Одеський національний політехнічний університет

Інститут комп'ютерних систем

Кафедра інформаційних систем

**КУРСОВА РОБОТА**

з дисципліни “Об’єктно-орієнтоване програмування”

Тема “Розробка десктопного застосунку для індивідуального планування завдань ”

Студента \_\_II\_\_ курсу АІ-166 групи

напряму підготовки 6.050101 – «Комп’ютерні науки»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник: Рудніченко М. Д.\_\_\_\_\_\_\_\_\_\_

Національна шкала\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали

м. Одеса – 2018 рік

Міністерство освіти і науки України

Одеський національний політехнічний університет

Інститут комп’ютерних систем

Кафедра інформаційних систем

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ

|  |  |
| --- | --- |
| студенту | Дідуху Едварду Геннадійовичу |
|  | (прізвище, ім’я, по батькові) |

1. Тема роботи: *Розробка десктопного застосунку для індивідуального планування завдань*
2. Термін здачі студентом закінченої роботи
3. Початкові дані до роботи: *програма має містити класи користувача, проекту, завдання, класи-контролери для користувацької взаємодії з доданком та класи представлення для реалізації користувацького інтерфейсу. Програма також буде містити базу даних користувачів(ідентифікатор, хеш-код, логін користувача, пароль), проектів(ідентифікатор, назва, опис, термін здачі проекту) та завдань(ідентифікатор, назва, опис, пріоритет, термін здачі та статус завдання).*
4. Зміст пояснювальної записки (перелік питань, що їх належить розробити): *Вступ. Аналіз задачі розробки десктопного застосунку. Проектування архітектури десктопного застосунку. Реалізація десктопного застосунку. Забезпечення якості десктопного застосунку. Висновки.*
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень): *Діаграми UML, діаграма бази даних*
6. Словесний опис програмного забезпечення: *Десктопний застосунок для індивідуального планування завдань*

*Доданок дозволяє:*

1. *Зареєструватися та увійти у свій аккаунт;*
2. *Створити власний проект;*
3. *У межах проекту створити власні завдання;*
4. *Змінювати статус завдання: до виконання → у процесі → виконано;*
5. *Редагувати та видаляти проекти/завдання.*

*Для менеджменту користувачів, проектів та завдань використовується база даних.*

*Застосунок буде складатися з основних вікон:*

1. *Стартове вікно, де користувачеві дається вибір: зареєструватись чи увійти у систему.*
2. *Вікно реєстрації, де користувач може ввести свої логін та пароль для реєстрації нового аккаунту в системі.*
3. *Вікно авторизації, де користувач може ввести логін та пароль зареєстрованого аккаунту для отримання доступу до програми.*
4. *Вікно збережених проектів, де користувач може переглянути створені ним проекти, обрати один зі списку та відкрити/видалити його. Також користувач може перейти у вікно створення нового проекту, вікно збережених проектів або вийти з системи.*
5. *Головне вікно програми, де користувач може переглянути завдання до відкритого проекту, створити нове завдання, відредагувати/видалити проект/обране завдання, змінити статус будь-якого завдання. Також користувач може перейти у вікно створення нового проекту, вікно збережених проектів або вийти з системи.*
6. *Вікно створення проекту, де користувач може ввести необхідну інформацію для створення нового проекту та підтвердити/відмінити його створення.*
7. *Вікно редагування проекту, де користувач може змінити інформацію про обраний проект та підтвердити/відмінити зміни.*
8. *Вікно створення завдання, де користувач може ввести необхідну інформацію для створення нового завдання та підтвердити/відмінити його створення.*
9. *Вікно редагування завдання, де користувач може змінити інформацію про обраний проект та підтвердити/відмінити зміни.*

|  |  |  |  |
| --- | --- | --- | --- |
| Завдання видано | \_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_ | Рудніченко М. Д. |
|  | (дата) | (підпис) |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Завдання прийнято до виконання | \_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_ | Дідух Е. Г. |
|  | (дата) | (підпис) |  |

**ЗМІСТ**

ВСТУП 4

1. АНАЛІЗ СПЕЦИФІКИ ПРЕДМЕТНОЇ ОБЛАСТІ 5

1.1. Особливості розробки десктопного застосунку 5

1.1.1 Принципи розробки користувацького інтерфейсу 6

1.2 Аналіз аналогів на ринку 10

1.3. Використані технології розробки 11

2. ПРОЕКТУВАННЯ 12

2.1. Backlog 12

2.2. БД 14

2.3. CRC 15

2.4. UML 16

2.4.1. Use Case Diagram 16

2.4.2. Activity Diagram 17

2.4.3. Class Diagram 21

2.5. Прототип 22

3. ОПИС ПРОЦЕСУ РОБОТИ 27

ВИСНОВКИ 39

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 40

ДОДАТОК А 41

**ВСТУП**

Впродовж останніх років спостерігається значний приріст швидкості інновацій у різних напрямах діяльності. Вимоги до співробітників, особливо ІТ-компаній зростають, та основним вектором руху діяльності підприємств та фрілансерів є швидкість виконання роботи. У централізованих компаніях використовуються загальні робочі плани, розраховані на команди працівників, у той самий час індивідуальне планування роботи для фрілансерів стає все важчим у зв’язку з частою необхідністю виконувати декілька проектів одночасно для збереження власної конкурентоспроможності.

Сучасна методологія планування робочого процесу передбачає розподілення цілісного проекту на певну кількість відносно незалежних завдань, кожне з яких може виконуватись без необхідності виконання інших. Така методологія зарекомендувала себе для роботи над командними проектами, у яких різні частини проекту розподіляються між різними групами працівників, однак такий підхід до менеджменту праці можна використовувати і для індивідуальної організації робочого процесу.

Десктопний застосунок для індивідуального планування завдань – це система для розподілення, організації завдань та менеджменту часу під час робочого процесу.

Застосунок створений за допомогою мови програмування Java 1.8 із застосуванням технологій: JavaFX, Maven, JDBC, СУБД MySQL під ОС Windows.

Метою застосунку є полегшення процесу організації роботи та моніторингу прогресу робочого процесу для фрілансерів/студентів різних напрямків діяльності.

1. **АНАЛІЗ СПЕЦИФІКИ ПРЕДМЕТНОЇ ОБЛАСТІ**

**1.1 Особливості розробки десктопного застосунку**

Зараз більша частина нового програмного забезпечення (ПЗ) розроблюється для мобільних платформ чи у якості веб-додатків. Такий підхід набув популярності завдяки стрімкому розвитку інтернет-технологій та удосконалення мобільних платформ.

У порівнянні з мобільними та веб-застосунками, десктопні програми мають певні переваги та недоліки:

Переваги:

* відсутність необхідності доступу до Інтернету;
* стандартизовані та чіткі правила розробки ПЗ;
* велика кількість паттернів проектування та фреймворків для розробки ПЗ;
* можливість розробки на більшій частині мов програмування (C, C++, C#, Java і т.д.);
* порівняно простий процес тестування та розгортання ПЗ;
* простий спосіб ліцензування ПЗ;

Недоліки:

* локальне зберігання даних для роботи з ПЗ;
* залежність від технічних характеристик комп’ютера, на якому розгортається програма;
* можливі труднощі оновлення ПЗ;
* порівняно низька гнучкість;

При розробці десктопного ПЗ основними пріоритетами є забезпечення автономної роботи програми, мінімізація використання ресурсів робочої станції користувача та максимізація швидкості роботи застосунку.

Десктопний доданок може містити у собі певний функціонал, що використовує онлайн-ресурси для зберігання чи обробки даних, однак основна бізнес-логіка ПЗ виконується на комп’ютері користувача.

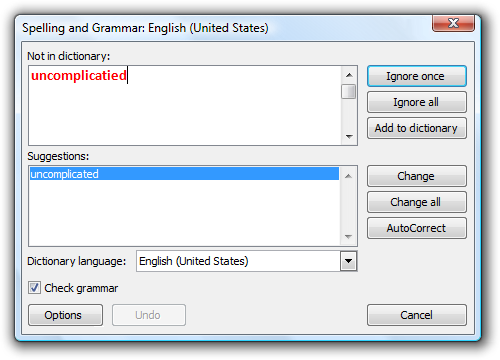
Фреймворки для розробки десктопного ПЗ існують довше, ніж для інших різновидів доданків, отже принципи розробки програм для персональних комп’ютерів без орієнтації на Інтернет стандартизовані та уніерсальні.

У даній курсовій роботі застосунок розроблювався з використанням паттерна MVC. Цей паттерн розробки дозволяє розділити програму на три основні частини – модель, контролер та представлення, що у свою чергу забезпечує можливість незалежної роботи над складовими застосунку та внесення змін у одну частину програми без необхідності змінювати інші.

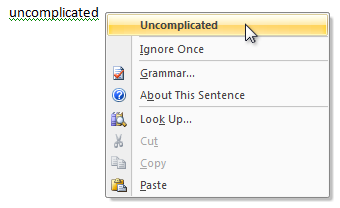
**1.1.1 Принципи розробки користувацького інтерфейсу**

Десктопний додаток має бути потужним і одночасно простим. Завдяки ретельно збалансованому вибору та презентації ви можете досягти і потужності, і простоти.

Потужність:



Простота:



**Що робить програму потужною?**

Програма може вважатися потужною, якщо вона містить повнофункціональні можливості, має величезну ширину функціональності, намагаючись стати всім необхідним для користувачів. Такий проект, швидше за все, не буде успішним, тому що нецільовий набір функцій навряд чи задовольнить потреби будь-кого. Це не той тип потужності, який слід розглядати.

Програма є потужною, коли вона має правильну комбінацію цих характеристик:

* Ефективність;
* Універсальність;
* Прямота;
* Гнучкість;
* Інтегрованість;
* Розширюваність.

Деякі з цих характеристик залежать від сприйняття користувача та відносних можливостей користувачів. Те, що вважається потужним, може змінюватися з часом, тому сьогоднішня функція розширеного пошуку може бути звичайною завтра.

**Що робить програму простою?**

Простота - зменшення або усунення атрибута дизайну, який цільові користувачі знають і вважають несуттєвими.

На практиці, простота досягається шляхом вибору правильного набору функцій та правильного представлення функцій. Це зменшує несуттєві, як реальні, так і сприймані.

Простота залежить від сприйняття користувачів. Для правильного спрощення користувацького інтерфейсу програми потрібно знати потреби цільової аудиторії користувачів ПЗ та розглядати функціонал, що їх задовольняє.

**Простота чи простота використання:**

Простота при правильному застосуванні призводить до легкості використання. Але простота і легкість використання не ті самі поняття. Зручність використання досягається, коли користувачі можуть самостійно виконувати завдання без складнощів або плутанини протягом потрібного періоду часу.

Усі користувачі, незалежно від складності їх роботи, хочуть, щоб вона виконувалася з мінімальною кількістю непотрібних зусиль.

