

2022 p.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Системотехніки

Спеціальність Комп'ютерні науки та інформаційні технології

Курс 2 група КНТ-20-1 семестр 4

ЗАВДАННЯ НА КУРСОВИЙ ПРОЕКТ

студентові Голуб Дмитру Костянтиновичу

(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна система розподілу робіт між виконавцями

2. Термін здачі студентом закінченої роботи 20.06.22

3. Вихідні дані до проекту Розробити клієнтську та серверну частину інформаційної системи, для серверної частини буде використано СУБД MySQL та Apache веб сервер, клієнтська частина має реалізовувати наступні функції: для усіх користувачів: вхід у систему (заглушка), форма зворотного зв'язку, перегляд загальної інформації; для зареєстрованого користувача (керівника): створення завдань та вказування виконавців, перегляд надісланих виконавцем звітів, прийняття або відхилення звітів, коментування звітів; для зареєстрованого користувача (виконавця): перегляд отриманих завдань, надсилання звіту, перегляд коментарів керівника

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці) Проаналізувати предметну область та виділити необхідні для автоматизації бізнес-процеси, сформулювати вимоги до ІС[1], провести функціональне моделювання та уточнити вимоги до ІС, провести логічне та фізичне моделювання БД, реалізувати фізичну модель БД, реалізувати необхідні бізнес-процеси ІС, відповідно до ГОСТ 34.69890 розробити «Керівництво користувача», підготувати відповідно до ГОСТ 19.401-78 програмний документ «Тест програми»[2]

5. Перелік графічного матеріалу (з точним визначенням обов'язкових креслень): функціональна модель ІС згідно стандарту IDEF0; ER-діаграми логічної моделі згідно до IDEF1X; UML діаграми

6. Дата видачі завдання: 12.03.22

Керівник роботи _____

проф. Міщеряков Ю.В.

(підпис)

(прізвище, ім'я, по батькові)

Студент _____




Голуб Д. К.

(підпис)

(прізвище, ім'я, по батькові)

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області	12.03.22-28.03.22	
2	Визначення основних бізнес-функцій інформаційної системи	28.03.21-5.04.22	
3	Визначення функцій інтерфейсу клієнтської частини інформаційної системи	5.04.22-11.04.22	
4	Логічне і фізичне моделювання даних, створення баз даних	11.04.22-17.04.22	
5	Розробка серверної частини інформаційної системи	17.04.22-3.05.22	
6	Розробка програмного забезпечення клієнтської частини інформаційної системи	3.05.22-21.05.22	
7	Тестування розробленого програмного забезпечення	21.05.22-4.06.22	
8	Підготовка пояснювальної записки та її додатків	4.06.22-20.06.22	
9	Захист курсової роботи	20.06.22	

Студент _____  _____ Голуб Д. К.
 (підпис) (посада, прізвище, ініціали)

Керівник роботи (проекту) _____ проф. Міщеряков Ю.В.
 (підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Робота містить: 122 сторінки, 19 рисунків, 5 таблиць, 3 додатки, 5 джерел.

БАЗА ДАНИХ, РОЗПОДІЛ ЗАДАЧ, ВЕБ ЗАСТОСУНОК, СИСТЕМА УПРАВЛІННЯ БАЗАМИ ДАНИХ, ІНФОРМАЦІЙНА СИСТЕМА, ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ.

Предметна область цього проекту – задача розробки системи розподілення задач між виконавцями.

Об'єкт дослідження – автоматизація електронного документообігу компанії щодо розподілення задач та контролю статусу їх виконання, отримання звітів, щодо виконання задачі, передача виправлень виконавцю.

Предмет дослідження – програмні засоби та методології, призначені для проектування, розробки, створення та підтримки серверних та клієнтських частин інформаційної системи розподілу робіт між виконавцями.

Мета курсової роботи – розробка клієнтської та серверної частини інформаційної системи розподілу завдань між виконавцями.

Методи досліджень – набір інструментів та методів програмної інженерії для проектування програмного забезпечення: CASE-засоби, методології функціонального та об'єктно-орієнтованого проектування, розробки реляційних баз даних та проектування веб-додатку за допомогою UML-моделювання.

Для опису предметної області було розроблено функціональну модель даних за допомогою методології IDEF0, використовуючи програмний продукт AllFusion Process Modeler.

Логічну модель даних представлено у вигляді ER-діаграми, фізичну модель даних реалізовано на основі логічної моделі за допомогою системи управління базами даних MySQL 5.7.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ,
ТЕРМІНІВ

URL	– uniform resource locator.
IC	– інформаційна система
БД	– база даних
СУБД	– система управління базами даних
SQL	– Structured Query Language
IDEF0	– нотація опису бізнес-процесів
IDEF1X	– методологія розробки моделей даних
UML	– Unified Modeling Language

ЗМІСТ

ВСТУП.....	7
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	8
1.1 Аналіз предметної області.....	8
1.2 Визначення основних бізнес-процесів та ролей користувачів	8
1.3 Постановку задачі.....	9
2 ВИМОГИ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ	11
2.1 Визначення вимог до ІС.....	11
2.2 Розробка функціональної моделі	12
2.3 Фізичне моделювання бази даних	17
2.4 UML Моделювання ІС	18
2.4.1 Діаграма прецедентів (Use Case Diagram).....	18
2.4.2 Діаграма послідовності (Sequence Diagram).....	19
2.4.3 Діаграма класів (Class Diagram).....	23
2.4.4 Діаграма активності (Activity Diagram)	24
2.4.5 Діаграма станів (State Machine Diagram) — це модель поведінки єдиного об'єкта, що вказує зміну станів об'єкта у часі.	26
2.4.6 Розробка вимог до функції інформаційної системи	27
2.4.7 Розробка вимог до клієнтської частини інформаційної системи...	27
3 ОПИС ПРИЙНЯТИХ ПРОЕКТНИХ РІШЕНЬ	28
3.1 Обґрунтування вибору СУБД, мови програмування та тестового серверу.....	28
3.2 Створення бази даних	29
Висновки	30
ДОДАТОК А Керівництво користувача	32
ДОДАТОК Б VISION AND SCOPE.....	39
ДОДАТОК В Текст програми	48

ВСТУП

Сьогодні, ефективність функціонування компанії, насамперед, залежить від ефективності комунікацій між замовником та працівником, якщо Ваша компанія має невеликий, обмежений штат працівників, комунікація роботодавця з виконавцями може не потребувати якихось складних систем та може бути навіть вербальною, але, якщо штат працівників достатньо великий, організація продуктивної роботи може бути дуже складною.

Насамперед, важливо чітко сформулювати задачу та донести її до кожного конкретного виконавця, отримати звіт з виконаної роботи, якщо звіт потребує коригувань, передати їх виконавцеві, отримати виправлений звіт, перевірити його, та або знову затребувати правки, або прийняти роботу виконаною. У випадку, якщо контроль завдань несистематизований, виконавець може або зовсім не отримати завдання, або неправильно його зрозуміти, звіт може загубитися та не дійти до замовника.

З розвитком комп'ютерних технологій, ми маємо можливість автоматизувати систематизацію розподілення задач та ввести електронний документообіг, що покращить комунікацію між замовником та виконавцем, дасть можливість чітко сформулювати завдання персонально для кожного виконавця, що зменшить ризик некоректного його сприйняття, знизить до мінімуму ризик «зникання» звіту та дозволить економити час як замовника, так і працівника, знизивши необхідний обсяг очних переговорів.

Через розповсюдження коронавірусної загрози, багато працівників вимушені працювати віддалено, тож система електронного документообігу дозволить таким працівникам зручно здавати звіти у електронному вигляді та отримувати зворотній зв'язок від замовника.

Об'єктом курсової роботи є електронна система розподілу задач між виконавцями, що підвищує ефективність роботи компаній, які використовують систему. Актуальність даної теми пояснюється необхідністю обробки та накопичення великих обсягів даних, спрощенням та покращенням ефективності взаємодії замовника з виконавцем.

Результатом курсового проекту повинна стати інформаційна система розподілення задач між виконавцями «Worker».

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Аналіз предметної області є важливим етапом для визначення основних напрямків проектування інформаційної системи, що задовольняє потреби користувачів.

Предметною областю курсового проекту є організація процесу взаємодії замовника та виконавця. Організацію бізнес-процесу розроблюваної системи можна представити об'єктами-сутностями: замовник, виконавець, задача, звіт, коментар до звіту.

Основною задачею розроблюваного сервісу є процес обліку завдань, що включає у себе процеси розподілу завдань між виконавцями, надсилання звіту, надсилання правок у формі коментарів, підтвердження виконання задачі в інтерактивному режимі.

Щоб додати завдання користувачеві зі статусом виконавець, користувач зі статусом замовник повинен вказати заголовок задачі, виконавця та описати задачу, нова задача буде відображатися в особистих кабінетах обох користувачів.

Коли користувач зі статусом виконавець отримав задачу, він виконує її та складає звіт щодо виконання задачі, виконавець додає звіт до задачі і тоді замовник може або прийняти виконану задачу, виконана задача автоматично перенесеться у архів, функції додавання та коментування звітів, управління статусом задачі будуть недоступні, або додати коментар до звіту та затребувати правки. Після того, як звіт повністю задовільнив керівника, він може помітити задачу виконаною, тоді задачу буде переміщено у архів та будь яке її редагування буде недоступно.

1.2 Визначення основних бізнес-процесів та ролей користувачів

Метою створення інформаційної системи “Worker” (електронна система розподілу задач між виконавцями) є автоматизація бізнес-процесів з метою спрощення взаємодії між керівником (замовником) та виконавцем. Основним

бізнес-процесом ІС є процес обліку задач та звітів, що представляє певну послідовність дій з боку обох вищенаведених користувачів.

Основний процес, що буде автоматизовано – облік задач та звітів, складається з наступних етапів: додавання нової задачі замовником; отримання задачі виконавцем та виконання її, надсилання звіту до керівника; узгодження виправлень з керівником з використанням системи коментарів; перенесення задачі в архів у випадку, якщо звіт задовольняє вимогам замовника.

Загалом можна виділити такі типи користувачів інформаційної системи, що можуть взаємодіяти із розроблюваним сервісом:

- Незареєстрований користувач (гість) – користувач, якому надаються оглядові можливості веб-застосунка. Такий користувач може дивитися загальну інформацію на головній сторінці, увійти у систему, створювати нові записи у вкладці «зворотній зв'язок». Усі зареєстровані користувачі системи також мають ці можливості.

- Зареєстрований користувач (керівник) – авторизований користувач системи, що може створювати нові завдання, переглядати звіти виконавців, переглядати список задач, які були створені для його облікового запису, змінювати статус задачі (виконано, потребує правок), коментувати звіти виконавців, переглядати звіти виконавців.

- Зареєстрований користувач (виконавець) – авторизований користувач системи, що має можливість переглянути усі задачі, що були створені для його облікового запису, додавати звіти з виконаної задачі, переглядати власні звіти, переглядати коментарі замовника, переглядати статус задачі.

1.3 Постановку задачі

До меж предметної області розроблюваної ІС входить такий функціонал: функції усіх користувачів: Вхід у систему (+ відновлення облікового запису), форма зворотнього зв'язку, перегляд загальної інформації; для авторизованих користувачів: керування власним аккаунтом, перегляд профілів інших користувачів; для керівника: створення завдання та казування виконавця, перегляд звітів, прийняття або відхилення звіту, коментування звіту, розглядання заявок зворотнього зв'язку; для виконавця: перегляд та сортування задач, позначання задачі як прийнятої до виконання, надсилання звіту, перегляд

коментарів; для адміністратора: створення нових облікових записів, управління існуючими обліковими записами;

Розроблювана ІС повинна реалізувати наступний функціонал: функції усіх користувачів: вхід у систему (заглушка), форма зворотнього зв'язку, перегляд загальної інформації; функції, що доступні керівнику: створення завдань та вказування виконавців, перегляд надісланих виконавцем звітів, прийняття або відхилення задачі, коментування звітів; функції, що доступні виконавцю: перегляд отриманих завдань, надсилання звіту про виконане завдання, перегляд коментарів керівника.

Для виконання курсового проекту, результатом якого стане інформаційна система розподілу завдань між виконавцями “Worker” необхідно виконати наступні завдання:

- провести аналіз та виділити основні бізнес-процеси з поділом бізнес-функцій, що використовуються на стороні клієнтської та серверної частини інформаційної системи;
- створити функціональну модель інформаційної системи, використовуючи стандарт методології функціонального моделювання IDEF0;
- розробити фізичну модель даних, використовуючи стандарт IDEF1X;
- обрати та обґрунтувати вибір СУБД;
- виконати UML моделювання ІС, створивши діаграми прецедентів (Use Case Diagram), послідовності дій (Sequence Diagram), станів (State Machine Diagram), активності (Activity Diagram) і класів (Class Diagram);
- розробити веб-додаток, що забезпечує функціонування обраних бізнес-процесів.

2 ВИМОГИ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Визначення вимог до ІС

Основний процес, що моделюється – процес обліку задач та звітів. Цей бізнес-процес може бути представлений у наступній послідовності дій:

- а) користувач (замовник) додає та закріплює за виконавцем задачу;
- б) у користувача (виконавця) у списку задач з'являється створена замовником задача;
- в) виконавець завантажує звіт про виконану задачу;
- г) замовник ознайомлюється зі звітом та підтверджує виконання задачі (статус задачі змінюється на «виконано», задача переміщується у архів, блокуються усі способи редагування задачі, додавання коментарів, звітів, зміни статусу);
- д) у випадку, якщо звіт не задовільнив замовника, замовник пише коментар до звіту та очікує виправленого звіту від виконавця, статус задачі змінюється на «нова» (тим самим процес повертається до пункту в).

Відповідно до вищенаведеного бізнес-процесу, користувачі у системі мають свій функціонал, що дозволяє їм здійснювати свої частини бізнес-процесу за допомогою веб-застосунку. Згідно з ролями користувачів, для кожної групи веб-застосунків надає окремий функціонал, продемонструємо його нижче.

- Функції усіх користувачів системи:
 - а) вхід у систему (заглушка);
 - б) форма зворотного зв'язку;
 - в) перегляд загальної інформації.
- Функції керівника:
 - а) створення завдань та вказування виконавця;
 - б) перегляд надісланих виконавцем звітів;
 - в) прийняття або відхилення звіту;
 - г) коментування звіту.
- Функції виконавця:
 - а) перегляд отриманих завдань;
 - б) надсилання звіту про виконане завдання;
 - в) перегляд коментарів керівника.

2.2 Розробка функціональної моделі

Відповідно до вимог ІС (зазначені у пункті 2.1) за допомогою програмного засобу All Fusion Process Modeler, розроблено наступну функціональну модель із застосуванням методології IDEF0, що наглядно демонструє процеси у ІС.

Функціональна модель системи представляє собою ієрархічно впорядковані діаграми SADT-моделі. Першим вузлом ієрархії функціональної моделі є контекстна діаграма — «чорний ящик». Контекстна діаграма показує всі механізми, дані управління, входи та виходи системи до зовнішнього середовища. Контекстну діаграму моделі представлено на рисунку 2.1.

