МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра Програмної інженерії

Звіт
з лабораторної роботи №3
з дисципліни: «Проектний практикум»
з теми: «Проектування та розробка проекту. Метод мозкового штурму. Структурні діаграми: діаграма класів, об'єктів»

Виконали:
ст. гр. ПЗПІ-23-2
Ситник Є. С.
Малишкін. А. С.
Краснокутська Ю. Є.
Семьонов. О. О.
Петах С. І.

Перевірили:
ст. викладач кафедри ПІ,
Онищенко К. Г.,
доц. каф. ПІ,
Афанасьєва І. В.,

Харків – 2025

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОЕКТУ. МЕТОД МОЗКОВОГО ШТУРМУ. СТРУКТУРНІ ДІАГРАМИ: ДІАГРАМА КЛАСІВ, ОБ'ЄКТІВ

3.1 Мета роботи

Отримати навички побудови статичного представлення логічної моделі проектованої інформаційної системи у вигляді CRC-карток, об’єктної моделі, діаграми класів UML.

3.2 Хід роботи

3.2.1 Створення Smart Use Case діаграми

SMART Use Case Diagram це один з інструментів моделювання в UML, що візуально відображає функціональні вимоги до системи з точки зору користувача. Вона складається з акторів, варіантів використання та зв’язків між ними, демонструючи, хто і як використовує систему для досягнення певних цілей. Ця діаграма допомагає розробникам, замовникам та іншим зацікавленим сторонам краще зрозуміти обсяг та призначення системи, сприяючи ефективній комунікації та збору вимог на ранніх етапах розробки програмного забезпечення.