Простота - найефективніший спосіб досягти простоти використання, а зручність використання - рівне використання. Складні, важкодоступні функції завзичай не використовуються більшістю користувачів. На відміну від них, прості, елегантні конструкції, які добре виконують свою функцію, використовуються регулярно та викликають у користувача ПЗ позитивну емоційну реакцію.

Щоб досягти простоти використання програми, зберігаючи її потужність, слід насамперед обрати центральний набір функцій, які має виконувати ПЗ.

**Вибір правильного набору функцій**

Наведені нижче методи розробки надають користувачам необхідні їм функції, досягаючи простоти шляхом фактичного зменшення кількості несуттєвих елементів інтерфейсу:

* Визначте функції, які необхідні користувачеві;
* Видаліть непотрібні елементи;
* Видаліть непотрібну надмірність.

Для досягнення цих цілей можна використовувати наступні принципи:

* Об'єднання того, що потрібно об'єднати;
* Розділення того, що слід розділити;
* Видалення того, що можна усунути;
* Послідовність, конфігурація та узагальнення;
* Розташування елементи у потрібному місці;
* Вибір правильних елементів керування.

**1.2 Аналіз аналогів на ринку**

На даний момент на ринку ПЗ присутня значна кількість програмних продуктів для контролю робочого процесу та тайм-менеджменту. Прикладами подібних програмних додатків є: Basecamp, Trello, Slack, Jira і т.д.

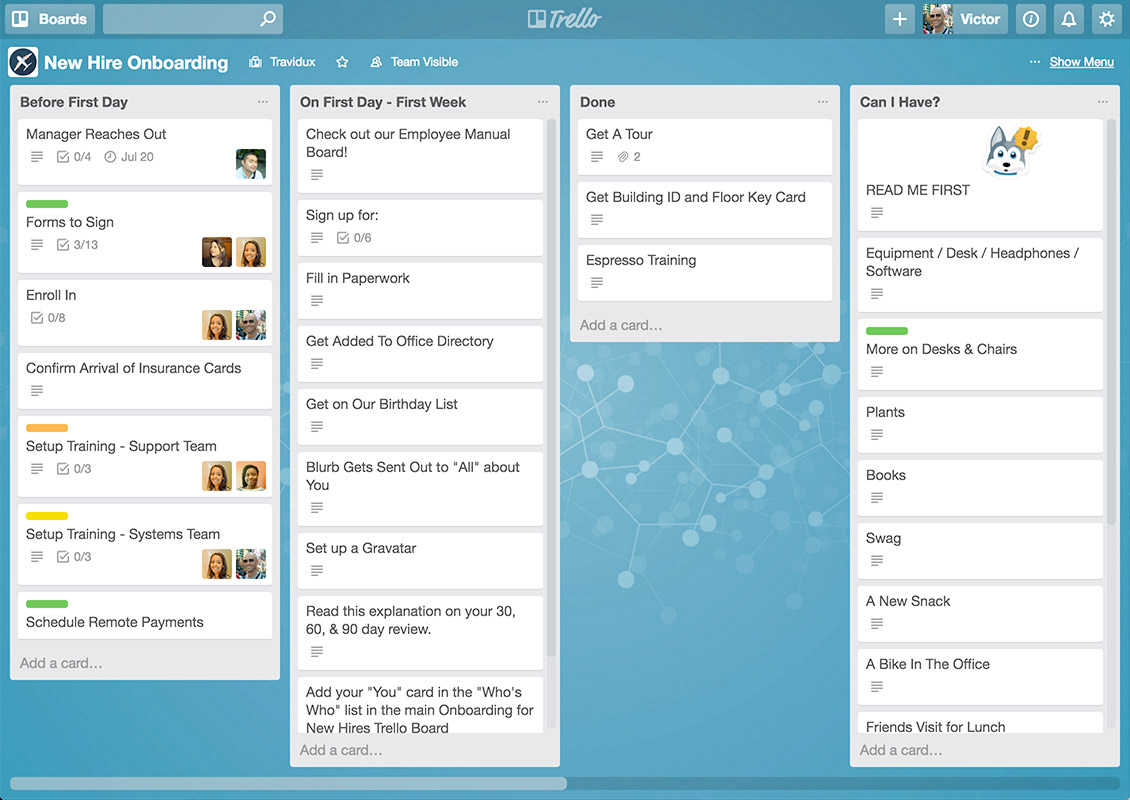


Рис. 1.2.1 Користувацький інтерфейс Trello

Загальною рисою вищезазначених програмних продуктів є їх орієнтованість на контроль робочого процесу командних проектів за участі менеджеру проекту та учасників команд працівників та те, що всі вони є веб-застосунками. Проектування подібного ПЗ у якості веб-сервісу виправдане у випадку, коли над проектом повинна мати можливість працювати велика кількість працівників, проте для індивідуального планування робочого процесу цей функціонал є надмірним.

**1.3 Використані технології розробки**

|  |  |  |  |
| --- | --- | --- | --- |
| Назва | Версія | Посилання | Опис |
| Java | 1.8 | - | - |
| FXML | 1 | - | Спеціалізована мова розмітки фреймворку JavaFX |
| JetBrains IntelliJ IDEA Community Edition | 2017 2.3 | [https://www.jetbrains.com/idea/download/#section=windows](https://www.jetbrains.com/idea/download/%23section=windows) | Інтегрована середа розробки Java |
| Maven | 3.3.9 | - | Інструмент компіляції та збірки Java-проекту |
| JavaFX Framework | 8.0.141 | - | Фреймворк для створення Java-додатків з графічним користувацьким інтерфейсом |
| Gluon SceneBuilder | 9.0.1 | <http://gluonhq.com/products/scene-builder/> | Інструмент візуального моделювання користувацького інтерфейсу |
| MySQL | 6.3 | <https://dev.mysql.com/downloads/windows/> | СУБД |

1. **ПРОЕКТУВАННЯ**

У даній курсовій роботі проектування та розробка програмного продукту виконувались за методологією Scrum.

**Scrum** – це гнучка система управління роботою з акцентом на розробку програмного забезпечення. Він призначений для команд від трьох до дев’яти розробників, які розбивають свою роботу на дії, які можуть бути завершені за часовими ітераціями, які називаються спринтами (30 днів або менше, частіше за два тижні), і відслідковує прогрес та перепланування в 15-хвилинному режимі роботи, денні зустрічі, що називаються **daily scrum**. Підходи до координації роботи декількох команд у більших організаціях включають широкомасштабну програму Scrum (LeSS), масштабовану програму Agile (SAFe) та **scrum of scrums**, серед іншого.

Далі у цьому розділі представлені послідовні етапи проектування застосунку за даною методологією.

**2.1 Backlog**

Беклог продукту містить упорядкований список вимог до продукту, який команда Scrum підтримує для продукту. Формат беклогу продуктів відрізняється, загальні формати включають в себе User stories, Use Case або будь-які інші вимоги формату, яку команда вважає корисною. Вони будуть визначати особливості, виправлення помилок, нефункціональні вимоги тощо – все, що потрібно зробити, щоб успішно доставити життєздатний продукт. Власник продукту визначає пріоритетність завдань беклогу (PBI) на основі таких міркувань, як ризик, вартість або термін виконання.

**Product Backlog:**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Task | Priority | User stories |
| R1 | Створення нових завдань | 10 | Як користувач, я можу створити нову задачу, задати їй ім'я, пріоритет і дедлайн, щоб завдання відображалася в категорії «прийнято до виконання». |
| R2 | Зміна статусу завдання | 9 | Як користувач, я можу пересувати завдання між категоріями «до виконання», «в процесі» і «виконано», щоб вони відображалися у відповідній категорії. |
| R3 | Видалення завдання | 9 | Як користувач, я можу видаляти створену задачу, щоб вона не відображалася у відповідній категорії. |
| R4 | Редагування завдання | 8 | Як користувач, я можу редагувати опис, пріоритет і дедлайн створеної завдання, поки вона відображається в категорії «Прийнято до роботи», щоб внести в задачу необхідні корективи перед переміщенням завдання в «У процесі». |
| R5 | Створення проекту | 7 | Як користувач, я можу створити новий проект, задати йому заголовок і опис, щоб він відображався в списку збережених проектів. |
| R6 | Відкриття проекту | 7 | Як користувач, я можу відкрити раніше збережений проект, щоб працювати з ним в поточній сесії. |
| R7 | Редагування проекту |  | Як користувач, я можу змінити назву і опис раніше збереженого проекту, щоб оновити інформацію про проект. |
| R8 | Видалення проекту | 7 | Як користувач, я можу видалити проект, щоб видалити з БД інформацію про проект та сам проект не відображався в вікні збережених проектів. |
| R9 | Реєстрація/Авторизація | 6 | Як користувач, я можу зареєструватися в системі, щоб отримати доступ до основного функціоналу програми.  Як зареєстрований користувач, я можу увійти в систему для отримання доступу до своїх проектів. |

**2.2 БД**

У розробленому програмному застосунку база даних використовується для зберігання інформації про користувачів, відповідні їм проекти та завдання. У якості СУБД обрана MySQL 6.3.

База даних включає в себе таблиці:

* користувачі;
* проекти;
* завдання.

Основний тип зв’язку між таблицями – один-до-багатьох: один користувач може мати багато збережених проектів, один проект може містити багато завдань до виконання.

**Схема бази даних:**

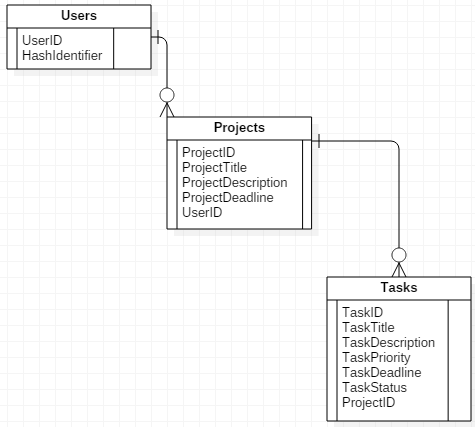


Рис. 2.1 Схема бази даних

**2.3 CRC**

**CRC-Картки –** це інструмент мозкового штурму що використовується при проектуванні об’єктно-орієнтованого ПЗ. Спершу вони були запропоновані Вордом Каннінгемом та Кентом Беком як навчальний інструмент, але також були популярними і серед спеціалістів і рекомендуються прихильниками екстремального програмування. Мартін Фаулер описав CRC-картки як життєздатну альтернативу діаграмам послідовностей в UML щоб проектувати динаміку взаємодії між об’єктами.