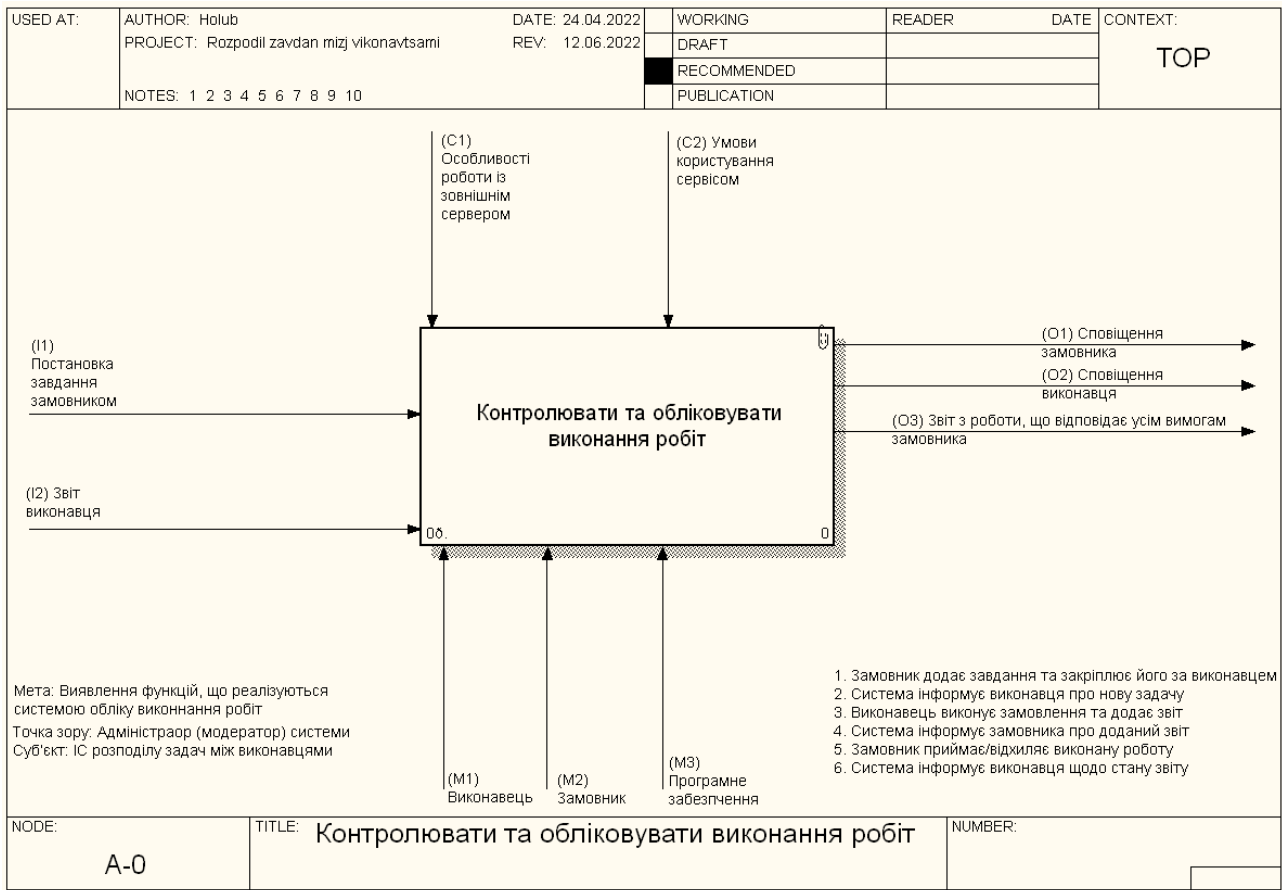


Рисунок 2.1 – Контекстна діаграма функціональної моделі

Контекстна діаграма представляє собою рівень А–0 та в подальшому декомпозується до діаграми рівня А0, функціональні блоки якої також декомпозуються до більш низьких рівнів. Ієрархічна структура функціональної моделі дозволяє показати суть процесів з уточненням до рівня, коли всі окремі процеси ІС будуть чітко зрозумілі.

Декомпонуємо наступні процеси (як ті, що потребують додаткового обговорення):

– «Обліковувати задачі та стани задач» — цей процес відповідає за взаємодію системи з базою даних та файловою системою і складається з наступних процесів:

а) «Зареєструвати задачу / правки / підтвердження / звіт» — додавання або редагування значень у базі даних, збереження звіту з використанням файлової системи;

б) «Обробити задачу / правки / підтвердження» — підготувати дані для додавання до бази даних;

в) «Обробити звіт» — підготувати дані для додавання до бази даних, підготувати завантажуваний файл для додавання до файлової системи;

– «Прийняти задачу / правки / підтвердження від замовника» — цей процес відповідає за взаємодію із користувачем (замовником), реалізацію функціоналу, права на який має замовник:

а) «Сформувати нову задачу / правки / підтвердження» — цей процес відповідає за підготовку даних для додавання у БД, базуючись на початкових вимогах замовника та інформації, отриманої під час взаємодії з користувачем, процес формує відповідні дані, що будуть додані до БД;

б) «Отримати від замовника правки / підтвердження» — цей процес відповідає за взаємодію із користувачем (замовником), коли виконавець відправив звіт, цей процес дозволить замовнику його прийняти або затребувати нові правки.

– «Прийняти звіт від виконавця» — цей процес відповідає за взаємодію із користувачем (виконавцем) та відповідає за можливість додати звіт, після того, як звіт було додано, відповідні дані будуть сформовані та відправлені на обробку.

– Окремо декомпонуємо блок A11 «Зареєструвати задачу / правки / підтвердження / звіт» — цей функціональний вузол відповідає безпосередньо за роботи з базою даних та файловою системою та складається з наступних процесів:

а) «Зберегти новий звіт / сповістити замовника / передати звіт замовникові» — цей вузол відповідає тільки за додавання нового звіту, якщо

процес додавання пройшов успішно, у БД з’явиться запис зі звітом, який зможе побачити замовник;

б) «Зберегти нову задачу / сповістити виконавця» — цей процес відповідає тільки за додавання нової задачі, якщо звіт було успішно додано, у БД з’явиться запис, що зможе побачити виконавець;

в) «Зберегти правки / сповістити виконавця» — якщо правки були успішно прийняті системою, у БД з’явиться запис, що зможе побачити виконавець;

г) «Зареєструвати підтвердження звіту / сповістити виконавця» — процес, що відповідає за реєстрацію підтвердження виконання роботи, коли підтвердження було успішно зареєстровано, у БД статус задачі зміниться на «виконано», що зможе побачити виконавець.

Відповідні діаграми декомпозиції представимо на рисунках 2.3–2.5

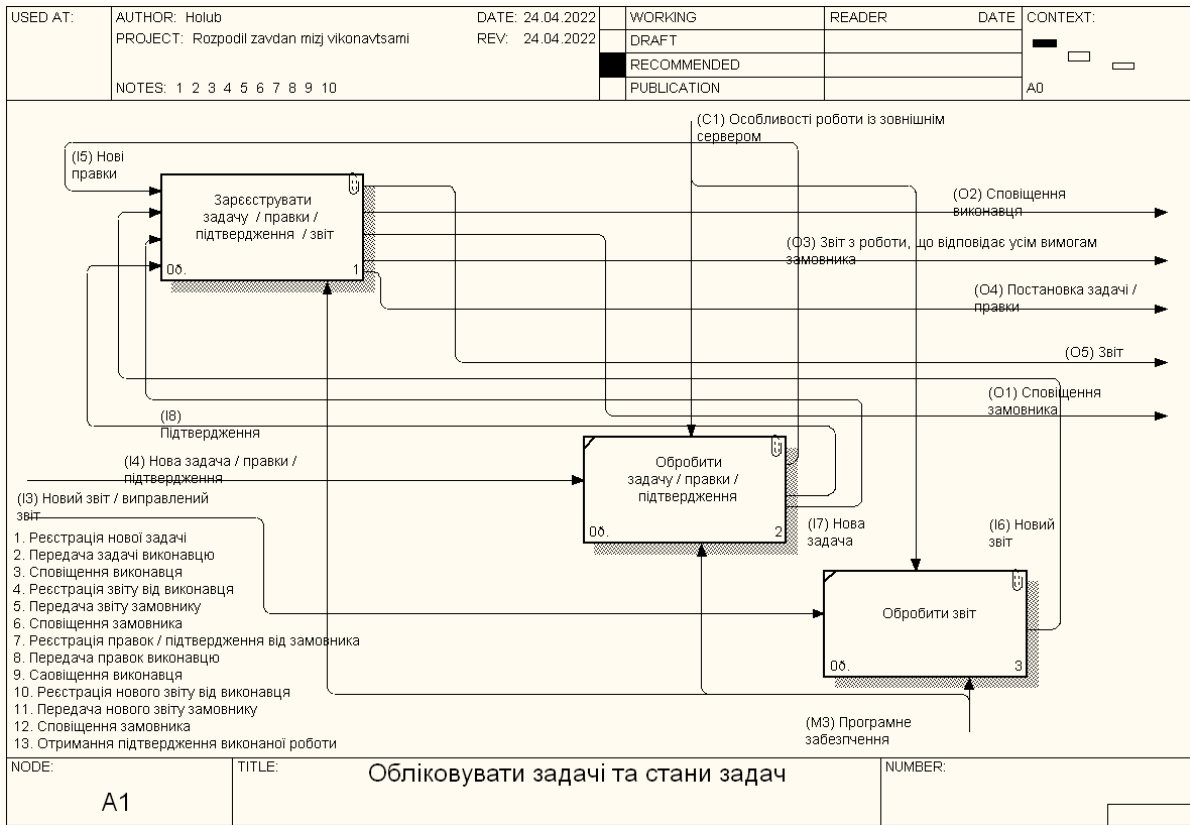


Рисунок 2.3 – Декомпозиція процесу «Обліковувати задачі та стани задач»

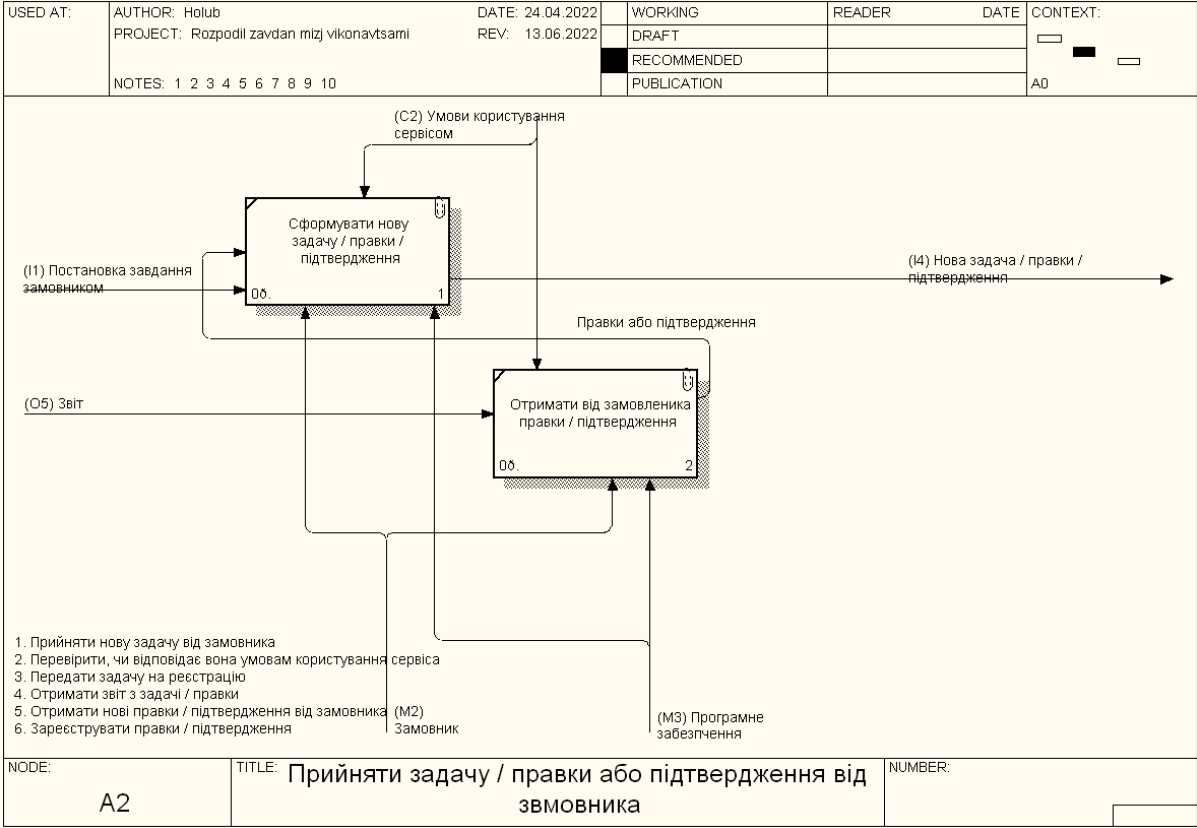


Рисунок 2.4 – Декомпозиція процесу «Прийняти задачу / правки або підтвердження від замовника»

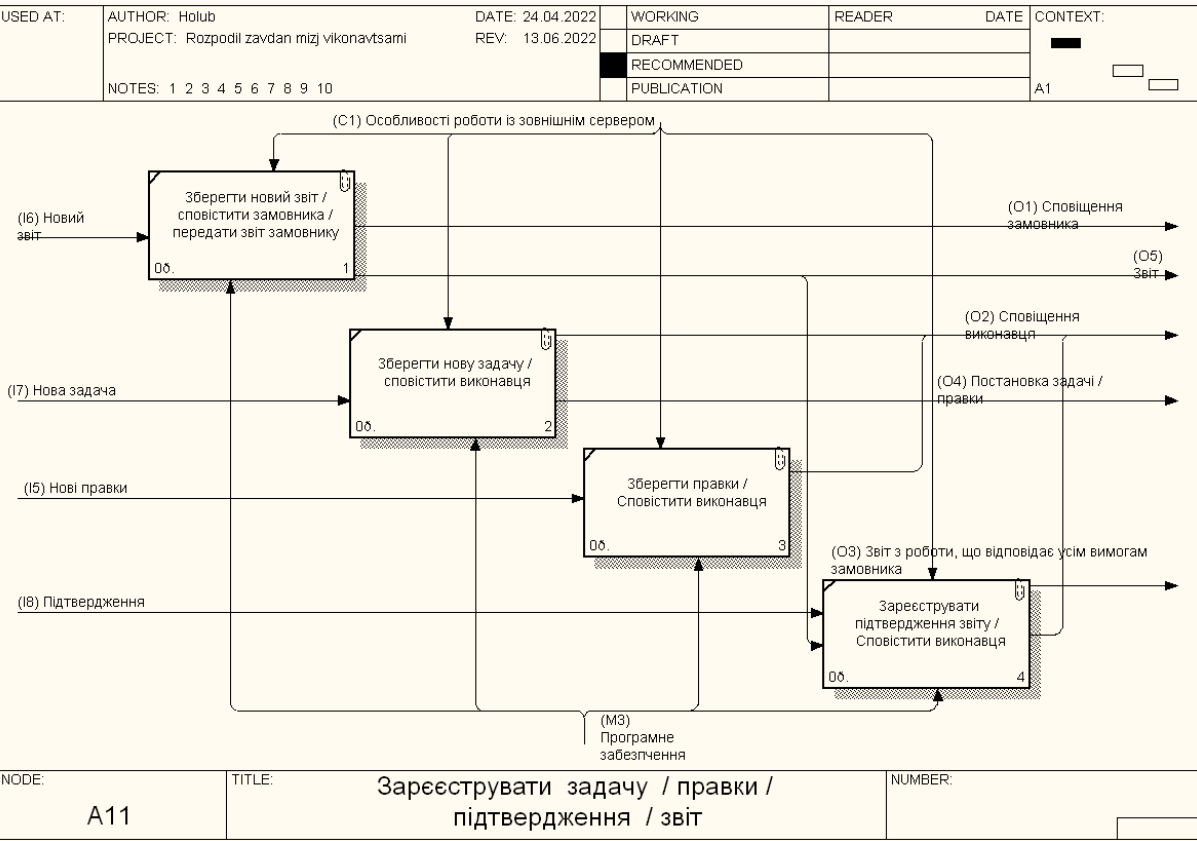


Рисунок 2.5 – Декомпозиція процесу «Зареєструвати задачу / правки / підтвердження / звіт»

2.3 Фізичне моделювання бази даних

Фізична модель бази даних – модель даних, що визначає представлення даних, фізична модель містить усі деталі, необхідні системі управління базами даних для створення бази даних.

Проаналізувавши сутність інформації, якою оперує система та яка повинна зберігатися в таблицях, було виявлено обмеження, типи даних, атрибути та їх особливості. При створенні бази даних, проконтролюємо, що модель даних задовольняла правилам побудови реляційних баз даних, мала атомарні атрибути, не мала транзитивних залежностей неключових атрибутів від первинного ключа, було розроблено фізичну модель даних, наведену на рисунку 2.7.