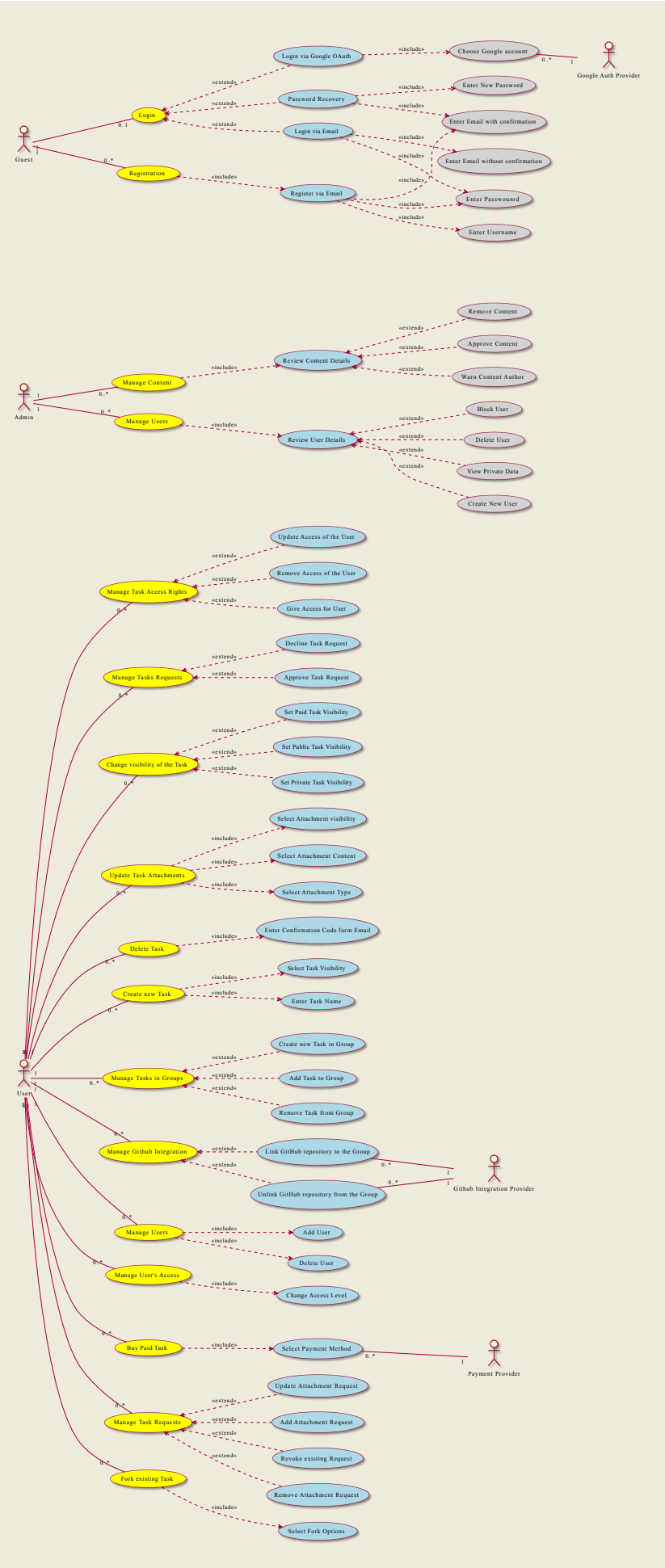

Рисунок 3.1 – SMART Use Case діаграма

3.2.2 Проєктування системи методом мозкового штурму (CRC-картки)

CRC-картки це інструмент для об’єктно-орієнтованого дизайну, що використовується переважно на початкових етапах розробки програмного забезпечення. Вони допомагають команді швидко обговорювати та визначати, які класи потрібні системі, за що відповідає кожен клас, і з якими іншими класами він взаємодіє для виконання своїх завдань.

Під час мозкового штурму нами було виділено картки із наступними класами: User, Task, Attachment, Solution, TaskGroup, Comment, Payment, Subscription

User	
subclasses:	
superclasses:	
responsibilities	collaborators
Authenticate user credentials	*Task*
Manage user profile information	*Solution*
Track subscription status and limits	*Attachment*
Maintain user activity history	*Payment*
Handle user preferences and settings	*Subscription*

User
Description:
Represents a platform user with authentication, profile management, and subscription capabilities
Attributes:
email
username
password
profileInfo

Task	
subclasses:	
superclasses:	
responsibilities	collaborators
Store task information and metadata	*User*
Manage task access permissions	*Solution*
Handle task pricing and monetization	*TaskGroup*
Track task completion status	*Comment*
Maintain task versioning and history	*Payment*

Task
Description:
Core entity representing a task with description, attachments, access controls, and monetization options
Attributes:
title
owner
tags
attachments

Attachment	
subclasses:	
superclasses:	
responsibilities	collaborators
Enforce storage limits per subscription tier	*Task*
Handle file upload and download	*Solution*
Support multiple formats	*User*
Generate secure URLs	*Subscription*

Attachment
Description:
Manages attachments for tasks and solutions with storage limits and security controls
Attributes:
creator
target
type
data

Solution	
subclasses:	
superclasses:	
responsibilities	collaborators
Store solution content and attachments	*User*
Track solution submission status	*Task*
Handle solution evaluation and rating	*Comment*
Manage solution visibility settings	*Rating*
Support various solution formats	*Attachment*

Solution
Description:
Represents user-submitted solutions to tasks with various formats and evaluation capabilities
Attributes:
task
owner
content
attachments

TaskGroup	
subclasses:	
superclasses:	
responsibilities	collaborators
Group related tasks together	*Task*
Manage group-level access permissions	*User*
Handle group monetization settings	*Payment*
Integrate with GitHub repositories	
Support bulk operations on tasks	

TaskGroup
Description:
Organizes related tasks into thematic or project-based collections with shared access settings
Attributes:
name
description
owner
tasks

Comment	
subclasses:	
superclasses:	
responsibilities	collaborators
Store comment content and metadata	*User*
Track comment timestamps	*Task*
Handle comment moderation	*Solution*

Comment
Description:
Enables user interaction through comments on tasks with rating capabilities
Attributes:
user
target
content

Payment	
subclasses:	
superclasses:	
responsibilities	collaborators
Process payment transactions	*User*
Validate payment information	*Task*
Handle payment failures and retries	*TaskGroup*
Track payment history	*Subscription*
Integrate with payment providers	

Payment
Description:
Handles financial transactions for task access, solutions, and subscriptions
Attributes:
user
target
amount
currency

Subscription	
subclasses:	
superclasses:	
responsibilities	collaborators
Manage subscription lifecycle	*User*
Enforce subscription limits	*Payment*
Handle subscription upgrades/downgrades	
Track usage against limits	
Process subscription renewals	

Subscription
Description:
Manages user subscription tiers, limits, and billing cycles
Attributes:
user
tier
startDate
endDate

3.2.3 Створення діаграми класів

Діаграма класів показує класи, їхні атрибути та операції а також зв’язки між ними. Це основна діаграма, що визначає архітектуру системи та взаємодію її компонентів.