У розробленому програмному додатку наявні функціональні класи моделі:

* Користувач;
* Проект;
* Завдання.

|  |  |
| --- | --- |
| User | |
| * Create User * Validate User | * Project |

|  |  |
| --- | --- |
| Project | |
| * Create Project * Edit Project * Save Project * Assign Project to User * Delete Project | * User * Task |

|  |  |
| --- | --- |
| Task | |
| * Create Task * Edit Task * Save Task * Move Task * Assign Task to Project * Delete Task | * Project |

**2.4 UML**

Діаграми UML дозволяють візуально описати функціональні можливості програмного продукту за допомогою спеціалізованої мови візуального моделювання.

У даній курсовій роботу програмний додаток описаний за допомогою:

* діаграми прецедентів(Use Case);
* діаграми діяльності(Activity);
* діаграми класів(Class).

**2.4.1 Use Case Diagram**

**Діаграма прецедентів** — в UML, діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

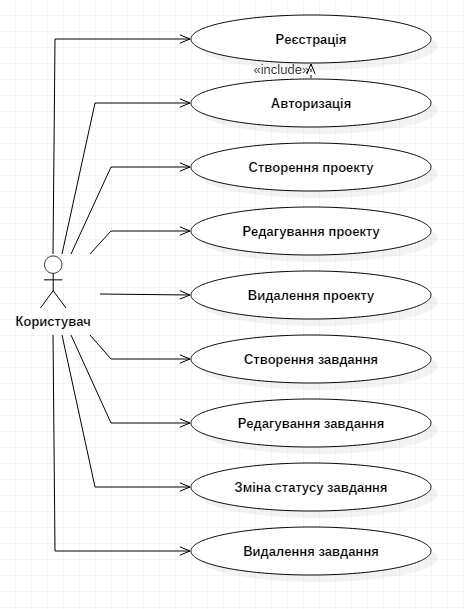


Рис 2.2 Загальна діаграма прецедентів

**2.4.2 Activity Diagram**

**Діаграма діяльності** — в UML, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності може дозволяти виконання декількох потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

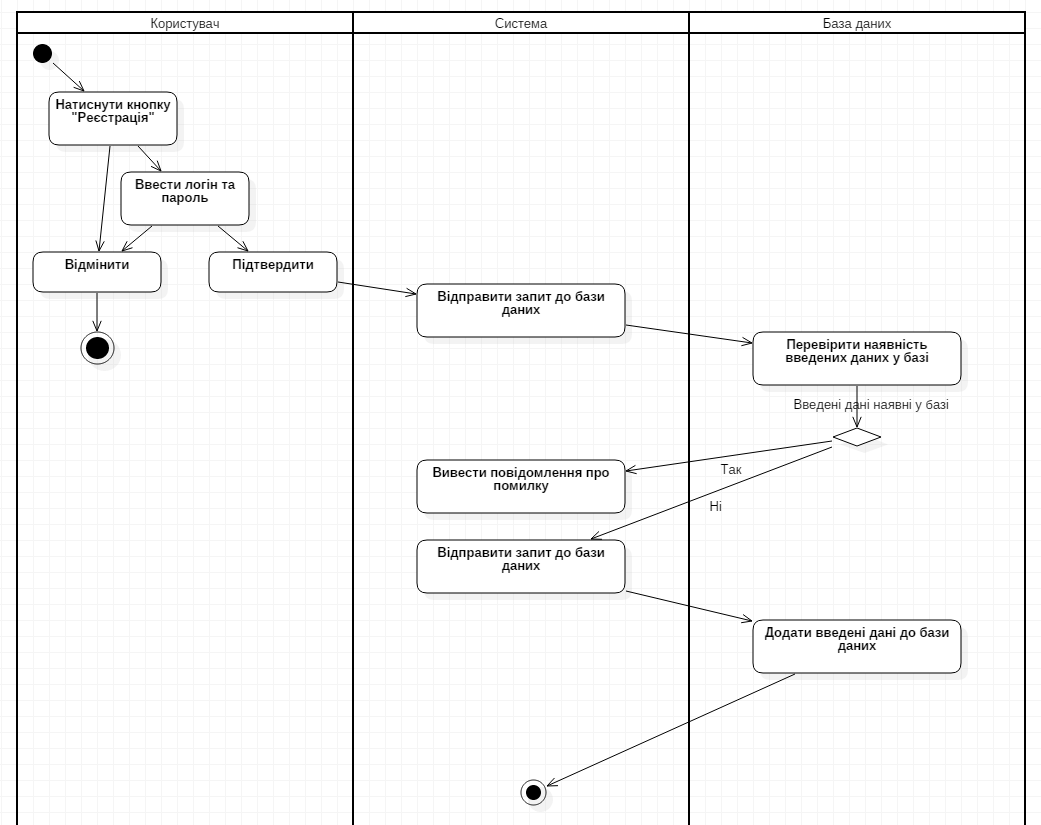


Рис. 2.3 Діаграма діяльності прецеденту «Реєстрація»

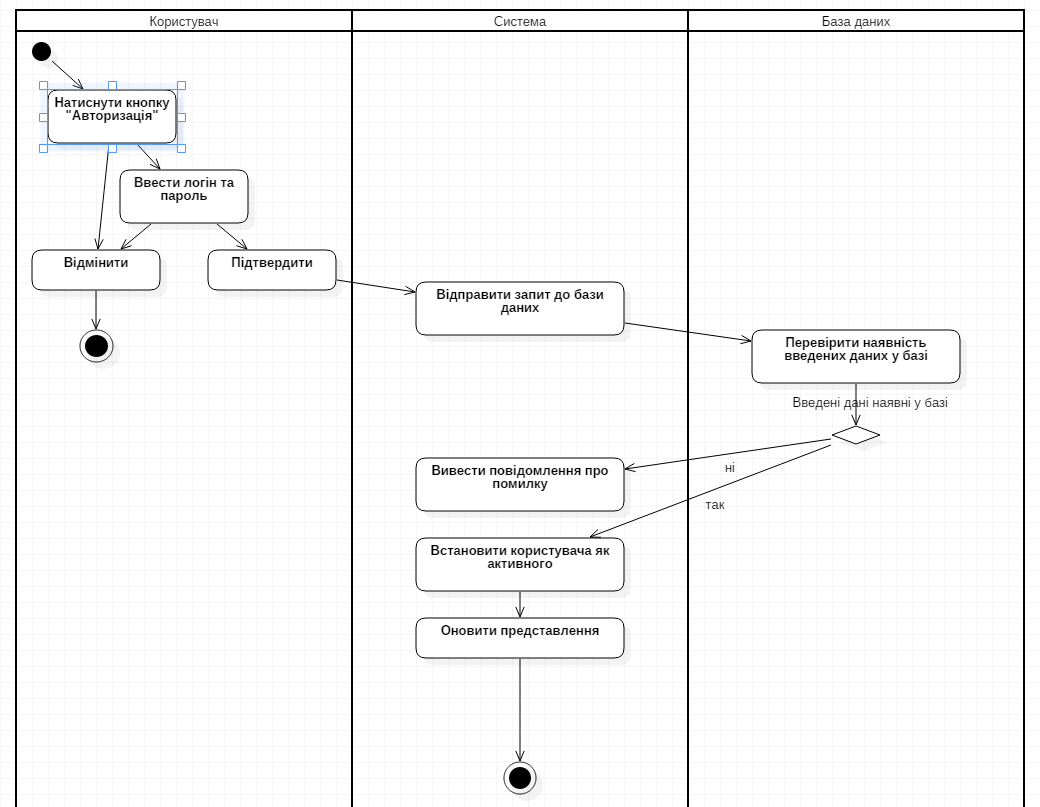


Рис. 2.4 Діаграма діяльності прецеденту «Авторизація»

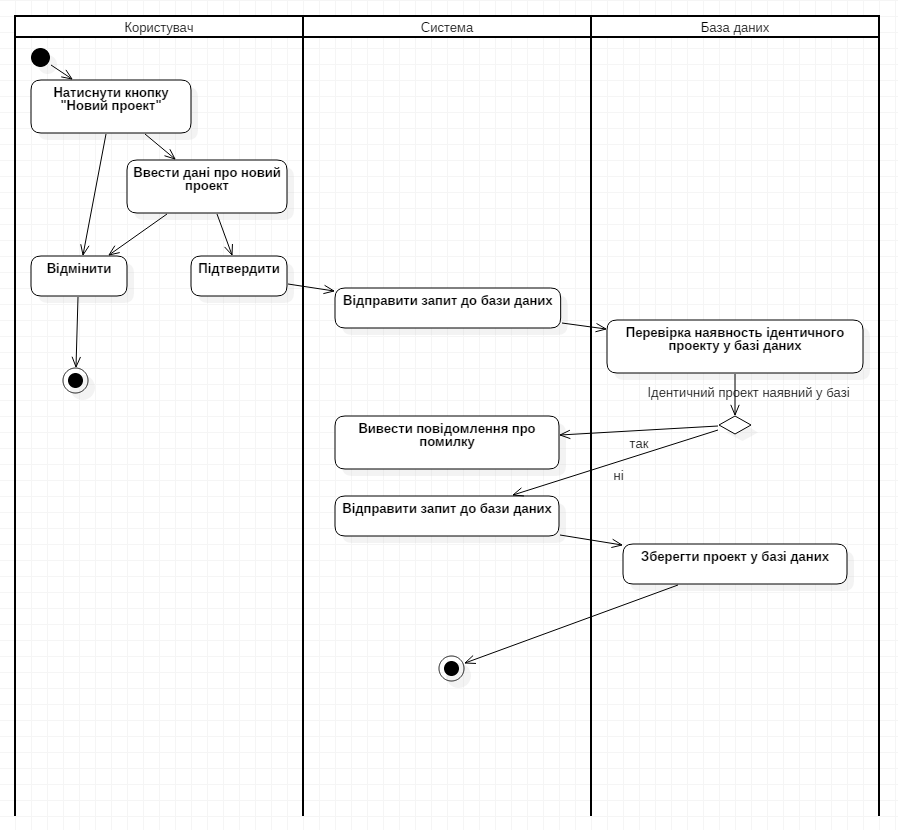


Рис. 2.5 Діаграма діяльності прецеденту «Створення нового проекту»