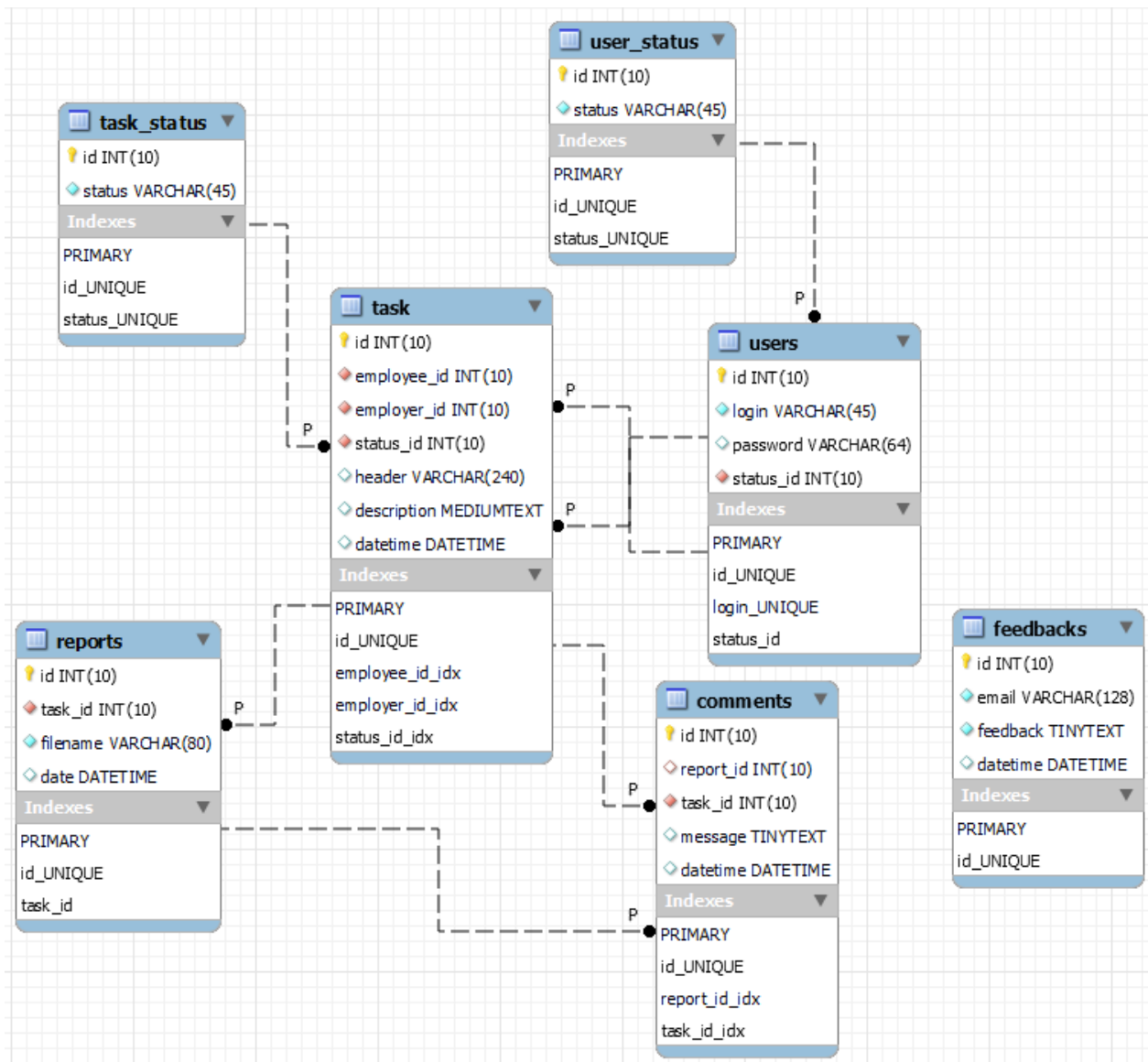


Рисунок 2.7 – Фізична модель бази даних

2.4 UML Моделювання ІС

UML – (Unified Modelling Language) – уніфікована мова моделювання, відкритий стандарт, використовуючи графічні позначення для створення абстрактних моделей систем. UML широко застосовується для об'єктно-орієнтованого аналізу та проектування для визначення, візуалізації та документування програмних засобів.

В рамках курсового проекту було розроблено такі UML діаграми:

- Use Case Diagram – діаграма прецедентів
- Sequence Diagram – діаграма послідовності
- Class Diagram – діаграма класів
- Activity Diagram – діаграма активності
- State Machine Diagram – діаграма станів (для об'єктів зі станами)

2.4.1 Діаграма прецедентів (Use Case Diagram)

Діаграма прецедентів[5] графічно описує варіанти взаємодії акторів (користувачів) з інформаційною системою, випадки використання мають назву прецеденти та описують функціональні аспекти системи. Предметною областю наведеної діаграми є функціональність, що доступна для кожної ролі (типу) користувача ІС. На рисунку 2.8 наведено вищеописану діаграму прецедентів.

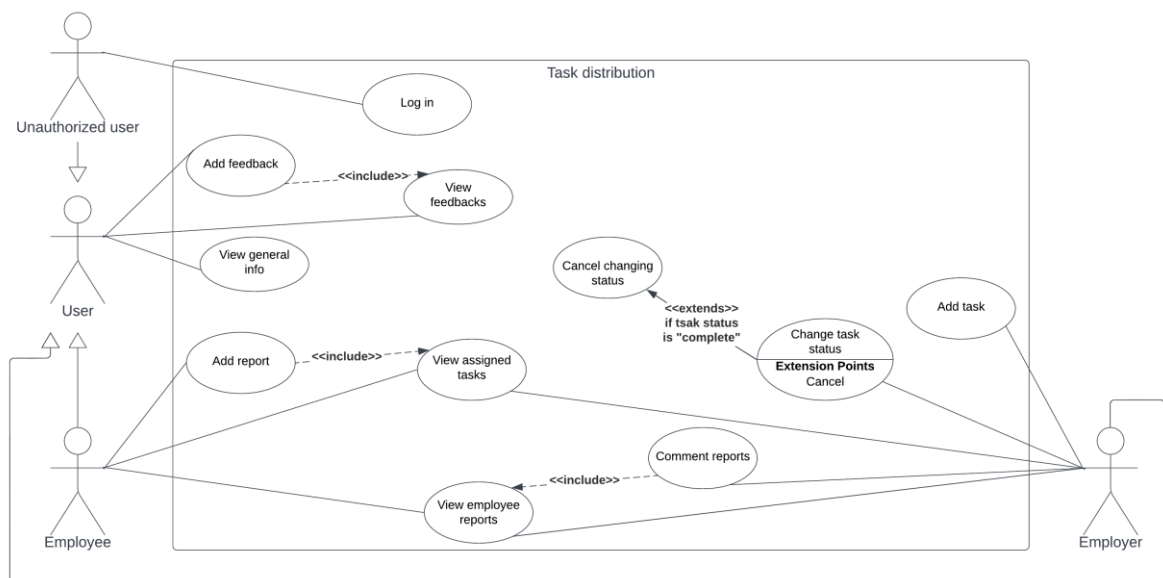


Рисунок 2.8 – Діаграма прецедентів (Use Case Diagram)

2.4.2 Діаграма послідовності (Sequence Diagram)

Діаграма послідовності (Sequence Diagram) відображає впорядковані за часом взаємодії об'єктів, такі діаграми відображають задіяні об'єкти та послідовність обміну повідомленнями між ними. Далі будуть наведені діаграми послідовностей для основних бізнес-процесів (дати задачу, переглянути звіти, додати звіт, прокоментувати звіт, змінити статус задачі). Діаграми проілюстровано на рисунках 2.9 – 2.13