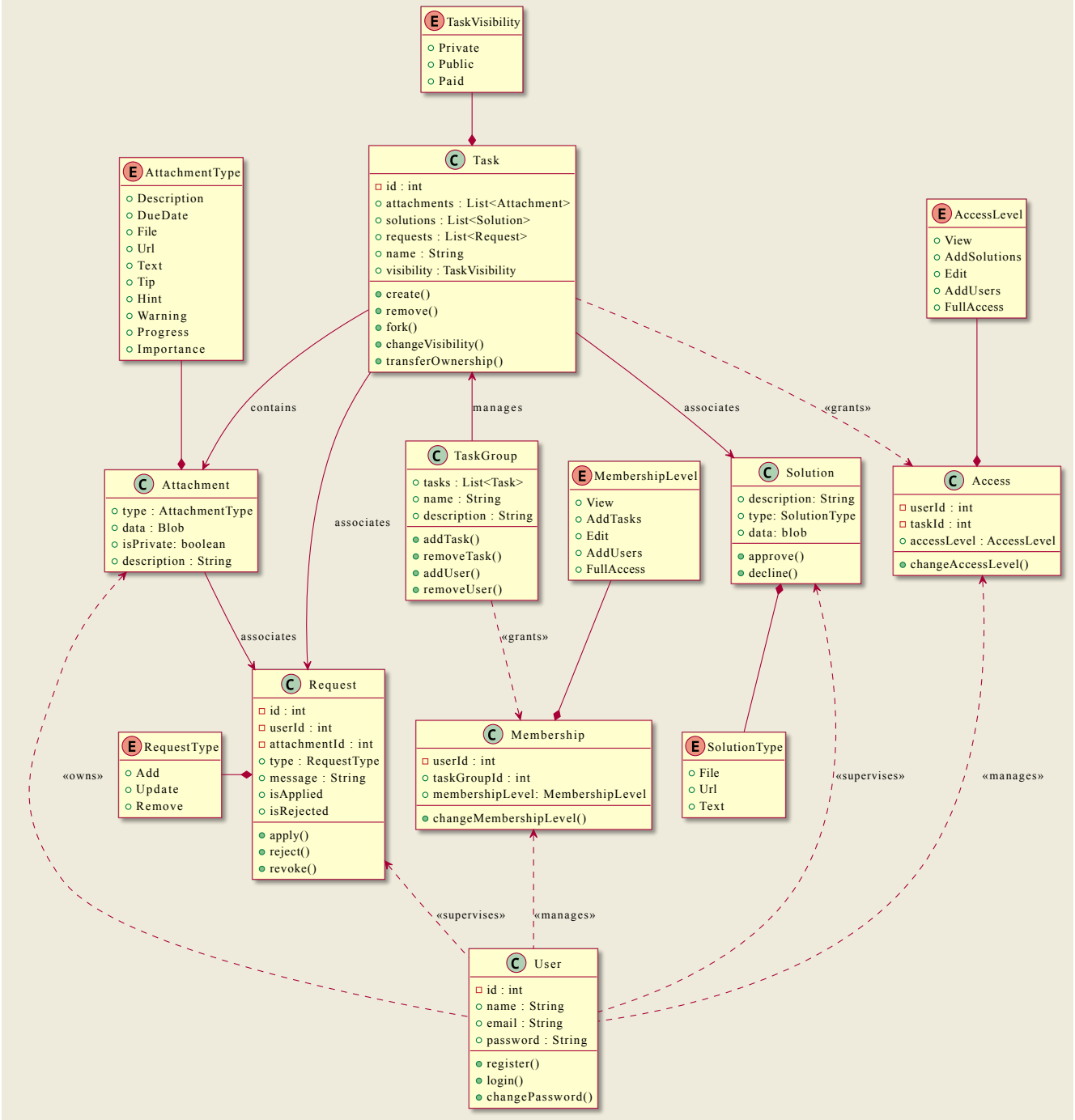

Рисунок 3.2 – Діаграма класів

3.2.4 Створення діаграми об’єктів

Діаграма об’єктів є миттєвим станом системи в конкретний момент часу, що демонструє конкретні екземпляри класів (об’єкти) та їхні зв’язки. На відміну від діаграми класів, яка показує загальну структуру та можливі зв’язки, діаграма об’єктів відображає реальні об’єкти з їхніми поточними значеннями атрибутів і фактичними зв’язками між ними. Вона використовується для візуалізації конкретних сценаріїв виконання та допомагає перевірити правильність діаграми класів.