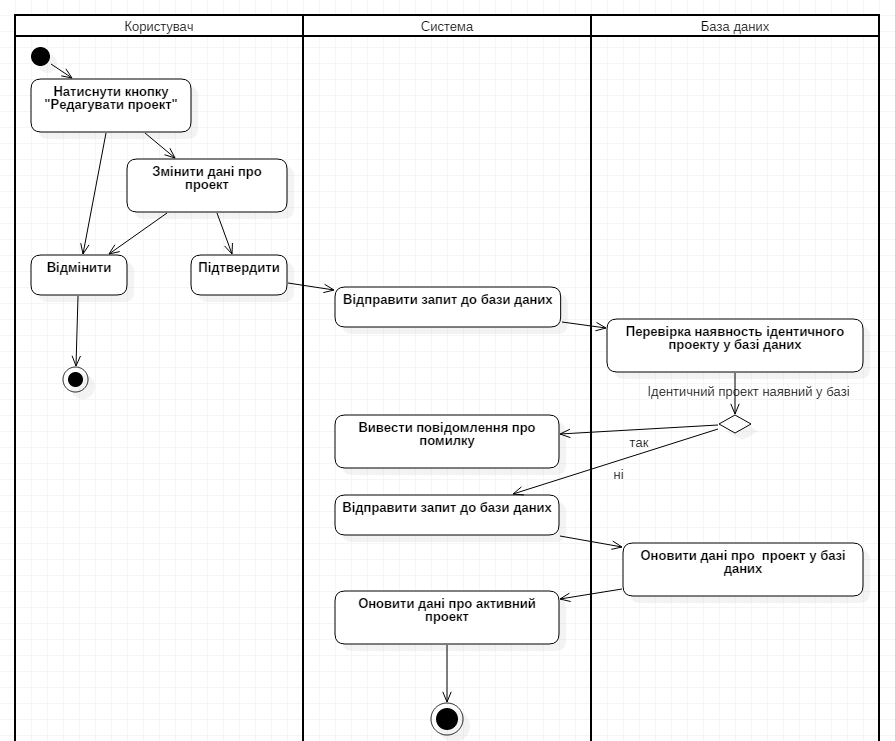


Рис. 2.6 Діаграма діяльності прецеденту «Редагування проекту»

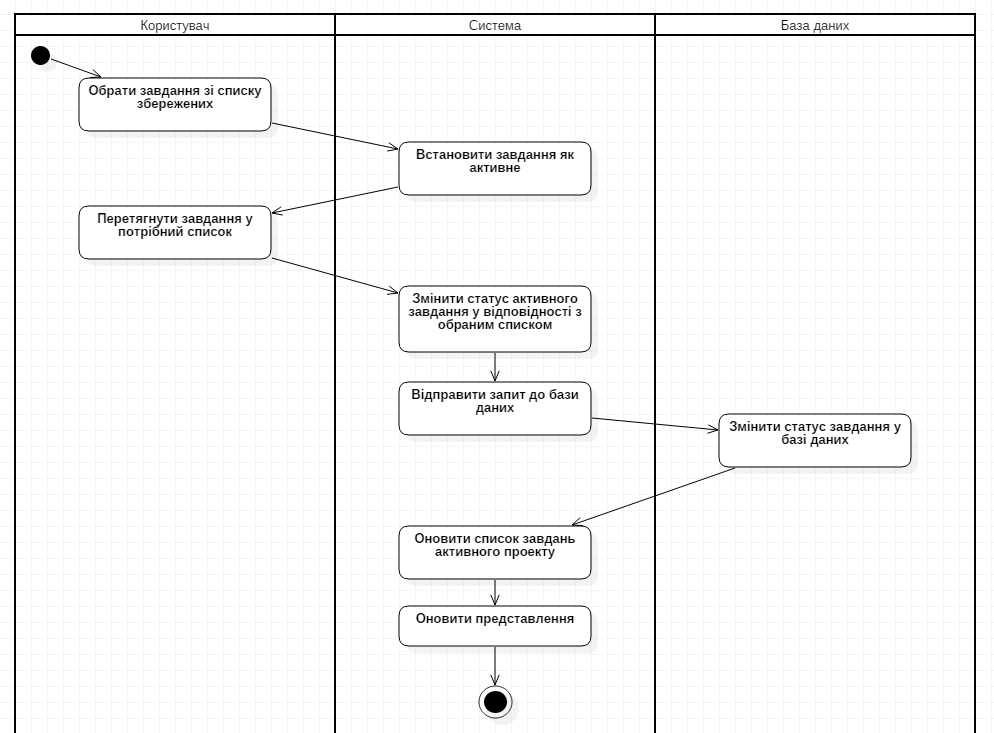


Рис. 2.7 Діаграма діяльності прецеденту «Зміна статусу завдання»

**2.4.3 Class Diagram**

**Діаграма класів** — статичне представлення структури моделі. Відображає статичні елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм. Діаграма класів служить для представлення статичної структури моделі системи в термінології класів об’єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об’єкти й кооперації, а також їхні відносини.

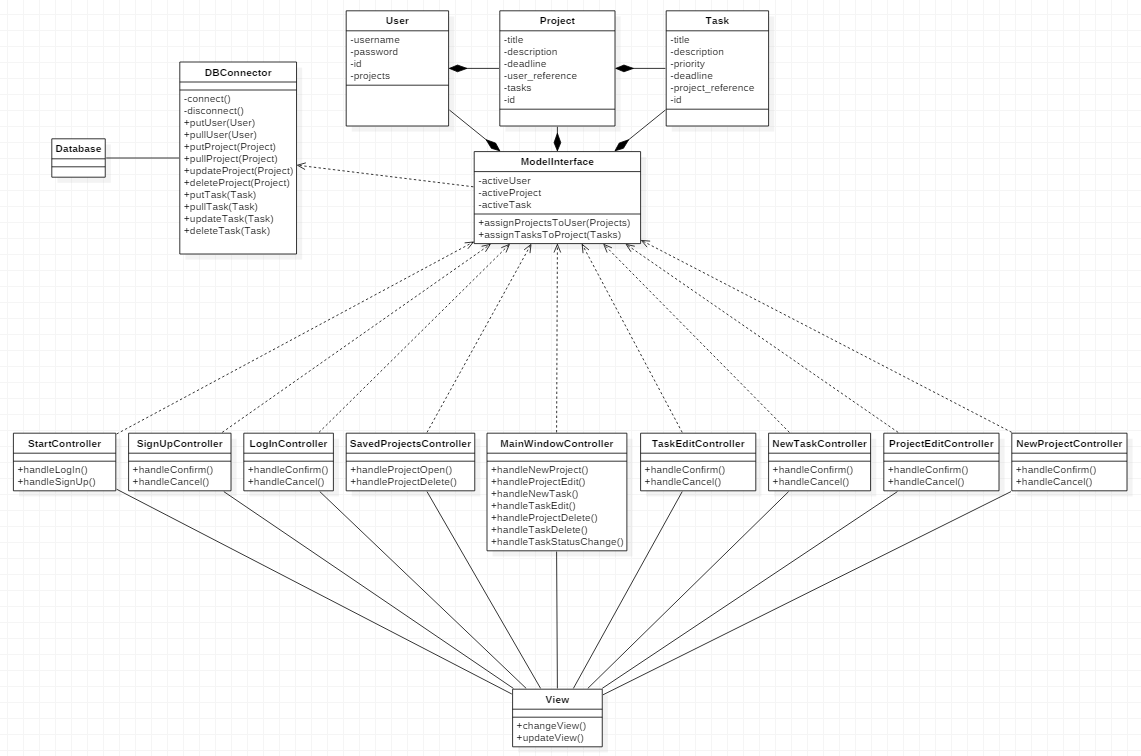


Рис. 2.8 Діаграма класів

**2.5 Прототип**



Рис. 2.9 Схема переходів між вікнами програми

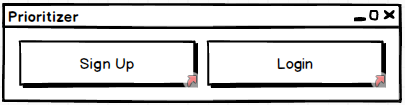


Рис. 2.10 Стартове вікно програми

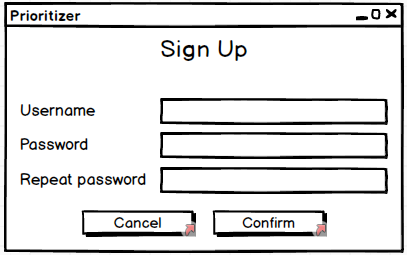


Рис. 2.11 Вікно реєстрації

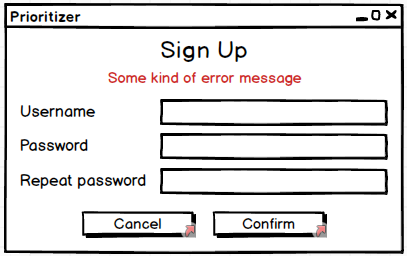


Рис. 2.12 Вікно реєстрації (помилка реєстрації)

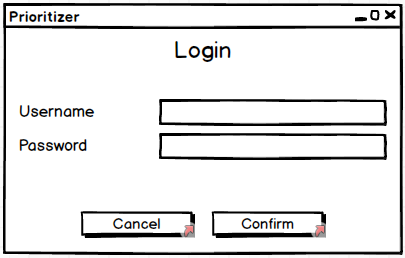


Рис. 2.13 Вікно авторизації

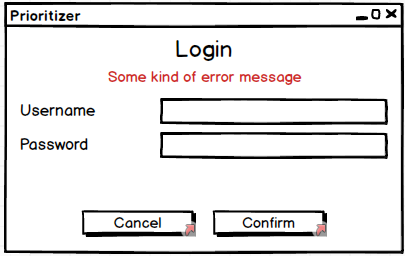


Рис. 2.14 Вікно авторизації (помилка авторизації)