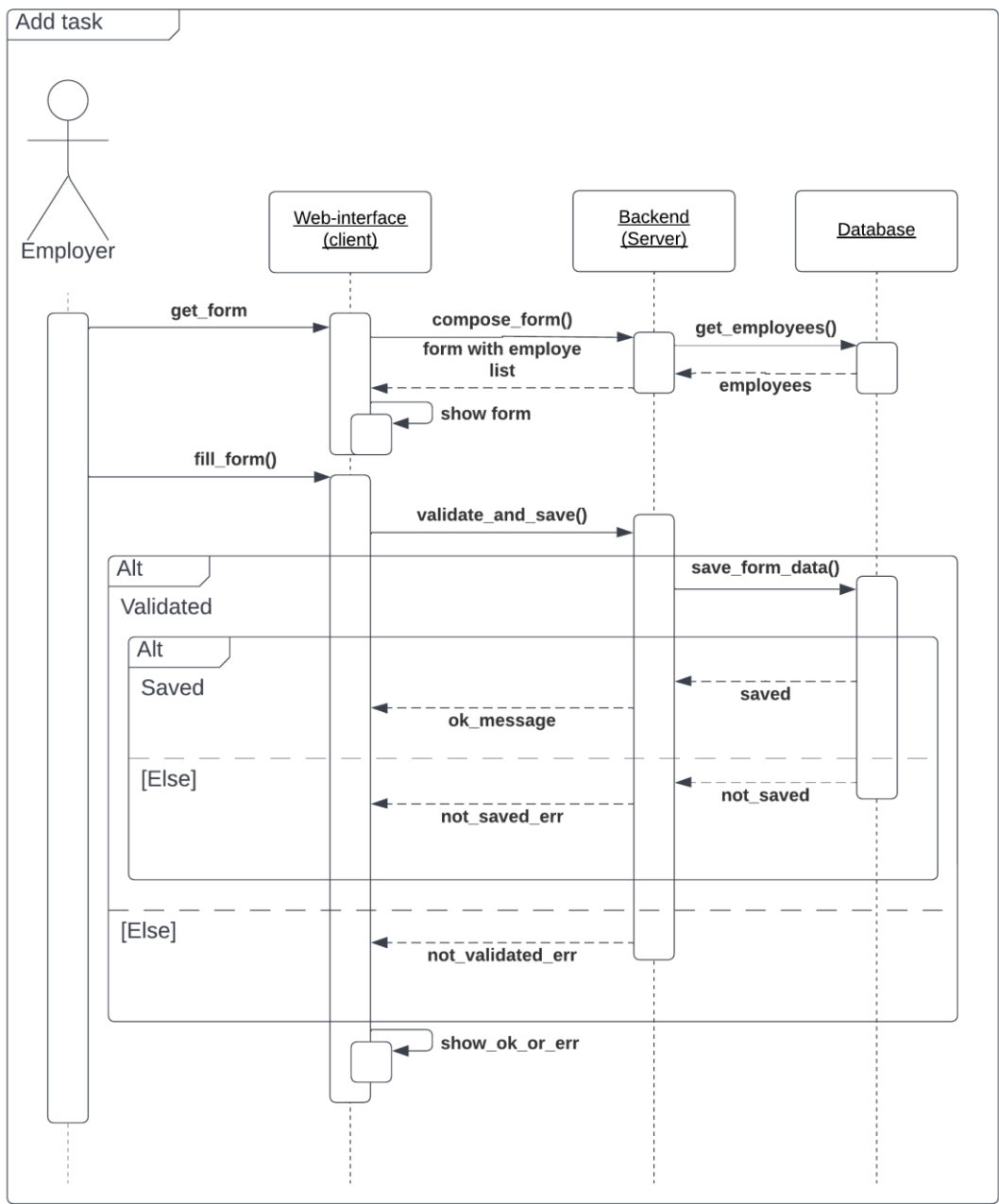


Рисунок 2.9 – Діаграма послідовності процесу додавання задачі

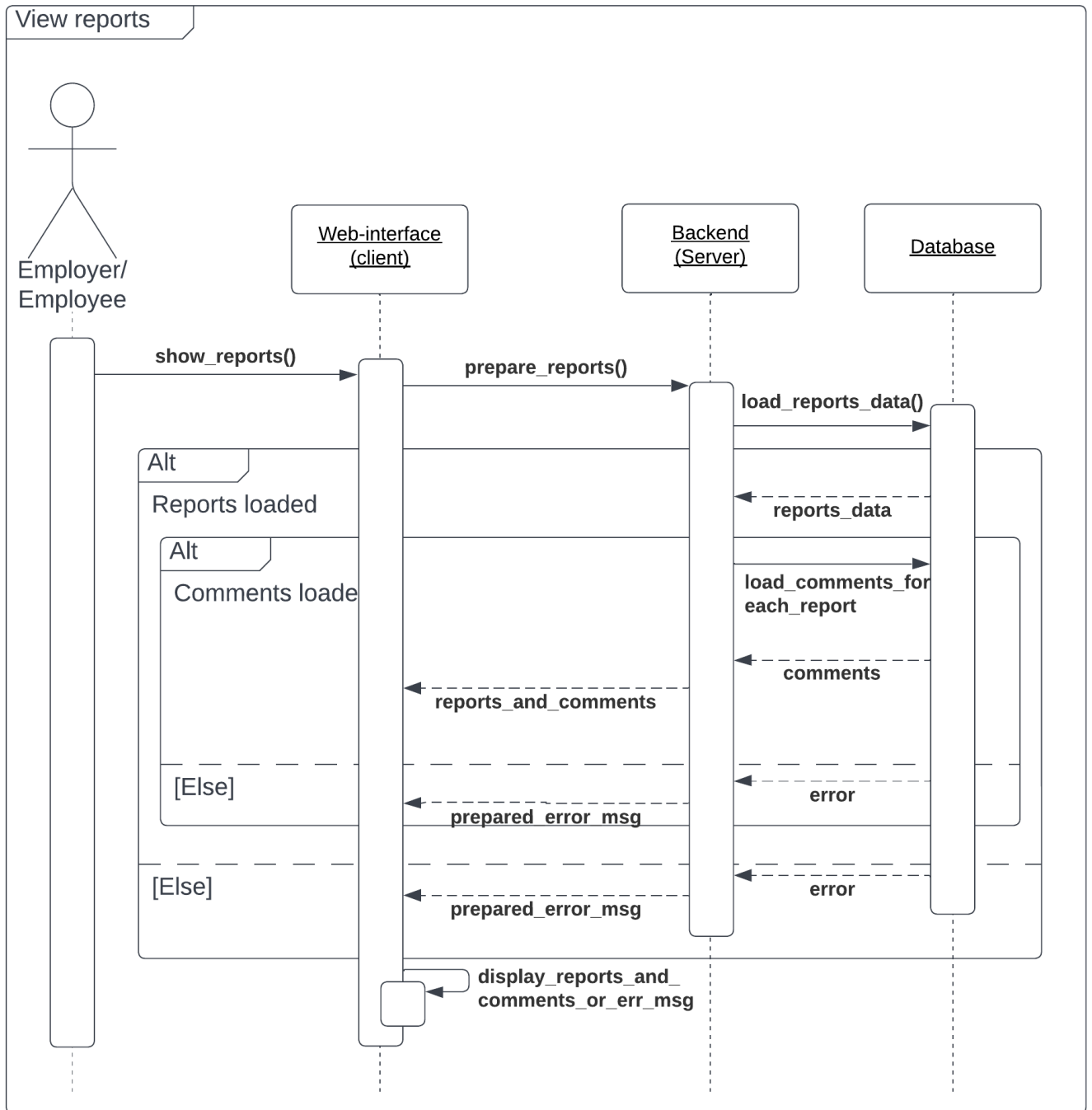


Рисунок 2.10 – Діаграма послідовності процесу отримання списку звітів

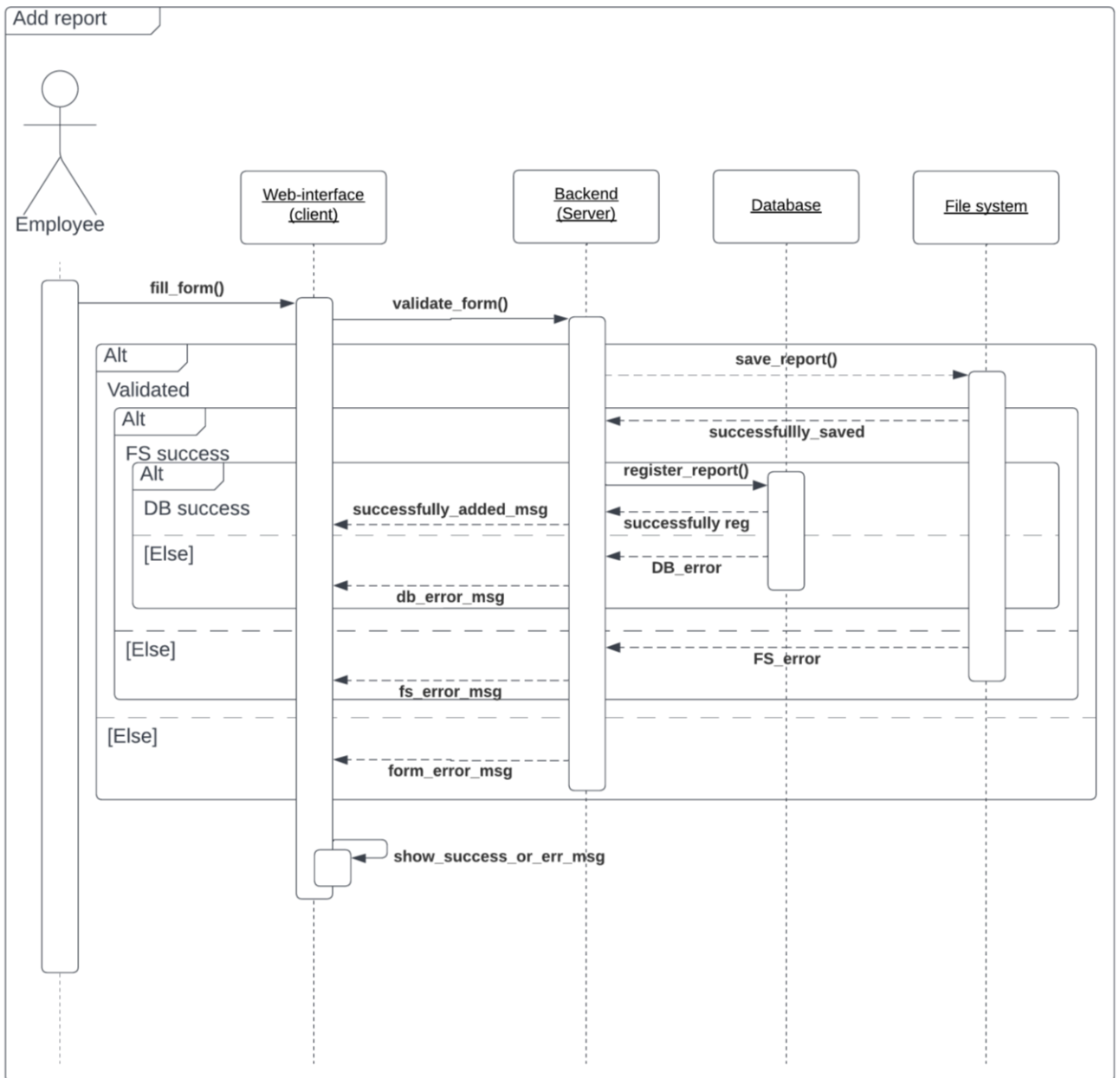


Рисунок 2.11 – Діаграма послідовності процесу додавання звіту

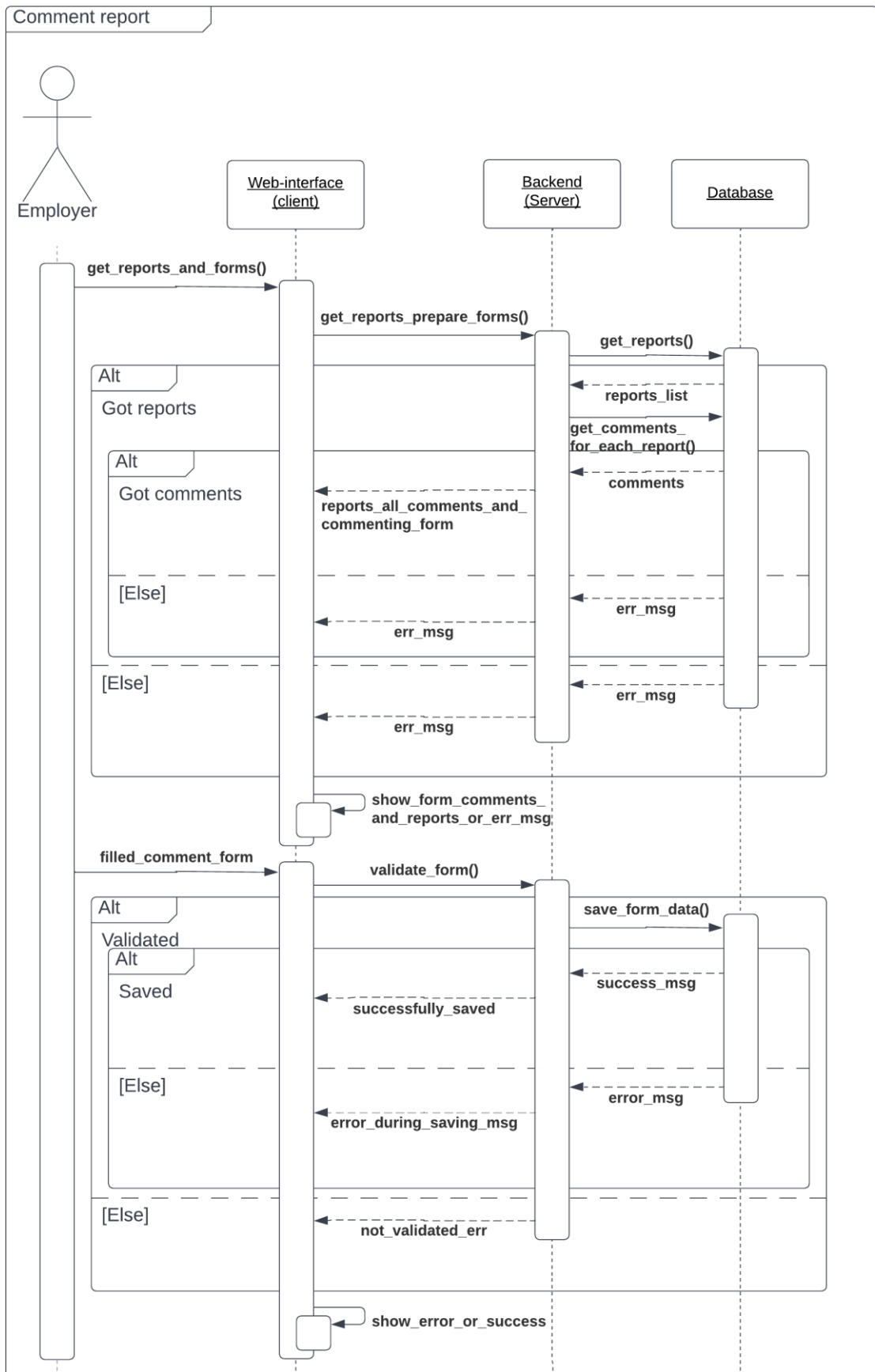


Рисунок 2.12 – Діаграма послідовності процесу коментування звіту

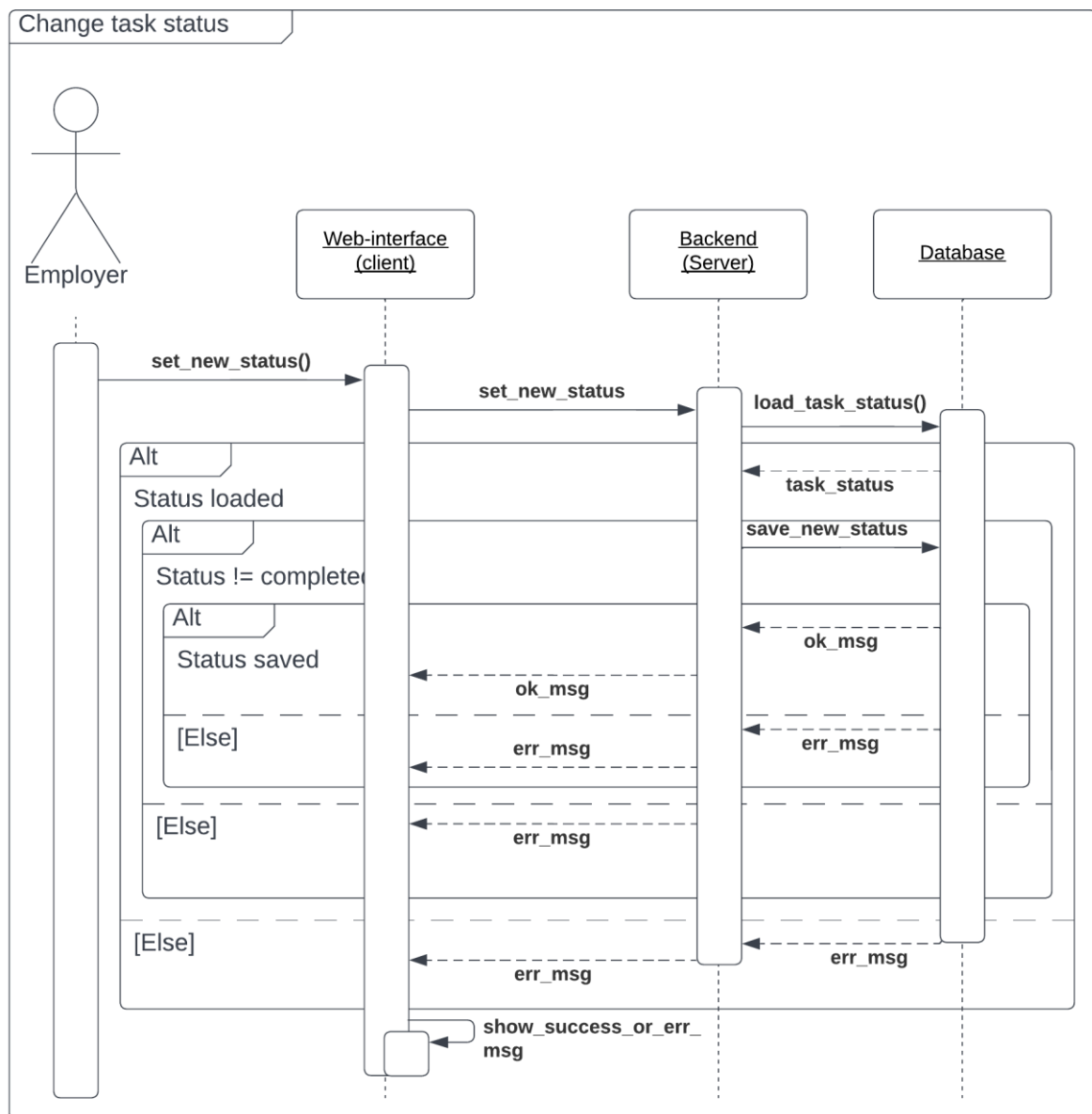


Рисунок 2.13 – Діаграма послідовності процесу зміни статусу задачі

2.4.3 Діаграма класів (Class Diagram)

Діаграма класів (Class Diagram) – статичне відображення інформаційної системи, що описує структуру системи, показуючи класи цієї системи, їх атрибути, методи та відносини між ними. Діаграми класів широко використовуються для моделювання інформаційних систем. У рамках курсового проекту було побудовано ряд класів сутностей, клас роботи з базою даних та

класи, що формують відображення графічного інтерфейсу, діаграми яких представлені на рисунку 2.14.

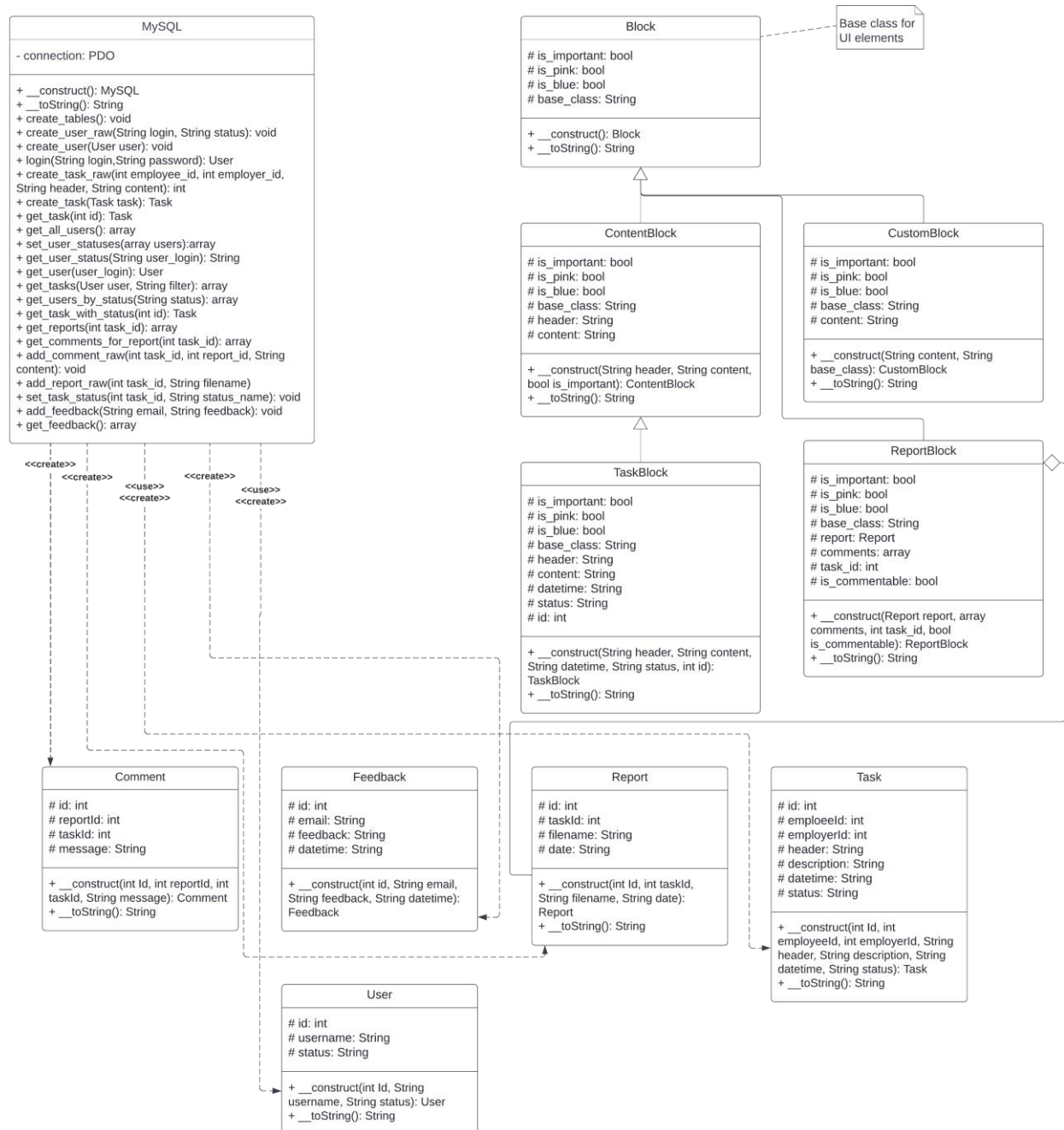


Рисунок 2.14 – Діаграма усіх класів інформаційної системи

2.4.4 Діаграма активності (Activity Diagram)

Діаграма активності (Activity Diagram) — спосіб представлення діаграми, що описує динамічні аспекти моделі. Діаграми діяльності дозволяють описувати бізнес процеси, логіку процедур та потоки робіт. Діаграми активності описують,

як діяльність координується для надання необхідної послуги на різних рівнях абстракції. На відміну від блок-схем, діаграми активності підтримують паралельні процеси. На рисунку нижче (рис 2.15) продемонстровано діаграму діяльності, що описує логіку поведінки інформаційної системи.