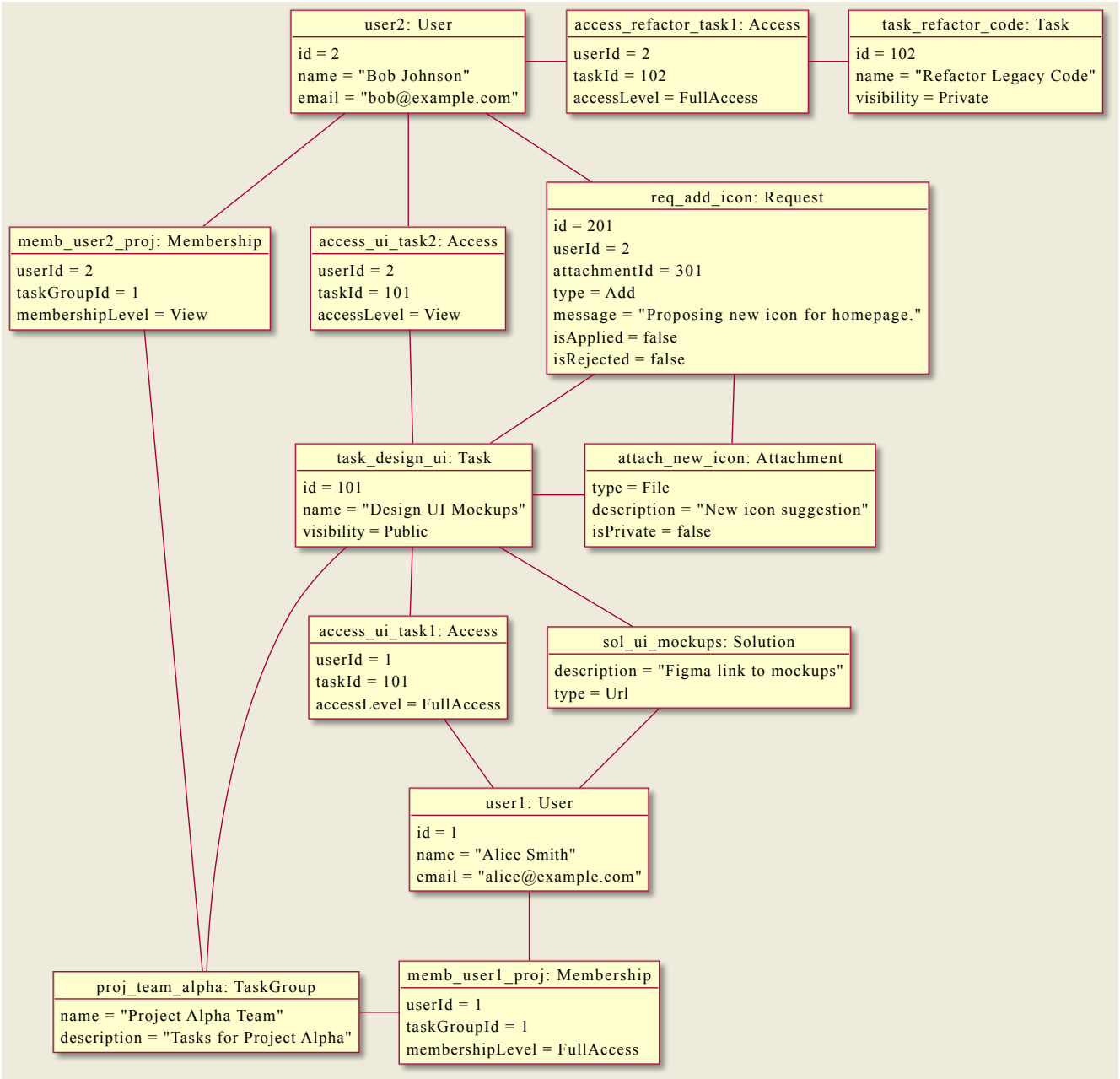

Рисунок 3.3 – Діаграма об’єктів

3.3 Висновки

За результатами цієї лабораторної роботи ми отримали навички побудови статичного представлення логічної моделі проектованої інформаційної системи у вигляді CRC-карток, об’єктної моделі та діаграми класів UML.