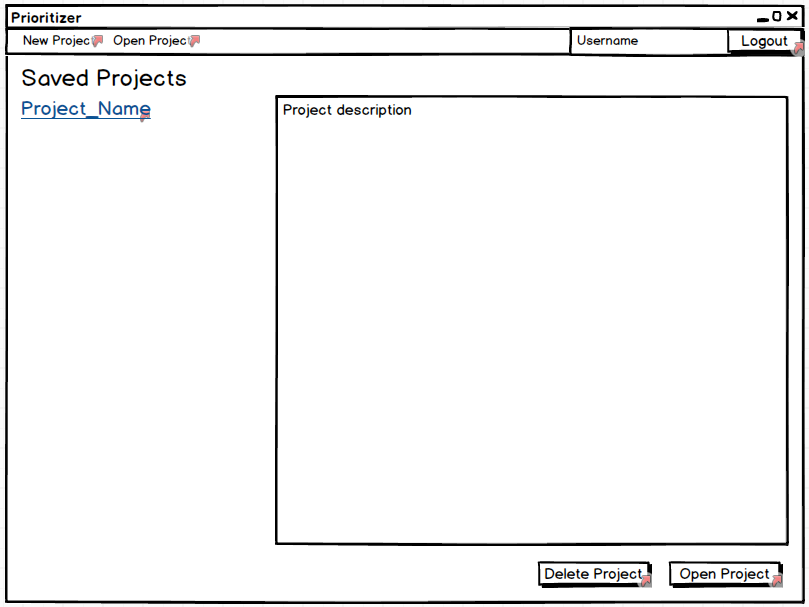


Рис. 2.15 Вікно збережених проектів

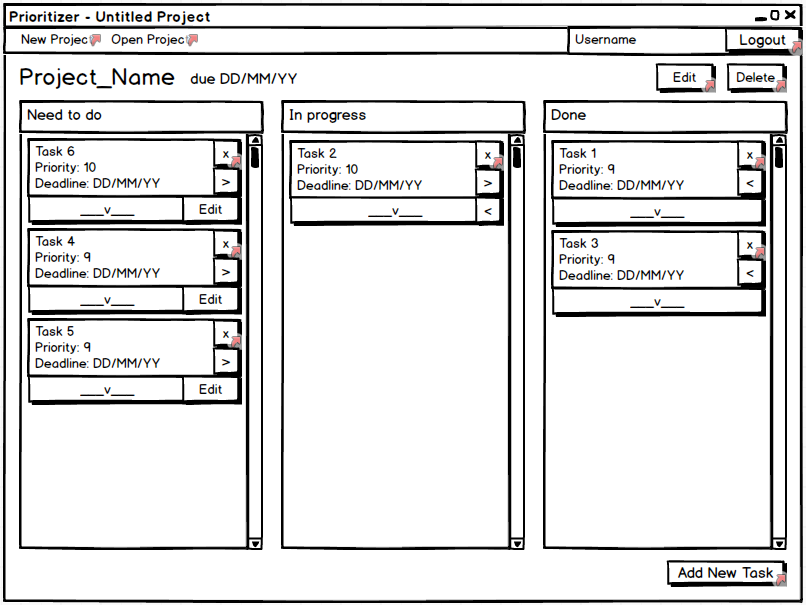


Рис. 2.16 Головне вікно програми

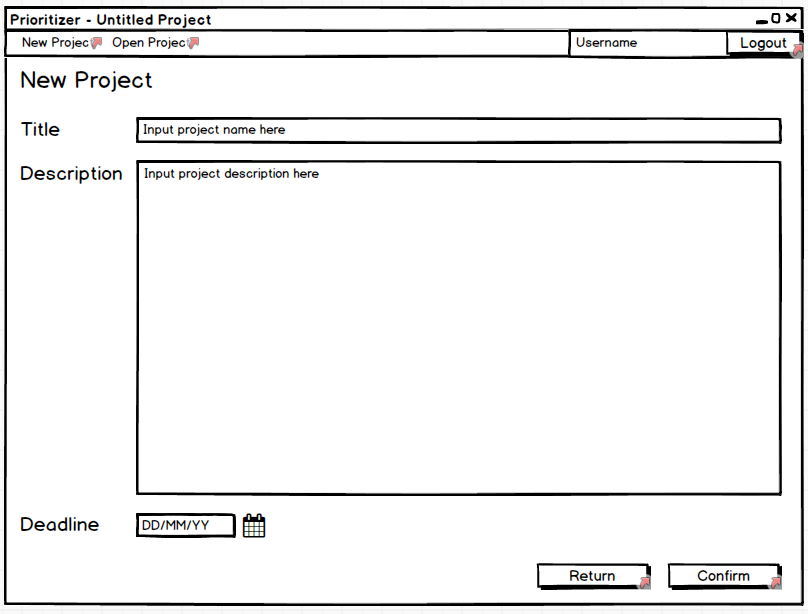


Рис. 2.17 Вікно створення нового проекту

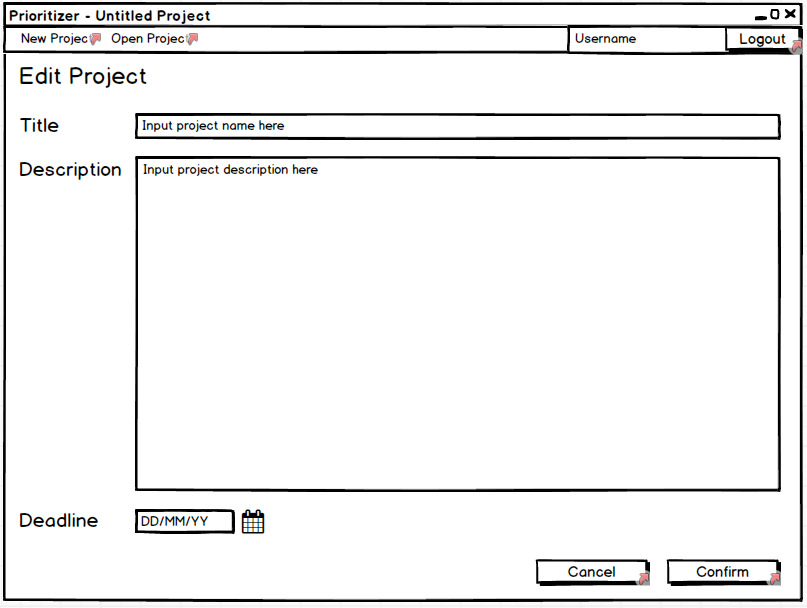


Рис. 2.18 Вікно редагування проекту

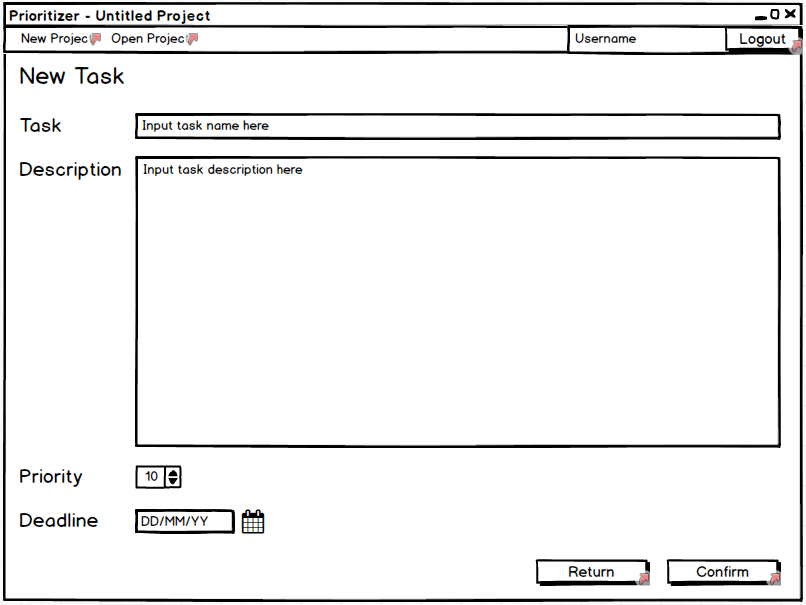


Рис. 2.19 Вікно створення нового завдання

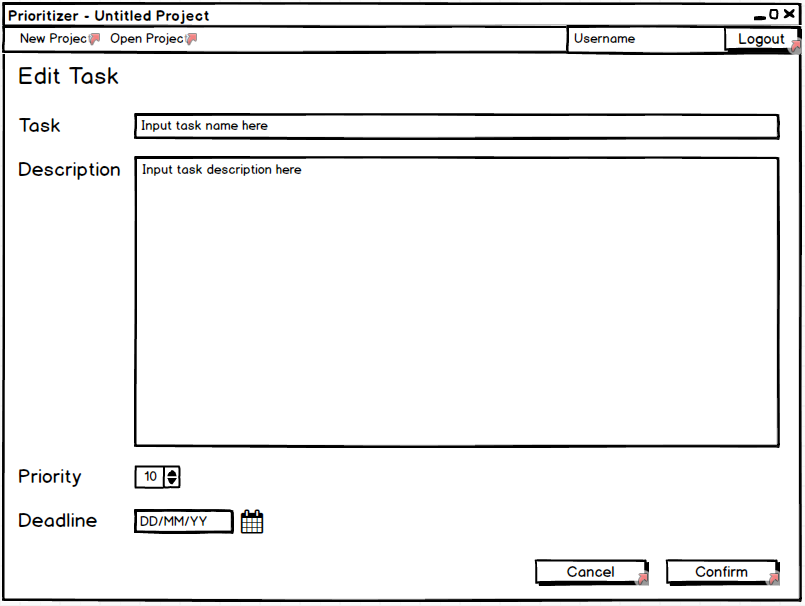


Рис. 2.20 Вікно редагування завдання

1. **ОПИС ПРОЦЕСУ РОБОТИ**

Процес роботи з програмою починається зі стартового вікна, де користувачеві дається вибір між 2 можливостями: Зареєструватися чи Авторизуватися. Для авторизації потрібен дійсний аккаунт, тому обираємо опцію «Реєстрація»:

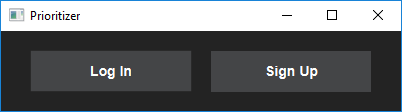


Рис. 3.1 Стартове вікно програми

У вікні реєстрації користувач має ввести логін та пароль аккаунту, якими він користуватиметься для авторизації. Для проходження процесу реєстрації користувач мусить заповнити усі текстові поля. При цьому паролі, введені у двох останніх полях, мають співпадати:

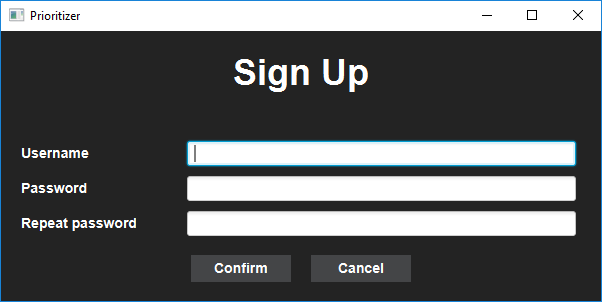


Рис. 3.2 Вікно реєстрації

Введемо дані для створення нового користувача:

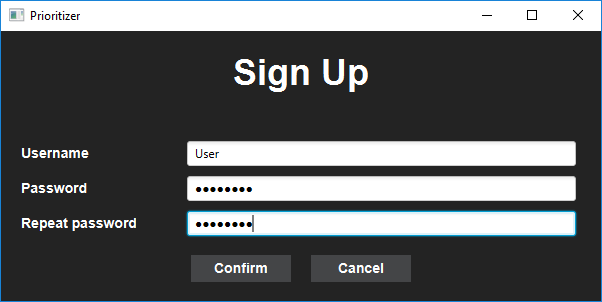


Рис. 3.3 Вікно реєстрації з введеними даними

Після успішної реєстрації користувач буде переадресований у вікно авторизації для підтвердження введених даних:

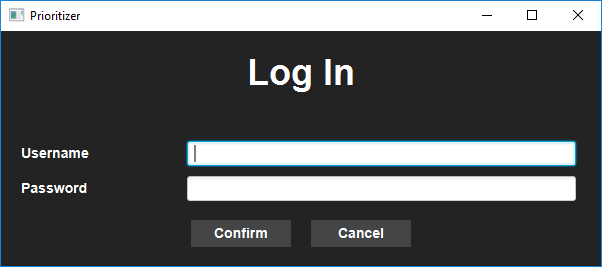


Рис. 3.4 Вікно авторизації

Для успішної авторизації необхідно заповнити обидва текстові поля, логін та пароль мають співпадати з раніше зареєстрованим аккаунтом. Підтвердимо введені раніше логін та пароль:

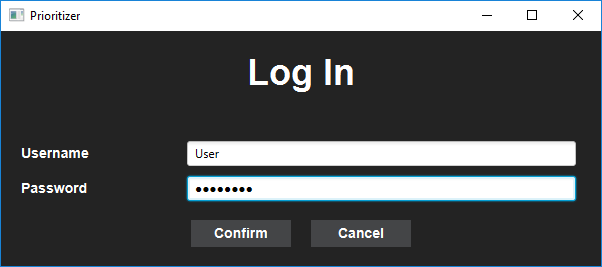


Рис. 3.5 Вікно авторизації з введеними даними

Після успішної авторизації користувач потрапляє у вікно збережених проектів. У цьому вікні користувачу доступні наступні функції:

* Створити новий проект;
* Переглянути список збережених проектів;
* Вийти з системи;
* Відкрити обраний проект;
* Видалити обраний проект.