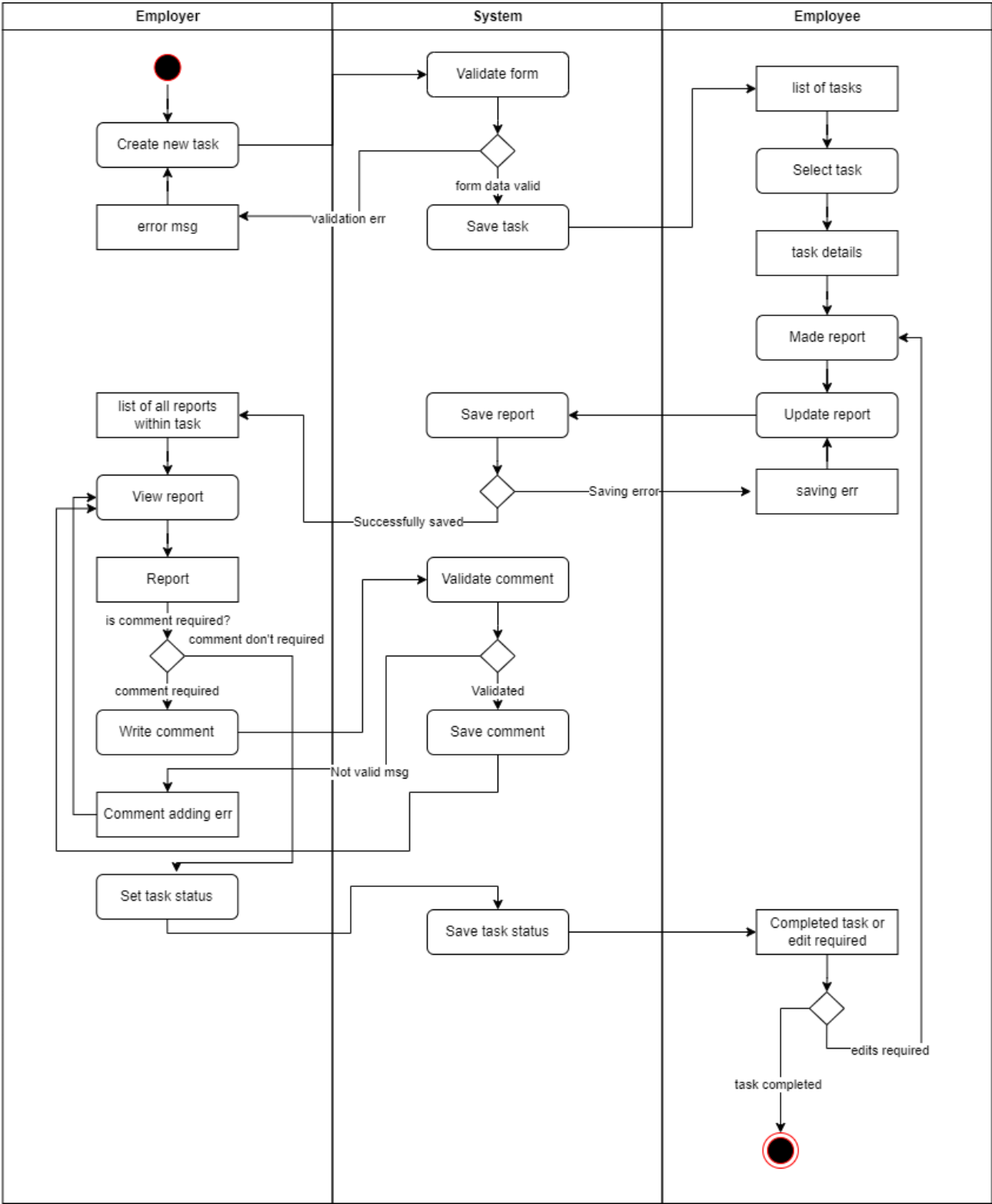


Рисунок 2.15 – Діаграма активності інформаційної системи

2.4.5 Діаграма станів (State Machine Diagram) — це модель поведінки єдиного об'єкта, що вказує зміну станів об'єкта у часі.

Діаграма станів на рис 2.16 відображає зміну станів (статусів) об'єкту «Задача», бо саме цей компонент інформаційної системи має можливість змінювати стан, де статус показує стан задачі у конкретний час. Коли задачу додано до системи, вона має статус «Нова», як тільки до задачі було закріплено звіт, стан задачі змінюється на «У процесі», коли замовник впевнився в коректності виконання задачі, він може змінити її статус на «Виконано», після чого блокуються можливості зміни статусу задачі, додавання коментарів, додавання звітів, задача буде назватися «архівованою».

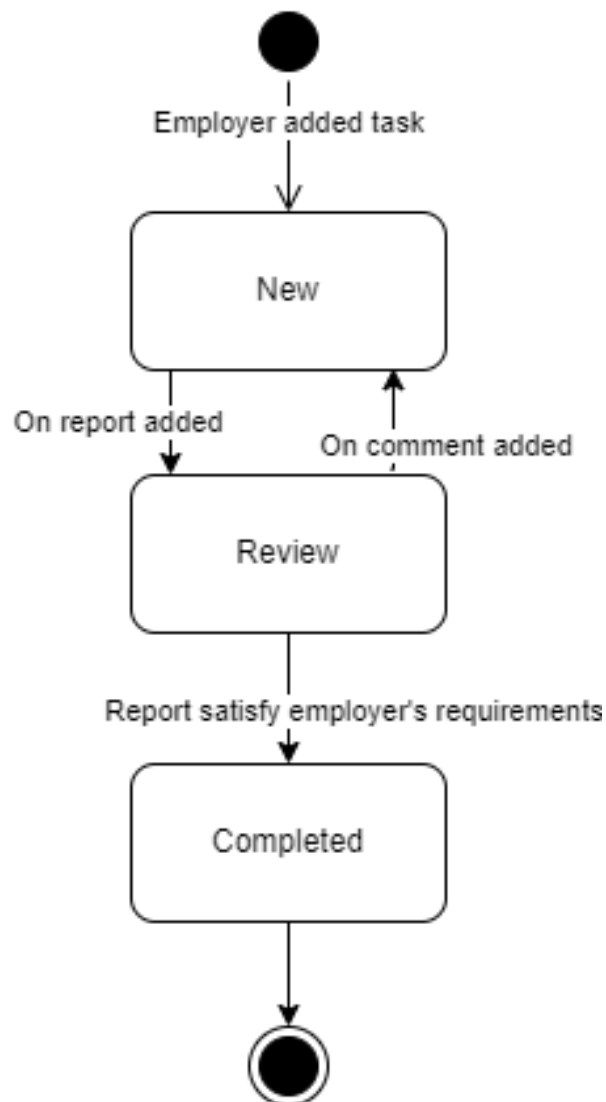


Рисунок 2.16 – Діаграма станів для сутності «Задача»

2.4.6 Розробка вимог до функції інформаційної системи

Серверна частина інформаційної системи повинна підтримувати наступні операції:

- вхід у систему (заглушка);
- додавання, видача та збереження інформації про задачу;
- додавання, видача та збереження коментарів замовника;
- додавання, видача та збереження звітів виконавця;
- додавання, видача та збереження коментарів зворотного зв'язку;
- управління статусом задачі;
- отримання списку виконавців у системі (для додавання задачі);
- пошук закріплених або створених задач (з фільтруванням або сортуванням).

2.4.7 Розробка вимог до клієнтської частини інформаційної системи

Клієнтська частина інформаційної системи повинна мати наступні сторінки:

- головна сторінка (базова інформація про розроблений сервіс);
- сторінка додавання задачі;
- сторінка перегляду та додавання коментарів зворотного зв'язку;
- сторінка завантаження або перегляду звіту;
- сторінка входу у систему;
- сторінка перегляду закріплених або створених задач з можливостями фільтрування та сортування;
- сторінка перегляду детальної інформації про задачу.

3 ОПИС ПРИЙНЯТИХ ПРОЕКТНИХ РІШЕНЬ

3.1 Обґрунтування вибору СУБД, мови програмування та тестового серверу

Для розробки веб-додатку було використано мову програмування PHP, СУБД MySQL, веб-інтерфейс було розроблено за допомогою мови розмітки HTML та мови стилів CSS.

Було розроблено саме браузерний веб-додаток, бо він дозволяє майже на будь-якому пристрої отримати доступ до інформаційної системи без необхідності встановлення додаткового програмного забезпечення.

Для розробки бази даних інформаційної системи було застосовано СУБД MySQL, що має ряд переваг:

- простота у встановленні та використанні;
- якісна та вичерпна документація;
- висока стабільність;
- можливість використання БД декількома користувачами одночасно[3];
- масштабованість;
- висока швидкість виконання команд.

Основною мовою програмування проекту було обрано мову PHP, що має наступні переваги:

- вичерпна документація;
- великий досвід розробки та використання;
- можливість використовувати як функціональну, так і об'єктно-орієнтовану парадигму програмування;
- відкритий доступ, що дозволяє використовувати PHP без необхідності купувати ліцензію[4];
- майже усі існуючі веб-хостинги підтримують PHP.

Графічний інтерфейс було розроблено за допомогою HTML та CSS, такий стек дозволяє відкрити веб-додаток навіть на малопотужних пристроях без необхідності використовувати стороннє ПЗ, також HTML інтегрується з PHP.

Для демонстрації та тестування веб-додатку було використано програмне забезпечення WAMP Server – набір інструментів, що повністю покриває потреби

системи, включаючи у себе веб-сервер Apache, інтерпретатор PHP та СУБД MySQL.

3.2 Створення бази даних

Для автоматичного створення бази даних, згідно до наведеної у пункті 2.3 діаграми, було розроблено SQL скрипт. Для швидкого та автоматичного розгортання БД, достатньо викликати службову сторінку `init.php` з будь-якого браузера, PHP підключиться до СУБД та виконає скрипт автоматичного розгортання, відобразивши у вікні браузера сам скрипт розгортання та статус виконання скрипта.

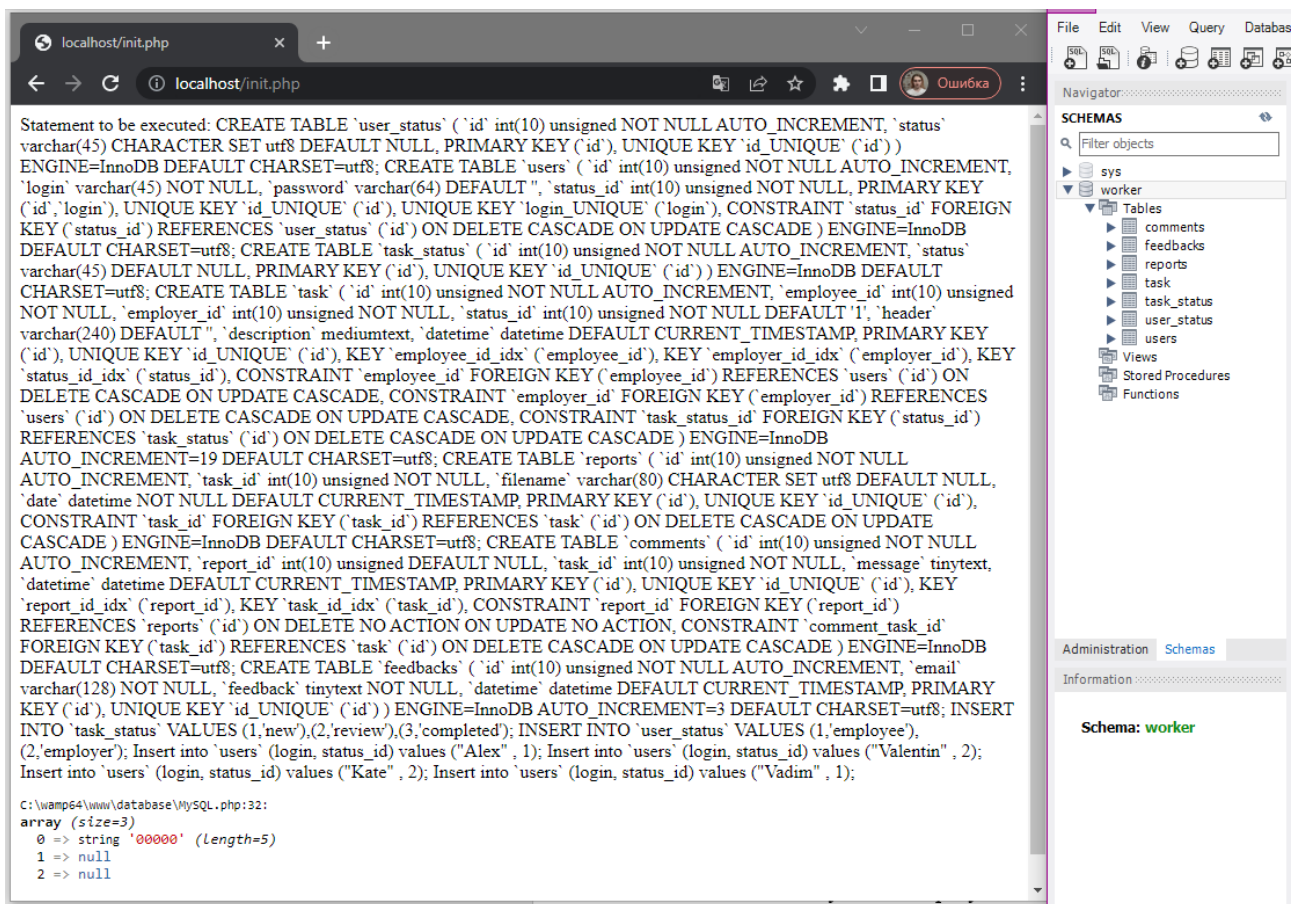


Рисунок 3.1 – результат запуску скрипта розгортання та створені у БД таблиці

ВИСНОВКИ

В ході виконання курсового проекту з дисципліни «Технології Комп'ютерного Проектування» було розроблено систему розподілення задач між виконавцями. В процесі роботи, було закріплено та отримано нові знання з дисциплін «Технології Комп'ютерного Проектування», «Веб-технології та Веб-дизайн», «Організація Баз Даних», «Об'єктно-Орієнтоване Програмування».

Результатом курсового проекту стала розроблена інформаційна система для розподілу завдань між виконавцями «Worker», що буде корисна для роботодавців та виконавців будь-якої компанії. Функціональну модель інформаційної системи створено за допомогою ПЗ AllFusion Process Modeler.

Було проведено аналіз предметної області та розроблено функціональну SADT-модель процесів у системі. На основі цього аналізу було встановлено сутності, їх атрибути та зв'язки між сутностями, було створено фізичну модель бази даних для обраної СУБД, що включає в себе визначення усіх атрибутів та типів даних.

Було проведено UML-проектування інформаційної системи за допомогою веб-додатків lucidchart.com та app.diagrams.net. В ході проектування було створено діаграму прецедентів (Use Case Diagram), діаграми послідовностей (Sequence Diagram) (для основних бізнес-процесів), діаграму класів (Class Diagram), діаграму активності (Activity Diagram), діаграму станів (State Machine Diagram) (для об'єкту зі станом).

За допомогою СУБД MySQL було створено базу даних, веб-додаток було створено за допомогою мови програмування PHP, клієнтська частина формується за допомогою HTML та CSS.

На даному етапі проектування було розроблено першу версію інформаційної системи, що підтримує мінімальний функціонал, вказаний у пояснювальній записці.

Результатом виконання курсового проекту стала інформаційна система з веб-інтерфейсом та серверною частиною, що надає користувачам можливості додавання задач, зміни їх статусу, додавання звітів, коментування звітів, розроблена система є важливою та актуальною на сьогоднішній день, бо дозволяє поліпшити комунікацію замовника з виконавцем та підвищує ефективність роботи компанії, що використовують систему.

