігаючи створенню нових об’єктів у некоректному стані. Користувач отримує сповіщення про недоступність проєкту.
UML діаграма створена за допомогою платформи PlantUML, також за посиланням наведено вихідний код діаграми у репозиторії GitHub.

4. Нефункціональні вимоги

4.1 Системні вимоги

ID	Назва	Опис
NFR-SYS-01	Доступність сервісу	Основні бекенд-сервіси повинні бути доступні не менше ніж 99.9% часу в місяць (за винятком регламентного обслуговування), щоб користувачі мали безперервний доступ до задач і проєктів.
NFR-SYS-02	Продуктивність / Латентність	95-й перцентиль відповіді критичних запитів (перегляд списку задач, зміна статусу задачі, додавання коментаря) не повинен перевищувати ~300 мс при нормальному навантаженні робочого простору малого/середнього розміру.
NFR-SYS-03	Масштабованість	Кожен сервіс повинен підтримувати горизонтальне масштабування без змін бізнес-логіки. Зокрема, шар WebSocketGW має дозволяти збільшувати кількість одночасних підключень користувачів без простою системи.
NFR-SYS-04	Стійкість та відновлення	Збій окремого сервісу (наприклад, нотифікацій) не повинен блокувати критичні дії (створення задачі, оновлення статусу). Має бути повторна обробка подій, черги повідомлень та стійке сховище для даних.
NFR-SYS-05	Аудит і трасування дій	Система повинна фіксувати критичні зміни (хто змінив статус задачі, кому призначено дедлайн) з часовими мітками. Логи повинні бути придатні для аналізу інцидентів та відстеження відповідальності.
NFR-SYS-06	Зберігання та резервні копії	Дані про задачі, коментарі та вкладення мають зберігатися у стійкому сховищі. Повинні виконуватись регулярні бекапи бази даних і мати можливість відновлення принаймні за останні 30 днів.

NFR-SYS-07	Спостережуваність / Моніторинг	Ключові метрики (латентність API, помилки 5xx, затримка обробки подій, навантаження WebSocketGW) повинні збиратися, відображатися на дашбордах і мати налаштовані алерти для оперативного реагування.

4.2 Вимоги по безпеці

ID	Назва	Опис
NFR-SEC-01	Аутентифікація сесій	Будь-яка дія, окрім реєстрації та входу, вимагає дійсного токена доступу. Токени повинні мати обмежений строк життя і оновлюватися через безпечний механізм (refresh). Значення токенів не повинно бути доступне стороннім скриптам.
NFR-SEC-02	Авторизація та ролі	Кожна операція, що змінює дані (створити задачу, призначити виконавця, оновити статус), повинна перевіряти дозволи користувача на бекенді. Наявність доступу до проєкту не означає автоматично право редагувати всі задачі.
NFR-SEC-03	Захист даних у транзиті та на зберіганні	Увесь трафік між клієнтом і сервером має бути шифрований (TLS). Вкладення та чутливі поля повинні зберігатися у зашифрованому вигляді. Доступ до файлів відбувається через короткоживучі підписані URL з перевіркою прав.
NFR-SEC-04	Валідація вводу та захист від зловживань	Система повинна виконувати валідацію і нормалізацію вхідних даних (назви задач, коментарі) для запобігання XSS та ін’єкціям. Критичні кінцеві точки (логін, створення задачі) мають бути захищені rate limiting.
NFR-SEC-05	Аудит і незмінність логів	Важливі операції (зміна статусу задачі, перепризначення виконавця, видалення задачі) мають журналюватися з вказанням користувача і часу. Логи мають бути захищені від несанкціонованої зміни.

NFR-SEC-06	Контроль доступу до вкладень	Користувач не повинен мати змоги отримати файл з задачі, до якої він не має доступу. Генерація посилання на вкладення завжди повинна включати перевірку прав доступу.
NFR-SEC-07	Політика персональних даних	Персональні дані користувачів (ім’я, email) мають бути видалені на запит власника або після видалення облікового запису з робочого простору. Історичні журнали мають зберігати мінімум PII.

4.3 Вимоги по документації

ID	Назва	Опис
NFR-DOC-01	Документація API	Для основних endpoint-ів бекенд-сервісів (автентифікація, задачі, коментарі, аналітика) має існувати формальний опис контракту (наприклад OpenAPI/Swagger), який доступний команді розробки та тестування.
NFR-DOC-02	Керівництво користувача	Повинні існувати інструкції для ролей “Менеджер команди” і “Учасник команди” (як створити проєкт, задачу, оновити статус, додати файл, відслідковувати дедлайни).
NFR-DOC-03	Опис ролей і прав	Повинен бути зафіксований список прав доступу (хто може створювати проєкти, змінювати статуси задач, призначати виконавців), щоб уникати неочікуваного розширення прав.
NFR-DOC-04	Архітектурна документація	Високорівнева схема сервісів (SOA), зв’язки між ними, зберігання даних, механізм реального часу і залежності від інфраструктури мають бути задокументовані й оновлюватися при змінах системи.
NFR-DOC-05	Runbook / Інциденти	Потрібні базові процедури реагування: які метрики критичні, які пороги алертів, кого повідомляти при

		деградації сервісу або відсутності сповіщень користувачам.
NFR-DOC-06	Політика резервного копіювання	Повинно бути описано, як часто робляться резервні копії даних, як довго вони зберігаються, які дані архівуються та як відновити стан системи (disaster recovery на високому рівні).