Оскільки аккаунт був щойно створений, список збережених проектів порожній. Створимо новий проект. Для цього у верхньому меню обираємо пункт Project->New Project:

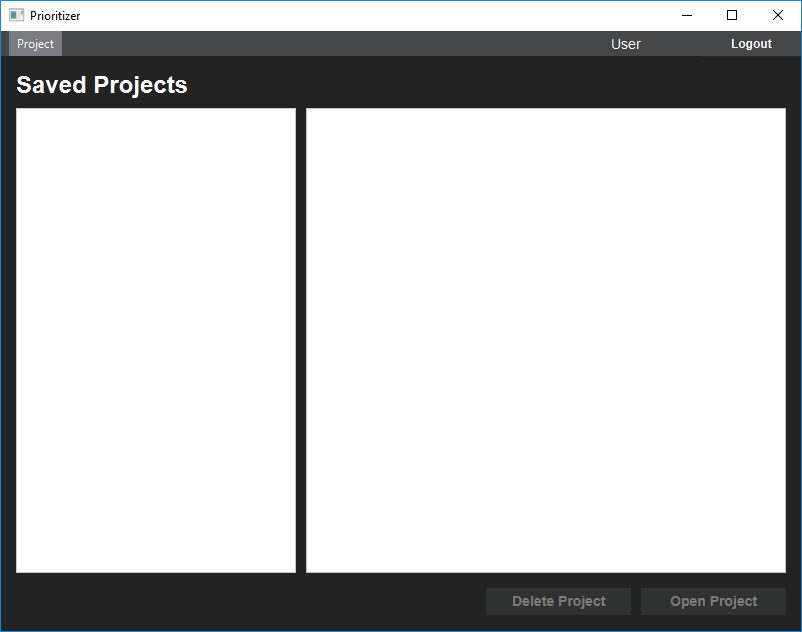


Рис. 3.6 Вікно збережених проектів

Після вибору відповідного пункту меню потрапляємо у вікно створення нового проекту:

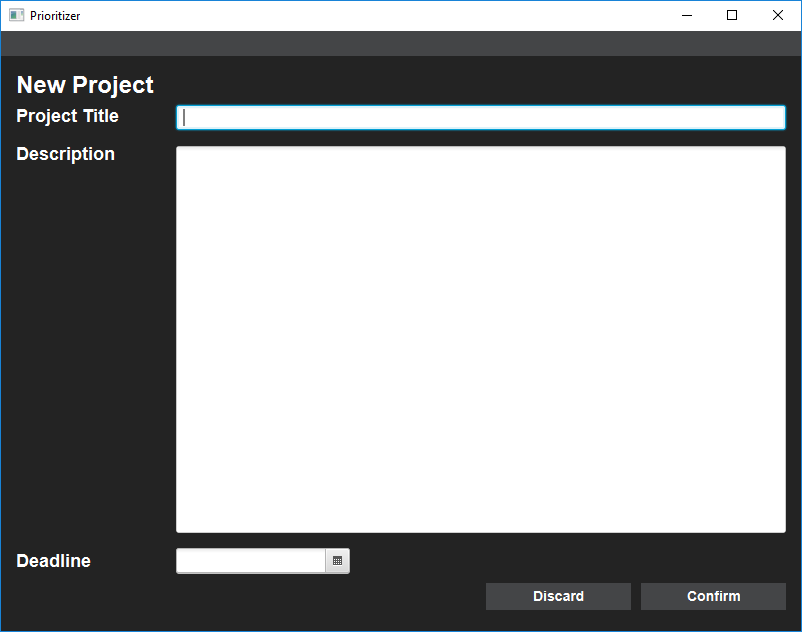


Рис. 3.7 Вікно створення нового проекту

Для успішного створення нового проекту необхідно заповнити усі поля вводу та засвідчитись, що поточний користувач не має збережених проектів з такою ж назвою. Вводимо необхідні дані для створення проекту:

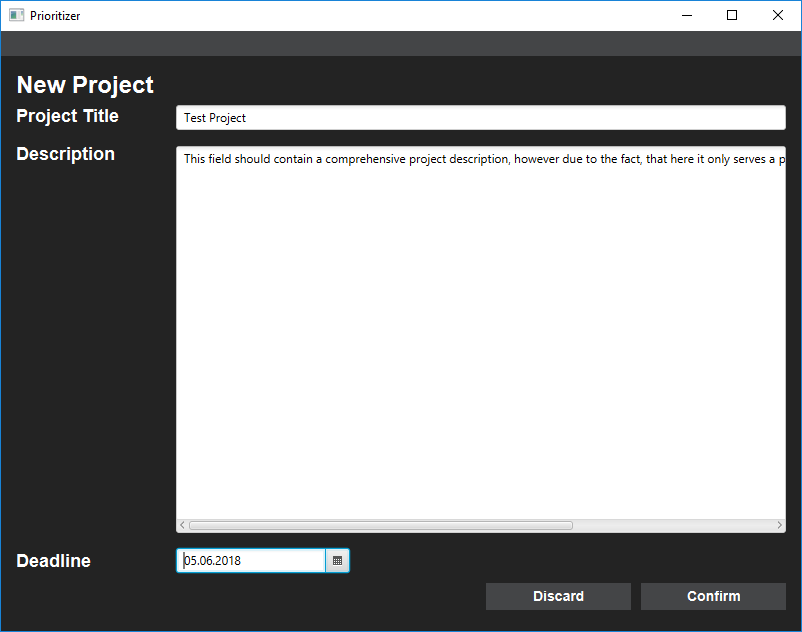
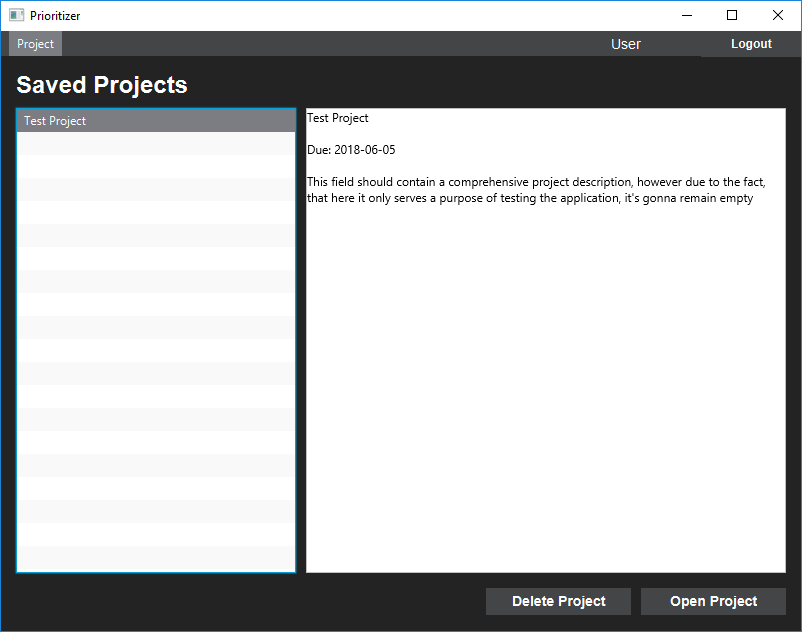


Рис. 3.8 Вікно створення нового проекту з введеними даними

Як бачимо, тепер створений проект відображається у списку збережених. Оберемо та відкриємо його:



Після вибору поточного проекту відбувається переадресація у головне вікно програми. У даному вікні маємо: назву та строк виконання проекту, клавіші видалення та редагування проекту, списки збережених завдань за статусом, клавіші для створення, редагування та видалення завдань.

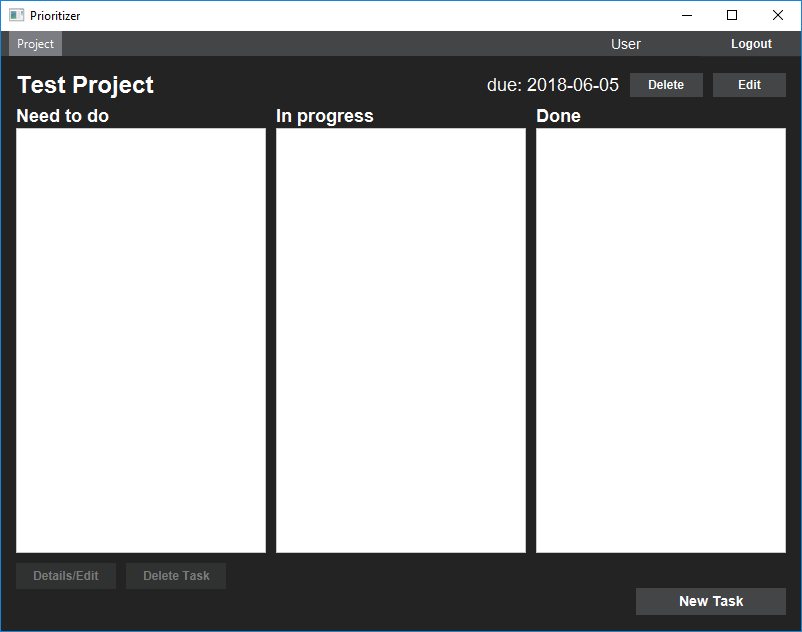


Рис. 3.9 Головне вікно програми

Створимо нове завдання. Для цього натискаємо клавішу New Task. Відбувається перехід у вікно створення завдання. Інтерфейс цього вікна практично ідентичний до вікна створення проекту з доданням графи пріоритету завдання. Натискаємо Confirm для підтвердження створення нового завдання:

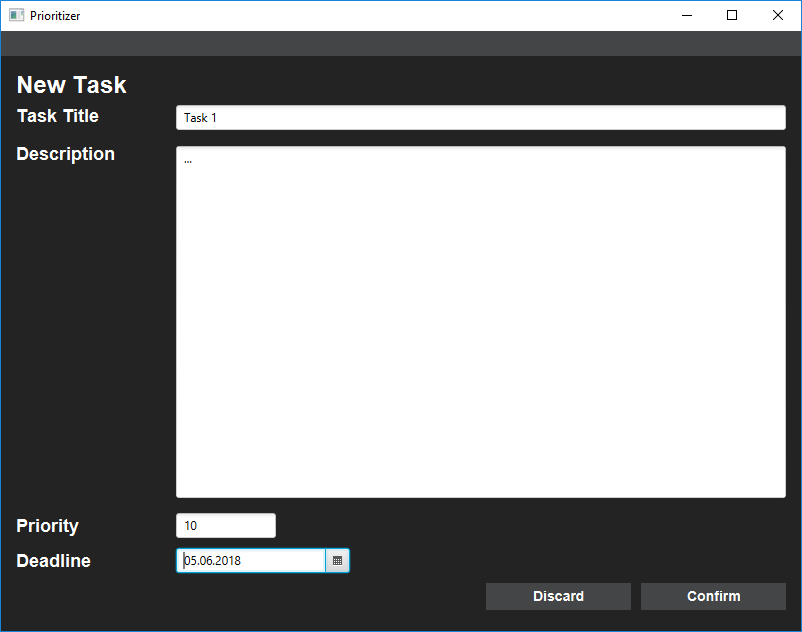


Рис. 3.10 Вікно створення нового завдання

Усі створені завдання автоматично потрапляють до списку Need to do.