ПЕРЕЛІК ПОСИЛАНЬ

1. Методичні вказівки до курсового проектування з дисципліни «Технології комп'ютерного проектування» для студентів першого (бакалаврського) рівня вищої освіти спеціальності 122 – «Комп'ютерні науки» / Упорядники: Ю.В. Міщеряков, А.І. Коваленко, В.М. Решетнік, А.І. Морозова. – Харків: ХНУРЕ, 2019. – 41 с.
2. ДСТУ ГОСТ 7.1:2006. Бібліографічний запис, бібліографічний опис. Загальні вимоги та правила складання : метод. рекомендації з впровадження / уклали: Галевич О. К., Штогрин І. М. – Львів, 2008. – 20 с
3. MySQL 5.7 Reference Manual [Електронний ресурс]: – режим доступу: <https://dev.mysql.com/doc/refman/5.6/en/>
4. PHP Manual [Електронний ресурс]: – режим доступу: <https://dev.mysql.com/doc/refman/5.7/en/>
5. Буч Г. Введение в UML от создателей языка. / Г. Буч, Д. Рамбо, Н. Якобеон — М.: ДМК Пресс, 2010. – 496с.

ДОДАТОК А
КЕРІВНИЦТВО КОРИСТУВАЧА

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

ЗАТВЕРДЖУЮ

Керівник курсового проекту,
посада

проф. Міщеряков Ю. В.

(Прізвище Ініціали)

Інформаційна система розподілу робіт між виконавцями

ПОСІБНИК КОРИСТУВАЧА

ЛИСТ ЗАТВЕРДЖЕННЯ

РОЗРОБНИК:

ст. гр. КНТ-20-1

Голуб Д. К.



(підпис розробника)

ЗАТВЕРДЖЕНО

Інформаційна система розподілу робіт між виконавцями

ПОСІБНИК КОРИСТУВАЧА

Аркушів – 4

2022

ЗМІСТ

Вступ..... 36

А.1 Призначення і умови застосування 36

А.2 Підготовка до роботи 36

А.3 Правила по експлуатації програмного засобу 36

А.4 Аварійні ситуації 38

А.5 Рекомендації з освоєння 38

ВСТУП

Застосунок даної ІС надає можливість покращити та пришвидшити комунікацію між замовником та виконавцем задачі, що збільшить ефективність роботи та потенційно підвищить економічну ефективність компанії, що використовує інформаційну систему. Для коректного та ефективного користування веб-сервісом, користувач має володіти базовими навичками використання веб-додатків.

А.1 Призначення і умови застосування

Веб-додаток представляє собою набір інтерактивних веб-сторінок, пов'язаних між собою за допомогою гіперпосилань, що містять графічний інтерфейс користувача, кожна сторінка надає певну інформацію або дозволяє ввести необхідні дані. Веб-додаток було розроблено з метою надання послуг розподілу задач між виконавцями з використанням сучасних комп'ютерних технологій.

А.2 Підготовка до роботи

Для початку роботи на робочому ПК повинен бути встановлений будь-який веб-браузер, комп'ютер повинен бути підключений до мережі Інтернет, де браузер надає можливість перегляду сторінок веб-сервісу, а з'єднання з Інтернетом можливість обміну інформацією з інформаційною системою.

А.3 Правила по експлуатації програмного засобу

Неавторизований користувач може отримати загальні дані про систему на вкладці «Головна», додати або ознайомитися з відгуками на вкладці «Зворотній зв'язок»

Після того, як користувач продовжив використання веб-додатку, він повинен авторизуватись (рисунок А.3.1), обравши свій робочий профіль, для версії застосунку 1.0 користувачі можуть бути додані у систему за запитом до системного адміністратора, в подальшому, процес реєстрації буде

автоматизовано, користувач зможе увійти у системи увівши свої дані (логін, пароль), ІС перевірить, чи існує такий користувач, та, якщо користувач не існує, дасть можливість заповнити форму реєстрації та увійти в обліковий запис.

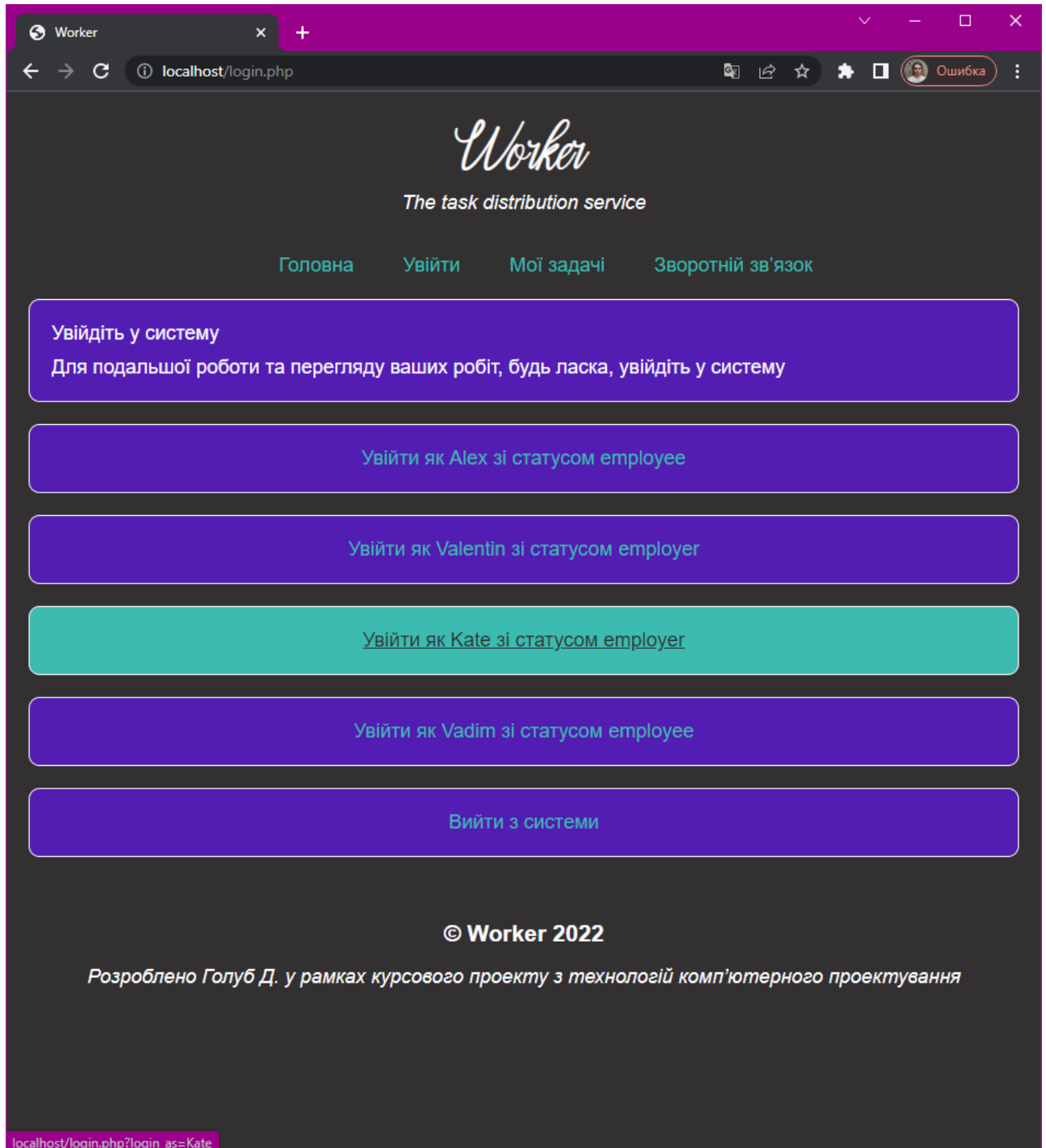


Рисунок А.3.1 – Приклад авторизації

Після успішної авторизації користувач перенаправляється до особистого кабінету.

А.4 Аварійні ситуації

Аварійна ситуація може виникнути при невалідних даних у полях заповненої форми, роботі з базою даних та файловою системою, у випадку аварійної ситуації система надрукує повідомлення, у такому разі необхідно звернутися до адміністратора інформаційної системи. (Приклад повідомлення аварійної ситуації наведено на рис А.4.1)

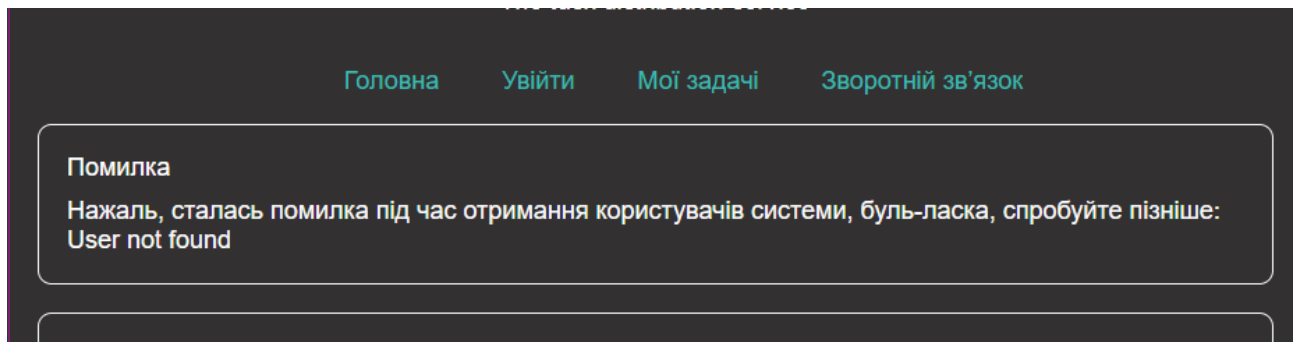


Рисунок А.4.1 – Приклад повідомлення аварійної ситуації

А.5 Рекомендації з освоєння

Перед початком роботи з додатком необхідно мати базові навички користування персональним комп'ютером, впевнитися, що комп'ютер підключено до мережі Інтернет, встановити веб-браузер на свій ПК та ознайомитися з посібником користувача.

ДОДАТОК Б
VISION AND SCOPE

Vision and scope document
For

Worker

Version 1.0

Prepared by Holub Dmytro

	41
Вступ.....	42
Ціль	42
Сфера використання.....	42
Бізнес-вимоги.....	42
Постановка задачі.....	42
Концепція (бачення).....	43
Бізнес-можливості	43
Бізнес ризики	44
Бізнес-контекст	45
Профілі зацікавлених сторін	45
Особливості продукту.....	46
Поза сферою дії MVP.....	46
Web-додаток.....	46
Інші вимоги до продукту	47
Системні вимоги.....	47

Вступ

Ціль

Мета цього документа – зібрати, проаналізувати та визначити високорівневі потреби та особливості системи розподілу робіт між виконавцями. Цей документ фокусується на можливостях, необхідних зацікавленим сторонам та цільовим користувачам, та на тому, чому ці потреби існують. Деталі того, як Система розподілу робіт між виконавцями задовольняє ці потреби, докладно описані нижче.

Сфера використання

Цей документ з описом та змістом відноситься до системи розподілу робіт між виконавцями. Система розподілу робіт між виконавцями надасть користувачам веб-браузерів, які користуються послугами системи, більш просте та зручне рішення в порівнянні з конкурентами. Основна мета проекту – швидке потрапляння до ринку з MVP. У майбутньому систему буде покращено за рахунок додавання нових можливостей, таких як: розподілення роботи між проектами, «прив'язки» виконавця до проекту, отримання статистики по роботодавцям (керівникам) та виконавцям, розширення системи коментарів та введення групових та особистих повідомлень.

Бізнес-вимоги

Постановка задачі

Проблема	Необхідність у системі обліку та контролю виконання завдань працівниками
Впливає на	Підзвітність виконання завдань, менеджмент проекту, легкість, зрозумілість та конкретність поставленого завдання та його статусу, що може викликати непорозуміння як з боку керівника так й з боку виконавця.
Причина	Неінформативність та ускладнений спосіб видання завдань виконавцям та труднощі з управлінням статусу

	завдання під час використання традиційних систем зв'язку.
Вдалим рішенням було б	Зручний та простий веб-застосунок, за допомогою якого керівництво зможе коректно та наглядно розподіляти роботу, а виконавці зручно отримувати робочі завдання, виконувати їх, відправляти звіт щодо виконаної роботи та отримувати відповідь від керівника.

Концепція (бачення)

Для	Керівництва та виконавців на підприємстві
Хто	Бажає покращити свій досвід управління персоналом/зв'язку з керівництвом
APPS	Це веб-застосунок, що дозволяє отримати доступ до інформаційної системи майже з будь-якої платформи
Це	Забезпечує простий та зручний спосіб робочих взаємовідносин
На відміну від	Існуючих систем обліку робочих задач
Наш продукт	Буде мати простий та лаконічний інтерфейс та багато корисних функцій, таких як: коментарі щодо звітів, що треба виправити, внутрішньої системи повідомлень між користувачами

Бізнес-можливості

У нашому швидко розвиваючомуся суспільстві багато людей бажають відкрити свій бізнес та стикаються з масою труднощів, одна з яких – розподілення та обліковність виданих виконавцю підприємцем задач. Облік «на бумазі» не є зручним рішенням, бо, через пандемію багато людей працюють з дому, що викликає необхідність отримувати завдання, наприклад, використовуючи електронну пошту, що є зовсім ненадійним методом передачі повноважень, такий облік потребує значних затрат людино-годин на заповнення бланків. В умовах, коли кожна людина має доступ у інтернет було б

розумно розробити систему, за допомогою якої керівництво може надійно сповістити виконавця про завдання та отримати відповідь у формі звіту, який може бути затверджено або ні.

У нашій країні існує декілька сервісів, що виконують поставлену задачу, такі як: Yourtrack, Trello. Однак, Trello не має функції коментування звітів, що ускладнює уточнення щодо поставлених задач, Yourtrack має складний інтерфейс, доступний лише англійською мовою, що ускладнює взаємодію україномовного користувача з ним.

Розроблювана система повинна задовольнити потреби україномовних користувачів та надати увесь необхідний функціонал її користувачу. Робочий процес застосунку складається з наступних кроків:

- Керівник вказує виконавця та встановлює для нього завдання.
- Виконавець у особистому кабінеті отримує задачу та виконує її.
- Виконавець завантажує звіт про виконану роботу.
- Керівник отримує звіт. У разі якщо робота виконана коректно підтверджує виконання задачі та помічає задачу як «виконану». Навпаки, якщо керівник незадоволений звітом, він коментує його, для виконавця це означає, що звіт має бути відредаговано згідно до зауважень у коментарі.

Після виходу системи на ринок, будуть добавлені такі можливості як: розподілення роботи між проектами, «прив'язки» виконавця до проекту, отримання статистики по роботодавцям (керівникам) та виконавцям, розширення системи коментарів та групових та особистих повідомлень.

Бізнес ризики

- У нашої системи будуть конкуренти – Yourtrack та Trello, ці системи не дуже поширені у нашій країні, однак вони успішно працюють над модернізацією своїх систем, тож важливо якомога швидше увійти на ринок та «завоювати» аудиторію.
- Систему буде розгорнуто на сторонніх серверах, особливості та складності роботи з якими ще не відомі, що може викликати затримку

виконання проекту, перевищення бюджету та невідомі наслідки у випадку збоїв під час тестування системи

Бізнес-контекст

Профілі зацікавлених сторін

Споживачі

Опис	Керівник та співробітник компанії
Тип	Користувач інтернет, підприємець, достатньо навіть незначних навичок володіння ком'ютером для зручної роботи з системою (веб-застосунком)
Критерій успіху	Успіх оцінюється тим, яка кількість клієнтів буде використовувати систему
Участь	Не визначено

Клієнти

Опис	Малі та середні приватні підприємства
Тип	Користувач інтернет, підприємець, достатньо навіть незначних навичок володіння ком'ютером для зручної роботи з системою (веб-застосунком)
Критерій успіху	Успіх визначається кількістю підприємств, що використовують нашу технологію, згодом зростатиме, і клієнти продовжують використовувати нашу систему.
Участь	Не визначено

Користувачі

Роль	Опис	Потреби та вимоги
Гість	Потенційний користувач	Дізнатися більше інформації про систему, її переваги перед рішеннями конкурентів
Керівник	Кінцевий користувач, що виставляє завдання виконавцю	Зручний спосіб користуватися можливостями системи, простий для використання застосунком,

		що надає можливості адміністрування підлеглих
Виконавець	Кінцевий користувач, що отримує завдання та відповідає на нього	Зручний спосіб користуватися можливостями системи, простий для використання застосунком, що надає можливості зв'язку з адміністрацією та отримання завдань
Адміністратор	Людина, що налаштовує акаунти користувачів (частіше сам підприємець)	Адміністрування облікових записів, зміна пароля та створення нових облікових записів
Хостинг-провайдер	Організація, що підтримує доступність системи у мережі інтернет	Не визначено

Особливості продукту

Поза сферою дії MVP

- Реалізація розподілення роботи між проектами, «прив'язки» виконавця до проекту, отримання статистики по роботодавцям (керівникам) та виконавцям, розширення системи коментарів та групових та особистих повідомлень. Ці функції можуть бути реалізовані у наступних версіях застосунку.
- Функції, які можуть допомогти у запобіганні шахрайству (див. пункт «Бізнес ризики»). Передбачається, що цей ризик буде покритий адміністративними заходами.

Web-додаток

- APPS-1: Login
- APPS-2: Manage assigned tasks
- APPS-3: Assign task

- APPS-4: Upload and Download report
- APPS-5: View and add comment
- APPS-6: View and add feedback

Інші вимоги до продукту

Системні вимоги

- Передбачається, що веб-застосунок повинен робити у більшості браузерів, серед яких: Mozilla Firefox, Google Chrome, Microsoft Edge, Safari.
- Веб-застосунок повинен витратити якомога менше трафіку.
- Бекенд повинен бути розроблений з використанням мови програмування PHP.
- База даних повинна бути реалізована на MySQL.

ДОДАТОК В
ТЕКСТ ПРОГРАМИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

ЗАТВЕРДЖУЮ

Керівник курсового проекту,
посада

проф. Міщеряков Ю. В

(Прізвище Ініціали)

Інформаційна система розподілу робіт між виконавцями

ТЕКСТ ПРОГРАМИ

ЛИСТ ЗАТВЕРДЖЕННЯ

РОЗРОБНИК:

ст. гр.КНТ-20-1

Голуб Д.К.



(підпис розробника)

ЗАТВЕРДЖЕНО

Інформаційна система розподілу робіт між виконавцями

ТЕКСТ ПРОГРАМИ

Аркушів – 70

```
/*
```

```
BG-black - 50, 48, 49 / 323031
```

```
Text-white - 251, 251, 251 / FBFBBF
```

```
Purple - 83, 28, 179 / 531CB3
```

```
Pink - 238, 66, 102 / EE4266
```

```
Blue - 60, 187, 177 / 3CBBB1
```

```
*/
```

```
@font-face {
```

```
    font-family: 'InjusticeDemo';
```

```
    src: url('/assets/InjusticeDemo.ttf');
```

```
}
```

```
body {
```

```
    font-family: "Trebuchet", Trebuchet, sans-serif;
```

```
    font-size: 18px;
```

```
    font-weight: 200;
```

```
    background-color: #323031;
```

```
    color: #FBFBFB;
```

```
    margin: 0;
```

```
    padding: 0;
```

```
}
```

```
.logo {
```

```
    padding: 20px 0;
```

```
    text-align: center;
```

```
    line-height: 100%;
```

```
}
```

```
.logo h1 a {
```

```
font-family: InjusticeDemo;  
color: #FBFBFB;  
text-decoration: none;  
}
```

```
.footer {  
  padding: 20px 0;  
  text-align: center;  
  line-height: 100%;  
}
```

```
.footer h1 a {  
  font-family: InjusticeDemo;  
  color: #FBFBFB;  
  text-decoration: none;  
}
```

```
.logo h1 a:hover {  
  color: #3CBBB1;  
  transition: all 500ms ease;  
}
```

```
a {  
  color: #3CBBB1;  
  text-decoration: none;  
}  
a:hover {  
  text-decoration: underline;  
}
```

```
.contentBlock {  
  border-radius: 10px;  
  margin: 20px;
```

```
padding: 20px;
border-color: #FBFBFB;
border-width: 1px;
border-style: solid;
}
.contentHeader {
    color: white;
    margin-bottom: 10px;
}
.contentContent {

}
.important {
    background-color: #531CB3;
}

.navigation {
    width: 100%;
    text-align: center;
}
.navigation li {
    display: inline;
    padding: 0 20px;
}
.navItem {

}
.pink {
    background-color: #EE4266;
}
.blue {
    background-color: #3CBBB1;
}
```

```
.color-black {
    color: #323031;
}

.righten {
    float: right;
}

.smallButton a {
    border-radius: 3px;
    margin: 5px;
    padding: 10px;
    border-color: #FBFBFB;
    color: #FBFBFB;
    border-width: 1px;
    display: inline;
    border-style: solid;
    background-color: #531CB3;
    text-align: center;
    transition: all 500ms ease;
}

.smallButton a:hover {
    text-decoration: none;
    border-radius: 3px;
    margin: 5px;
    padding: 10px;
    border-color: #3CBBB1;
    border-width: 1px;
    display: inline;
    border-style: solid;
    background-color: #531CB3;
    text-align: center;
    transition: all 500ms ease;
}
```

```
.bigButton a {
  border-radius: 10px;
  margin: 20px;
  padding: 20px;
  border-color: #FBFBFB;
  border-width: 1px;
  border-style: solid;
  background-color: #531CB3;
  display: block;
  text-align: center;
  transition: all 500ms ease;
}

.bigButton a:hover {
  border-radius: 10px;
  margin: 20px;
  padding: 20px;
  border-color: #FBFBFB;
  border-width: 1px;
  border-style: solid;
  background-color: #3CBBB1;
  color: #323031;
  transition: all 500ms ease;
}

.right_flex {
  display: flex;
  align-items: top;
  justify-content: right;
}

.flex-item-right {
  display: block;
  align-self: stretch;
  background-color: #2F4F4F;
```

```
}

.half-sized {
  width: 50%;
}

.centred {
  display: flex;
  justify-content: center;
}

.linal {
  display: inline;
}

.grid-vertical {
  display: grid;
}

.grid-vertical-item {
}

.from-fields {
  padding-top: 10px;
  padding-bottom: 10px;
}

.from-fields label {
  color: #FBFBFB;
  width: 100%;
}

.from-fields input {
  margin-top: 10px;
  background-color: #323031;
  color: #FBFBFB;
```



```
width: 100%;  
font-size: 110%;  
border-radius: 10px;  
border-color: #531CB3;  
}  
  
.from-fields select {  
  margin-top: 10px;  
  background-color: #323031;  
  color: #FBFBFB;  
  width: 102%;  
  font-size: 110%;  
  border-radius: 10px;  
  border-color: #531CB3;  
}  
  
.from-fields textarea {  
  margin-top: 10px;  
  background-color: #323031;  
  color: #FBFBFB;  
  width: 102%;  
  font-size: 110%;  
  border-radius: 10px;  
  border-color: #531CB3;  
}  
  
.from-fields-small textarea {  
  margin-top: 10px;  
  background-color: #323031;  
  color: #FBFBFB;  
  width: 500px;  
  height: 100px;  
  font-size: 110%;  
  border-radius: 10px;  
  border-color: #531CB3;
```

```
}

.from-fields-small label {
  color: #FBFBFB;
  width: 100%;
}

.from-fields-small input {
  margin-top: 10px;
  background-color: #323031;
  color: #FBFBFB;
  width: 500px;
  font-size: 110%;
  border-radius: 10px;
  border-color: #531CB3;
}

.inactive {
  background-color: #444444;
}

.blue a {
  color: #323031 !important;
}

.filter-form-fields input{
  border-color: #531CB3;
  color: #FBFBFB;
  background-color: #323031;
  border-radius: 10px;
  font-size: 90%;
  padding: 0 10px;
}

.filter-form-fields select{
  border-color: #531CB3;
  color: #FBFBFB;
```

```

background-color: #323031;
border-radius: 10px;
font-size: 90%;
padding: 0 10px;
}

```

Файл database/MySQL.php

```

<?php

include_once 'utils/Constants.php';
include_once 'utils/logger.php';
include_once 'entity/User.php';
include_once 'entity/Task.php';

class MySQL {
    private $connection;

    public function __construct()
    {
        global $DATABASE_HOST, $DATABASE_PORT, $DATABASE_NAME,
        $DATABASE_PASSWORD, $DATABASE_USER;
        $dsn =
        "mysql:dbname=".$DATABASE_NAME.";host=".$DATABASE_HOST;
        $this->connection = new PDO($dsn, $DATABASE_USER,
        $DATABASE_PASSWORD);
    }

    public function __toString()
    {
        return "Database";
    }

    public function create_tables(): void {

```

```

$statement = file_get_contents("res/tables.sql");
echo "Statement to be executed: " . $statement;
if($statement) {
    $res = $this->connection->exec($statement);
    if($res) {
        my_log("DB successfully created");
    } else {
        var_dump($this->connection->errorInfo());
        my_log("DB error: " . var_dump($res));
    }
} else {
    my_log("Error during reading DB file!");
}
}

public function create_user_raw($login, $status): void { // login, status_name
    $statusIndex = $this->connection->prepare("select id from user_status where
status = ?");
    $is_ok = $statusIndex->execute(array($status));
    $statusIndex = $statusIndex->fetch(PDO::FETCH_ASSOC);
    if(!$is_ok || !$statusIndex) {
        throw new Exception("Failed to get id of the status");
    }
    $statusIndex = $statusIndex["id"];

    my_log("Get user_status index for " . $status . ": " . $statusIndex);

    $statement = $this->connection->prepare("Insert into `users` (login, status_id)
values (?, ?);");
    $is_ok = $statement->execute(array($login, $statusIndex));
    if(!$is_ok) {
        throw new Exception("Failed to create user");
    }
}

```

```

        my_log("User successfully created");
    }

    public function create_user(User $user) {
        $this->create_user_raw($user->getUsername(), $user->getStatus());
    }

    public function login($login, $password): User {
        $user = $this->connection->prepare("select * from users where login = ?;");
        $user->execute(array($login));
        $cur_user = $user->fetch(PDO::FETCH_ASSOC);
        my_log("Trying to get user: " . var_export($cur_user, true));
        if ($user->rowCount() > 0 && isset($cur_user["id"])) {
            $new_user = new User($cur_user["id"], $cur_user["login"],
$cur_user["status_id"]);
        } else {
            throw new Exception("User not found");
        }

        $status = $this->connection->prepare("select * from user_status where id =
?;");
        $is_ok = $status->execute(array($new_user->getStatus()));
        $status_name = $status->fetch(PDO::FETCH_ASSOC);
        if($is_ok && isset($status_name["id"])) {
            $new_user->setStatus($status_name["id"]);
        } else {
            throw new Exception('Status not found');
        }

        return $new_user;
    }

    public function create_task_raw($employee_id, $employer_id, $header, $content)
    {

```

```

        $query = $this->connection->prepare("insert into task (employee_id,
employer_id, status_id, header, description) values (?, ?, ?, ?, ?);");
        $is_ok = $query->execute(array($employee_id, $employer_id, 1, $header,
$content));
        if ($is_ok) {
            return $this->connection->lastInsertId();
        } else {
            throw new Exception("Can't add new task");
        }
    }

    public function create_task(Task $task): Task {
        $id = $this->create_task_raw($task->getEmployeeId(), $task->
getEmployerId(), $task->getHeader(), $task->getDescription());
        $task->setId($id);
        return $task;
    }

    public function get_task($id): Task {
        $query = $this->connection->prepare("SELECT * FROM task where id = ?;");
        $is_ok = $query->execute(array($id));
        $task = $query->fetch(PDO::FETCH_ASSOC);
        if ($is_ok && isset($task["id"])) {
            return new Task($task["id"], $task["employee_id"], $task["employer_id"],
$task["header"], $task["description"], $task["datetime"], $task["status"]);
        } else {
            throw new Exception("Task not found");
        }
    }

    /**
     * @return array
     * @throws Exception
     */

```

```

public function get_all_users(): array
{
    $query = $this->connection->prepare("SELECT * FROM users");
    $is_ok = $query->execute();
    if($is_ok) {
        $users = $query->fetchAll(PDO::FETCH_ASSOC);
        $user_arr = array();
        foreach ($users as $user) {
            my_log("User: " . var_export($user, true));
            $user_arr[] = new User($user["id"], $user["login"], $user["status_id"]);
        }

        return $user_arr;
    } else {
        throw new Exception("Cannot get all users in DB");
    }
}

/**
 * @param $users
 * @return array
 * @throws Exception
 */

public function set_user_statuses(array $users):array {
    $query = $this->connection->prepare("SELECT * FROM user_status");
    $is_ok = $query->execute();
    if($is_ok) {
        $statuses = $query->fetchAll(PDO::FETCH_ASSOC);
        $statuses_arr = array();
        foreach ($statuses as $status) {
            $statuses_arr[$status["id"]] = $status["status"];
        }
        foreach ($users as $user) {
            $user->setStatus($statuses_arr[$user->getStatus()]);
        }
    }
}

```

```

    }
    return $users;
} else {
    throw new Exception("Fail to load user statuses");
}
}

public function get_user_status($user_login) {
    $query = $this->connection->prepare("SELECT * FROM user_status right
join users
on users.status_id = user_status.id where login = ?;");

    $is_ok = $query->execute(array($user_login));
    $result = $query->fetch(PDO::FETCH_ASSOC);
    if($is_ok && isset($result["id"])) {
        return $result["status"];
    } else {
        throw new Exception("Can't get user status");
    }
}

public function get_user($user_login): User {
    $query = $this->connection->prepare("SELECT * FROM user_status right
join users
on users.status_id = user_status.id where login = ?;");

    $is_ok = $query->execute(array($user_login));
    $result = $query->fetch(PDO::FETCH_ASSOC);
    if($is_ok && isset($result["id"])) {
        return new User($result["id"], $result["login"], $result["status"]);
        //$result["status"];
    } else {
        throw new Exception("Can't get user status");
    }
}

```



```

}

public function get_tasks(User $user, $filter): array {

    global $EMPLOYEE_USER_STATUS, $EMPLOYER_USER_STATUS,
    $SORT_OLDER, $SORT_NEW, $SORT_COMPLETED, $SORT_REVIEW
        , $TASK_STATUS_COMPLETED, $TASK_STATUS_REVIEW,
    $TASK_STATUS_NEW;

    $sort_postfix = "order by datetime desc;";
    $params = array($user->getUsername());
    if ($filter != Null) {
        if ($filter == $SORT_OLDER) {
            $sort_postfix = "order by datetime asc";
        } else if ($filter == $SORT_NEW) {
            $sort_postfix = "and task_status.status = ?; ";
            $params = array($user->getUsername(), $TASK_STATUS_NEW);
        } else if ($filter == $SORT_REVIEW) {
            $sort_postfix = "and task_status.status = ?; ";
            $params = array($user->getUsername(), $TASK_STATUS_REVIEW);
        } else if ($filter == $SORT_COMPLETED) {
            $sort_postfix = "and task_status.status = ?; ";
            $params = array($user->getUsername(),
$TASK_STATUS_COMPLETED);
        }
    }

    if (is_numeric($user->getStatus())) {
        $user->setStatus($this->get_user_status($user->getUsername()));
    }
    if ($user->getStatus() == $EMPLOYEE_USER_STATUS) {
        $query = $this->connection->prepare("SELECT task.id, task.employee_id,
task.employer_id, task.header, task.description,

```

```

        task.datetime, task_status.status FROM task inner join task_status on
task.status_id = task_status.id left join users
        on users.id = task.employee_id where users.login = ? " . $sort_postfix);
        my_log("User is employee");
    }
    else if ($user->getStatus() == $EMPLOYER_USER_STATUS) {
        $query = $this->connection->prepare("SELECT task.id, task.employee_id,
task.employer_id, task.header, task.description,
        task.datetime, task_status.status FROM task inner join task_status on
task.status_id = task_status.id left join users
        on users.id = task.employer_id where users.login = ? " . $sort_postfix);
        my_log("User is employer");
    }
    else {
        throw new Exception("Invalid user status: " . $user->getStatus());
    }
    $is_ok = $query->execute($params);

    if($is_ok) {
        $result = $query->fetchAll(PDO::FETCH_ASSOC);
        $tasks = array();
        foreach ($result as $task) {
            if(isset($task["id"])) {
                $tasks[] = new Task($task["id"], $task["employee_id"]
                    , $task["employer_id"], $task["header"], $task["description"],
                    $task["datetime"], $task["status"]);
            }
        }
        return $tasks;
    }
    else {
        var_dump($this->connection->errorInfo(), $is_ok);
        throw new Exception("Can't get user's projects");
    }
}