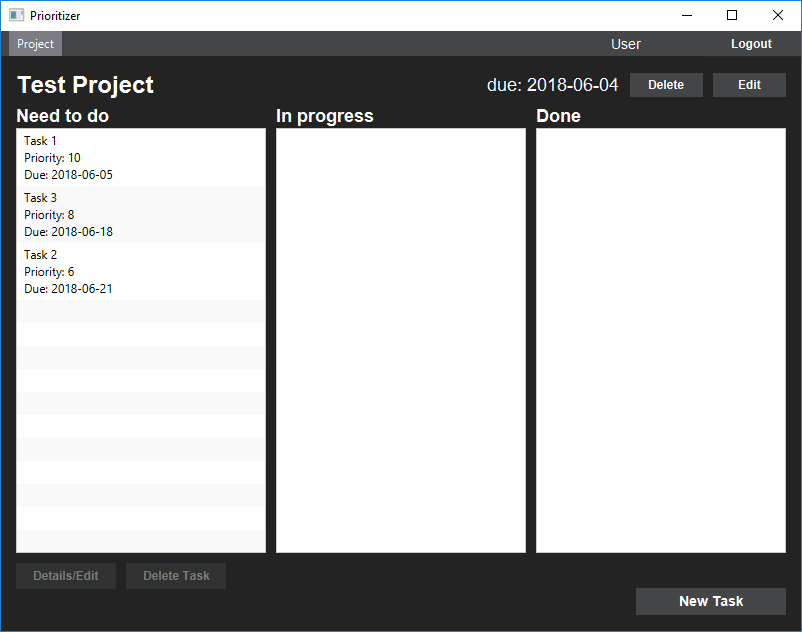


Рис. 3.11 Головне вікно програми — відображення збережених завдань

Припустимо, що ми почали виконання Task 1. Перетягнемо його в In progress. Як бачимо, тепер завдання Task 1 відображається у списку In progress.

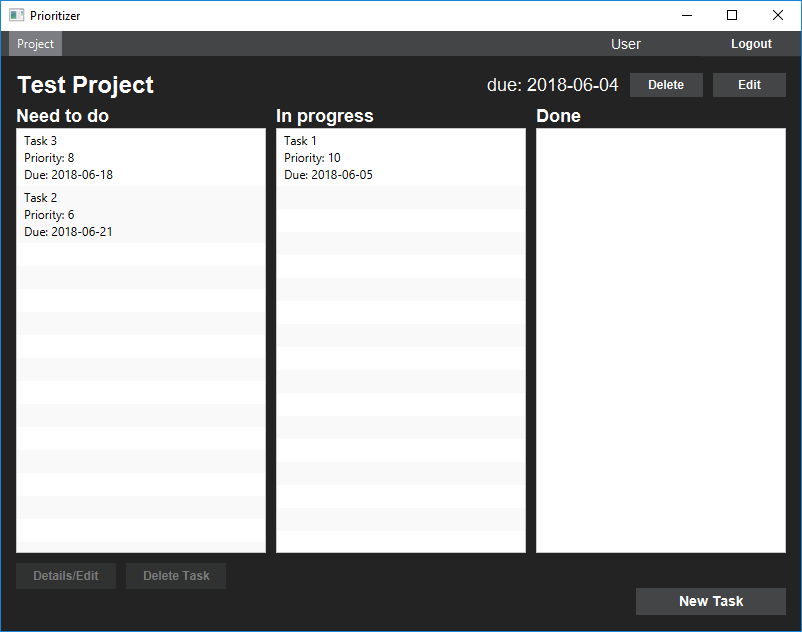


Рис. 3.12 Завдання у списку “In progress”

Припустимо, що ми виконали це завдання. Перемістимо його до списку Done.

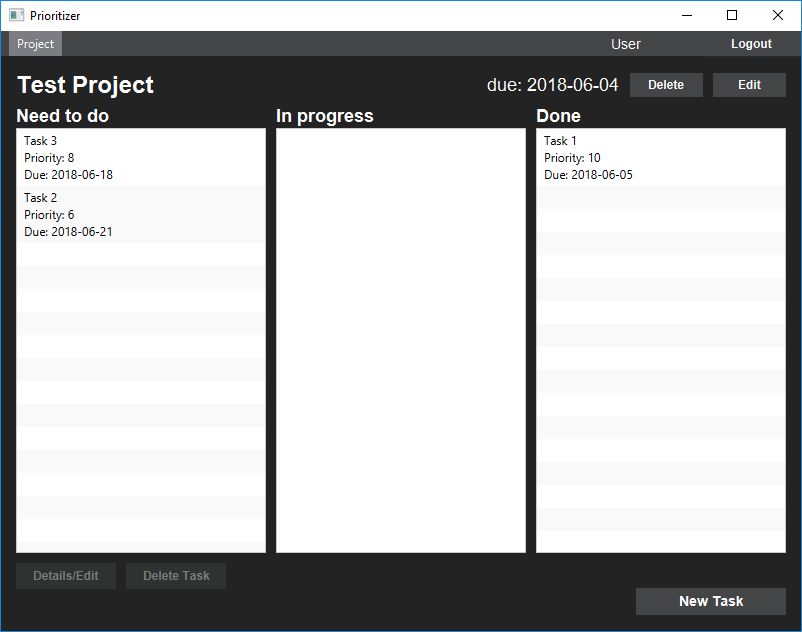


Рис. 3.13 Завдання у списку “Done”

Оскільки завдання Task 1 виконане, є сенс видалити його з проекту. Для цього вибираємо необхідне завдання та натискаємо Delete Тask:

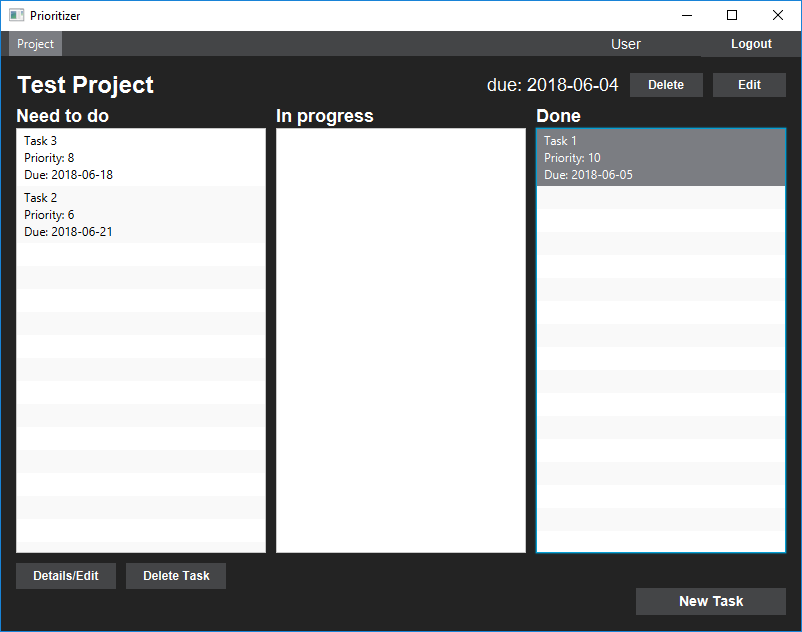


Рис. 3.14 Виділення завдання Task 1 в списку

Завдання 1 успішно видалене. Тепер спробуємо змінити дані про завдання Task 2.

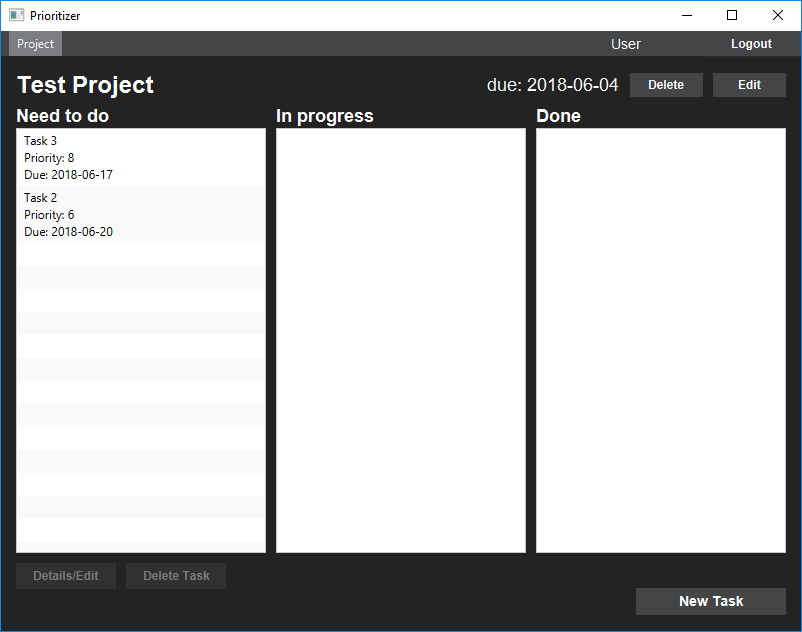


Рис. 3.14 Видалення завдання

Для цього видираємо необхідне завдання зі списку та натискаємо Details/Edit:

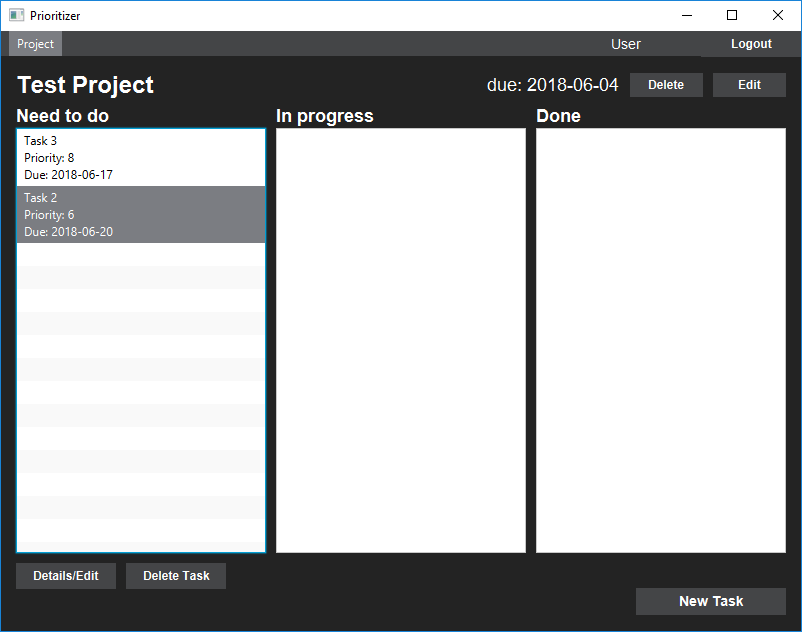


Рис. 3.15 Виділення завдання Task 2 в списку

Відбувається перехід у вікно редагування завдання. У ціому вікні ми можемо змінити усі дані про завдання: назву, опис, пріоритет, термін виконання:

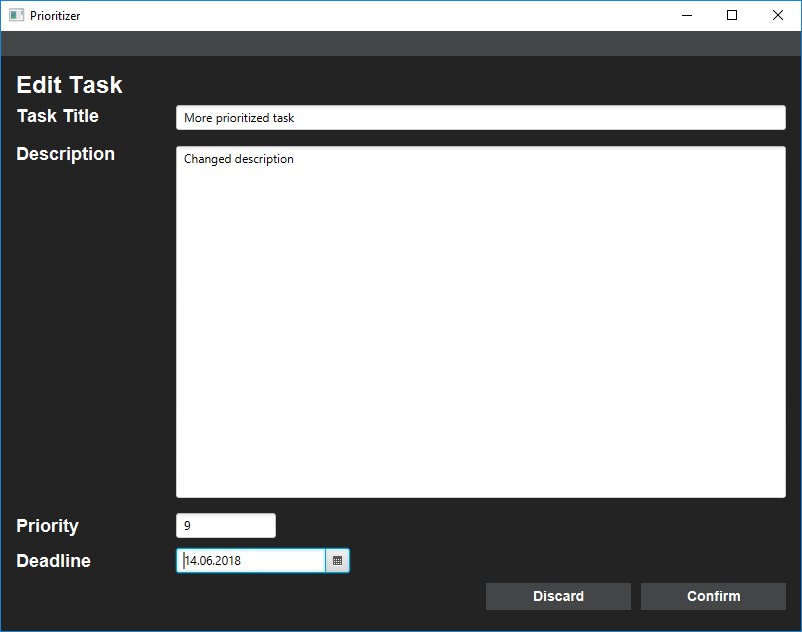


Рис. 3.16 Вікно редагування завдань

Підтвердимо внесені зміни, натиснувши Confirm. Як бачимо, тепер відредаговане завдання відображається у списку з новими даними.

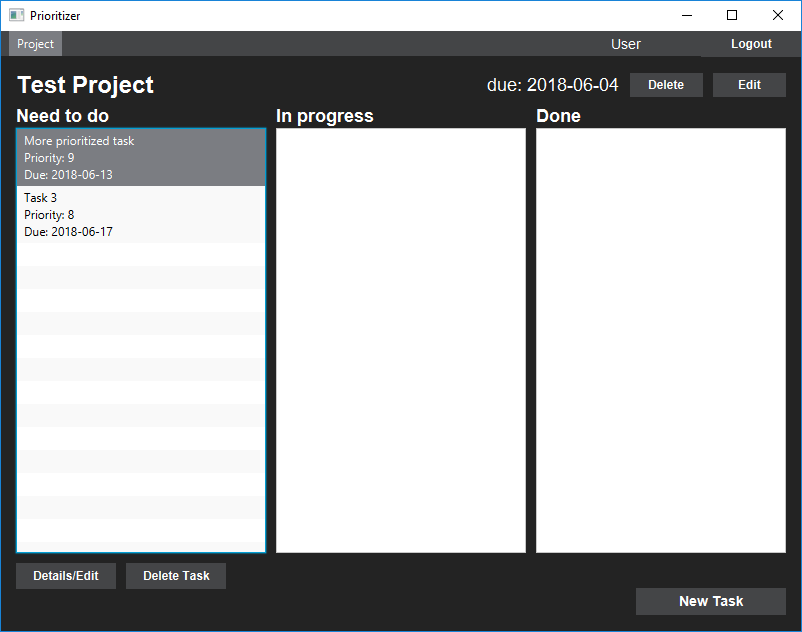


Рис. 3.17 Відредаговане завдання в списку

Спробуємо змінити інформацію про проект. Для цього натискаємо клавішу Edit навпроти назви проекту. Відбувається переадресація у вікно редагування проекту, де користувач може змінювати будь-яку інформацію про проект:

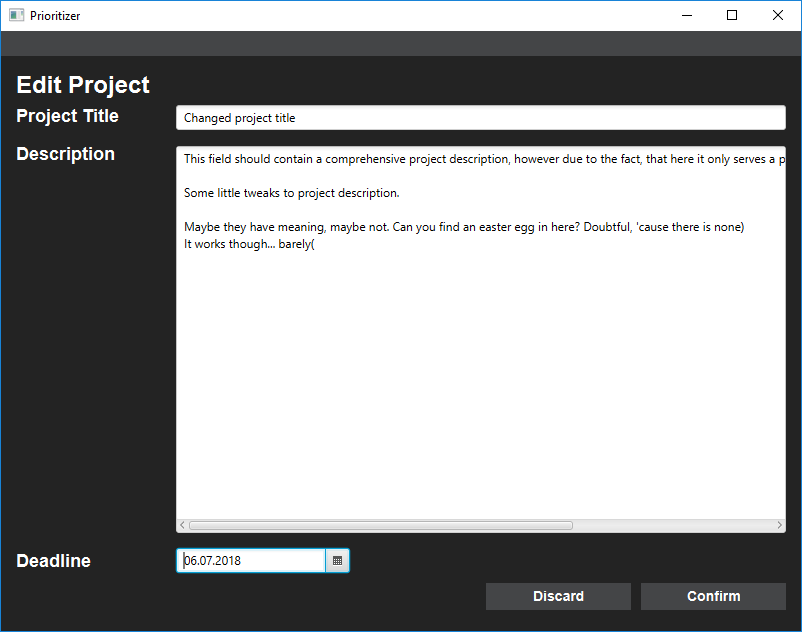


Рис. 3.18 Вікно редагування проектів

Перевіримо внесені зміни у вікні Збережених проектів (Project->Saved Projects):

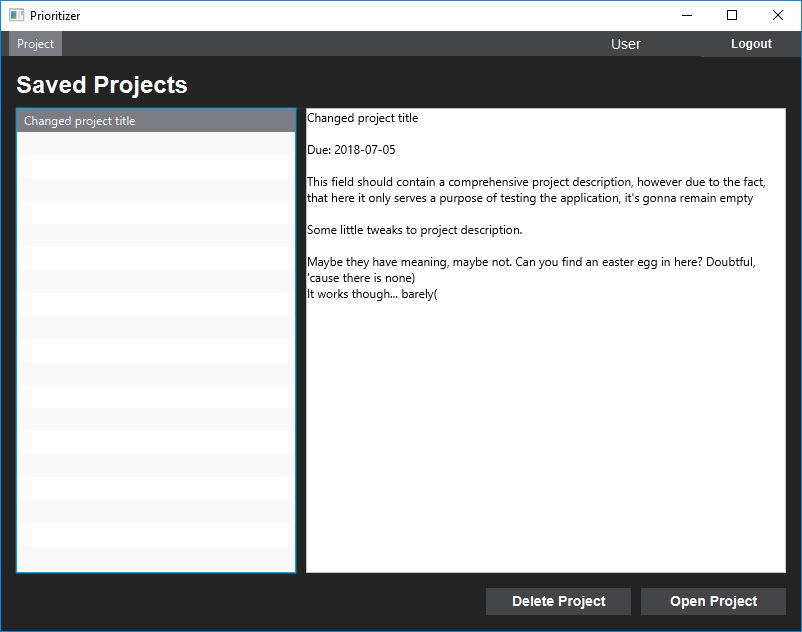


Рис. 3.19 Відредагований проект у списку збережених

Для видалення проекту вибираємо його у списку збережених проектів та натискаємо Delete Project. Для виходу з аккаунту та завершення сесії роботи натискаємо Logout.

**ВИСНОВКИ**

В ході виконання курсової роботи був розроблений програмний додаток для індивідуального планування робочого процесу на мові програмування Java.

За методологією Scrum складений беклог продукту з преліком фуекцій додатку, які мали бути реалізовані. У процесі проектування програми створені UML діагарми прецедентів, діяльності та класів для більш детального опису функціоналу програмного продукту. Обраний паттерн розробки MVC. Був розроблений прототип графічного інтерфейсу для реалізації представлення. Для реалізації паттерну MVC використаний фреймворк JavaFX. Для організації зв’язку з БД застосований плагін JDBC. Сама база даних спроектована та реалізована за допомогою СУБД mySQL. Для збірки проекту використаний інструмент Maven.

Функції програми, визначені на першому етапі проектування, у повній мірі реалізовані. До функціоналу додатку входить:

* Реєстрація користувача;
* Авторизація зареєстрованого користувача;
* Створення проекту;
* Перегляд збережених проектів;
* Редагування проекту;
* Видалення проекту;
* Створення завдання;
* Сортування завдань за пріоритетністю;
* Зміна статусу завдань за прогресом виконання;
* Редагування завдань;
* Видалення завдань;

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

* Herbert Schild “Java: A Beginner’s Guide. 7th edition” – 2017;
* Carl Dea “JavaFX 2.0: Introduction by Example” – 2011;
* Timur Mashnin “Java^ Graphical User Interface for Desktop and Web” – 2013;
* Maven Documentation - <https://maven.apache.org/guides/>;
* JDBC Documentation – https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/;

**ДОДАТОК А. ВИХІДНИЙ КОД РЕАЛІЗАЦІЇ**

Вихідний код реалізації доступний за посиланням:

<https://github.com/mrSurvivorr/Semester_Project>