```

```

    }

    public function get_users_by_status(String $status) {
        global $EMPLOYEE_USER_STATUS;
        $query = $this->connection->prepare("select users.id, users.login,
user_status.status
        from users left join user_status on user_status.id =
        users.status_id where user_status.status = ?;");
        $is_ok = $query->execute(array($status));
        if($is_ok) {
            $result = $query->fetchAll(PDO::FETCH_ASSOC);
            $users = array();
            foreach ($result as $user) {
                $users[] = new User($user["id"], $user["login"], $user["status"]);
            }
        } else {
            throw new Exception("Can't get users");
        }
        return $users;
    }

    public function get_task_with_status($id): Task {
        $query = $this->connection->prepare("select task.id, task.employee_id,
        task.employer_id, task.header, task.description, task.datetime,
task_status.status
        from task left join task_status on task.status_id = task_status.id where task.id
= ?;");
        $is_ok = $query->execute(array($id));
        $task = $query->fetch(PDO::FETCH_ASSOC);
        if ($is_ok && isset($task["id"])) {
            return new Task($task["id"], $task["employee_id"], $task["employer_id"],
$task["header"], $task["description"], $task["datetime"], $task["status"]);
        } else {
            throw new Exception("Task not found");
        }
    }

```

```

    }
}

public function get_reports($task_id) {

    include_once 'entity/Report.php';

    $query = $this->connection->prepare("select * from reports where task_id =
?;");
    $is_ok = $query->execute(array($task_id));
    $reports = $query->fetchAll(PDO::FETCH_ASSOC);
    $res = array();
    if ($is_ok) {
        foreach ($reports as $report) {
            $res[] = new Report($report["id"], $report["task_id"],
$report["filename"], $report["date"]);
        }
        return $res;
    } else {
        throw new Exception("Reports not found");
    }
}

public function get_comments_for_report($task_id) {

    include_once 'entity/Comment.php';

    $query = $this->connection->prepare("select * from comments where
report_id = ?;");
    $is_ok = $query->execute(array($task_id));
    $comments = $query->fetchAll(PDO::FETCH_ASSOC);
    $res = array();
    if ($is_ok) {
        foreach ($comments as $comment) {

```

```

        $res[] = new Comment($comment["id"], $comment["report_id"],
            $comment["task_id"], $comment["message"]);
    }
    return $res;
} else {
    throw new Exception("Comments not found");
}
}

public function add_comment_raw($task_id, $report_id, $content) {
    $query = $this->connection->prepare("insert into comments (report_id,
task_id, message) values (?, ?, ?);");
    $is_ok = $query->execute(array($report_id, $task_id, $content));
    if (!$is_ok) {
        throw new Exception("Cannot add comment");
    }
}

public function add_report_raw($task_id, $filename) {
    $query = $this->connection->prepare("insert into reports (task_id, filename)
value (?, ?);");
    $is_ok = $query->execute(array($task_id, $filename));
    if (!$is_ok) {
        throw new Exception("Cannot add report");
    }
}

public function set_task_status($task_id, $status_name) {
    $query = $this->connection->prepare("update task set status_id =
(select id from task_status where status = ?)
where id = ?;");
    $is_ok = $query->execute(array($status_name, $task_id));
    if (!$is_ok) {
        throw new Exception("Cannot update status");
    }
}

```

```

    }
}

public function add_feedback($email, $feedback) {
    $query = $this->connection->prepare("insert into feedbacks (email, feedback)
values (?, ?);");
    $is_ok = $query->execute(array($email, $feedback));
    if (!$is_ok) {
        throw new Exception("Cannot add feedback");
    }
}

public function get_feedback() {

    include_once 'entity/Feedback.php';

    $query = $this->connection->prepare("SELECT * FROM worker.feedbacks
order by datetime desc;");
    $is_ok = $query->execute();
    $feedbacks = $query->fetchAll(PDO::FETCH_ASSOC);
    $res = array();
    if ($is_ok) {
        foreach ($feedbacks as $feedback) {
            $res[] = new Feedback($feedback["id"], $feedback["email"],
$feedback["feedback"], $feedback["datetime"]);
        }
        return $res;
    } else {
        throw new Exception("Feedbacks not found");
    }
}
}
?>

```

Файл entity/Block.php

```
<?php

class Block {
    protected $is_important; // bool
    protected $is_pink;
    protected $is_blue;
    protected $base_class = "contentBlock";

    public function __construct()
    {

    }

    /**
     * @return mixed
     */
    public function getIsImportant()
    {
        return $this->is_important;
    }

    /**
     * @param mixed $is_important
     */
    public function setIsImportant($is_important): void
    {
        $this->is_important = $is_important;
    }

    /**
     * @return mixed
     */
    public function getIsPink()
```

```

{
    return $this->is_pink;
}

/**
 * @param mixed $is_pink
 */
public function setIsPink($is_pink): void
{
    $this->is_pink = $is_pink;
}

/**
 * @return mixed
 */
public function getIsBlue()
{
    return $this->is_blue;
}

/**
 * @param mixed $is_blue
 */
public function setIsBlue($is_blue): void
{
    $this->is_blue = $is_blue;
}

public function getStyle() {
    $res = $this->base_class;
    if ($this->is_important) {
        $res = $res . " important";
    }
    if ($this->is_blue) {

```



```

        $res = $res . " blue";
    }
    if ($this->is_pink) {
        $res = $res . " pink";
    }
    return $res;
}

public function __toString()
{
    return
        "<div class=\"".$this->getStyle()."\"><div>"
        ;
}

}

class ContentBlock extends Block {
    protected $header;
    protected $content;

    /**
     * @param $header
     * @param $content
     */
    public function __construct($header, $content, $is_important = false)
    {
        $this->header = $header;
        $this->content = $content;
        $this->is_important = $is_important;
    }

    /**

```

```
* @return mixed
*/
public function getHeader()
{
    return $this->header;
}

/**
 * @param mixed $header
 */
public function setHeader($header): void
{
    $this->header = $header;
}

/**
 * @return mixed
 */
public function getContent()
{
    return $this->content;
}

/**
 * @param mixed $content
 */
public function setContent($content): void
{
    $this->content = $content;
}

public function __toString()
```

```

{
    return
        "<div class=\"".$this->getStyle().">
            <div class=\"contentHeader\">
                ".$this->header."
            </div>
            <div class=\"contentContent\">
                ".$this->content."
            </div>
        </div>"
    ;
}
}

class CustomBlock extends Block {
    protected $content;

    /**
     * @param $content
     */
    public function __construct($content, $base_class = "contentBlock")
    {
        $this->content = $content;
        $this->base_class = $base_class;
    }

    /**
     * @return mixed
     */
    public function getContent()
    {
        return $this->content;
    }
}

```

```

/**
 * @param mixed $content
 */
public function setContent($content): void
{
    $this->content = $content;
}

public function __toString()
{
    return
        "<div class=\"".$this->getStyle()."\"><
        ".$this->content."
        </div>"
    ;
}
}

class TaskBlock extends ContentBlock {
    protected $datetime;
    protected $status;
    protected $id;

    /**
     * @param $datetime
     */
    public function __construct($header, $content, $datetime, $status, $id)
    {
        include_once 'utils/Constants.php';
        global $TASK_STATUS_NEW, $TASK_STATUS_COMPLETED,
        $TASK_STATUS_REVIEW;
        $this->datetime = $datetime;
        $this->header = $header;
    }

```

```
$this->content = $content;
$this->id = $id;
$this->status = $status;
if($status == TASK_STATUS_NEW) {
    $this->is_blue = true;
} else if ($status == TASK_STATUS_REVIEW) {
    $this->is_important = true;
}
}

/**
 * @return mixed
 */
public function getStatus()
{
    return $this->status;
}

/**
 * @param mixed $status
 */
public function setStatus($status): void
{
    $this->status = $status;
}

/**
 * @return mixed
 */
public function getId()
{
    return $this->id;
}
```

```

/**
 * @param mixed $id
 */
public function setId($id): void
{
    $this->id = $id;
}

/**
 * @return mixed
 */
public function getDatetime()
{
    return $this->datetime;
}

/**
 * @param mixed $datetime
 */
public function setDatetime($datetime): void
{
    $this->datetime = $datetime;
}

public function getStyle() {
    $res = $this->base_class;
    if ($this->is_important) {
        $res = $res . " important";
    }
    if ($this->is_blue) {
        $res = $res . " blue";
    }
}

```

```

        if ($this->is_pink) {
            $res = $res . " pink";
        }
        return $res;
    }

    public function __toString()
    {
        return
            "<div class=\"".$this->getStyle()."\">
                <div class=\"contentHeader\">
                    <a href = \"/showTask.php?id=\".$this->id.\"\">\".$this->header.\"</a>
                </div>
                <div class=\"contentContent\">
                    \".$this->content.\"
                </div>
                <div class='righten'>\".$this->datetime.\"</div>
                <br />
                <div class='righten'>\".$this->status.\"</div>
                <br />
            </div>";
    }
}

class ReportBlock extends Block {
    protected $report;
    protected $comments;
    protected $task_id;
    protected $is_commentable;

    /**
     * @param $report

```

```

* @param $comments
*/
public function __construct($report, $comments, $task_id, $is_commentable =
false)
{
    $this->report = $report;
    $this->comments = $comments;
    $this->task_id = $task_id;
    $this->is_commentable = $is_commentable;
}

/**
* @return mixed
*/
public function getReport()
{
    return $this->report;
}

/**
* @param mixed $report
*/
public function setReport($report): void
{
    $this->report = $report;
}

/**
* @return mixed
*/
public function getComments()
{
    return $this->comments;
}

```



```

/**
 * @param mixed $comments
 */
public function setComments($comments): void
{
    $this->comments = $comments;
}

public function __toString()
{
    $res = "<div class=\"".$this->getStyle()."\">
        3BIT <a href='/getReport?filename=".$this->report-
>getFilename()."\">".$this->report->getFilename()."</a>";

    foreach ($this->comments as $comment) {
        $res = $res .
            "<div class='contentBlock'>
                Коментар: ".$comment->getMessage()."
            </div>";
    }
    if ($this->is_commentable) {
        $res = $res .
            "<div class='contentBlock from-fields-small'>
                <form action='/showTask.php?approve=false&id=" . $this->task_id
                . "&reportId=" . $this->report->getId() . "' method=\"post\" >

                <label for=\"comment\">Коментар до звіту:</label><br>
                <textarea id=\"comment\" name=\"comment\"></textarea><br>
                <input type=\"submit\" value=\"Додати коментар\">
            </form>
            </div>";
    }
}

```

```

        $res = $res . "<div class='righten'>". $this->report->getDate() ." </div>";

        $res = $res . "</div>";
        return $res;
    }

}
?>

```

Файл entity/Comment.php

```

<?php

class Comment {
    protected $Id;
    protected $reportId;
    protected $taskId;
    protected $message;

    /**
     * @param $Id
     * @param $reportId
     * @param $taskId
     * @param $message
     */

    public function __construct($Id, $reportId, $taskId, $message)
    {
        $this->Id = $Id;
        $this->reportId = $reportId;
        $this->taskId = $taskId;
        $this->message = $message;
    }
}

```

```
}

/**
 * @return mixed
 */
public function getId()
{
    return $this->Id;
}

/**
 * @param mixed $Id
 */
public function setId($Id): void
{
    $this->Id = $Id;
}

/**
 * @return mixed
 */
public function getReportId()
{
    return $this->reportId;
}

/**
 * @param mixed $reportId
 */
public function setReportId($reportId): void
{
    $this->reportId = $reportId;
}
```

```
/**
 * @return mixed
 */
public function getTaskId()
{
    return $this->taskId;
}

/**
 * @param mixed $taskId
 */
public function setTaskId($taskId): void
{
    $this->taskId = $taskId;
}

/**
 * @return mixed
 */
public function getMessage()
{
    return $this->message;
}

/**
 * @param mixed $message
 */
public function setMessage($message): void
{
    $this->message = $message;
}

public function __toString()
{

```

```

        return "Comment: (id: " . $this->Id . ", reportId: "
            . $this->reportId . ", taskId: " . $this->taskId . ", content: "
            . $this->message . ")";
    }

}

?>

```

Файл entity/feedback.php

```

<?php

class Feedback {
    protected $id;
    protected $email;
    protected $feedback;
    protected $datetime;

    /**
     * @param $id
     * @param $email
     * @param $feedback
     * @param $datetime
     */
    public function __construct($id, $email, $feedback, $datetime)
    {
        $this->id = $id;
        $this->email = $email;
        $this->feedback = $feedback;
        $this->datetime = $datetime;
    }

    /**

```

```
* @return mixed
*/
public function getId()
{
    return $this->id;
}

/**
 * @param mixed $id
 */
public function setId($id): void
{
    $this->id = $id;
}

/**
 * @return mixed
 */
public function getEmail()
{
    return $this->email;
}

/**
 * @param mixed $email
 */
public function setEmail($email): void
{
    $this->email = $email;
}

/**
 * @return mixed
 */
```

```

public function getFeedback()
{
    return $this->feedback;
}

/**
 * @param mixed $feedback
 */
public function setFeedback($feedback): void
{
    $this->feedback = $feedback;
}

/**
 * @return mixed
 */
public function getDatetime()
{
    return $this->datetime;
}

/**
 * @param mixed $datetime
 */
public function setDatetime($datetime): void
{
    $this->datetime = $datetime;
}
}

?>

```

Файл entity/Report.php

```
<?php

class Report {
    protected $Id; // Unique numeric value
    protected $taskId; // Id of linked task
    protected $filename; // Generated name of report file
    protected $date; // DateTime of sending

    /**
     * @param $Id
     * @param $taskId
     * @param $filename
     * @param $date
     */
    public function __construct($Id, $taskId, $filename, $date)
    {
        $this->Id = $Id;
        $this->taskId = $taskId;
        $this->filename = $filename;
        $this->date = $date;
    }

    /**
     * @return mixed
     */
    public function getId()
    {
        return $this->Id;
    }

    /**
     * @param mixed $Id
```



```
*/  
public function setId($Id): void  
{  
    $this->Id = $Id;  
}  
  
/**  
 * @return mixed  
 */  
public function getTaskId()  
{  
    return $this->taskId;  
}  
  
/**  
 * @param mixed $taskId  
 */  
public function setTaskId($taskId): void  
{  
    $this->taskId = $taskId;  
}  
  
/**  
 * @return mixed  
 */  
public function getFilename()  
{  
    return $this->filename;  
}  
  
/**  
 * @param mixed $filename  
 */  
public function setFilename($filename): void
```

```

{
    $this->filename = $filename;
}

/**
 * @return mixed
 */
public function getDate()
{
    return $this->date;
}

/**
 * @param mixed $date
 */
public function setDate($date): void
{
    $this->date = $date;
}

public function __toString()
{
    return "Report: (id: " . $this->Id . ", task(Id): "
        . $this->taskId . ", filename: " . $this->filename . ", date: " .
        $this->date . ")";
}
}

?>

```

Файл entity/Task.php

```
<?php
```

```
class Task {
```

```

protected $Id; // Unique numeric value (int)
protected $employeeId; // ID виконавця (int)
protected $employerId; // ID замовника (int)
protected $header; // Header of the task (text)
protected $description; // Description of the task (text)
protected $datetime;
protected $status;

/**
 * @return mixed
 */
public function getStatus()
{
    return $this->status;
}

/**
 * @param mixed $status
 */
public function setStatus($status): void
{
    $this->status = $status;
}

/**
 * @return mixed
 */
public function getDatetime()
{
    return $this->datetime;
}

/**
 * @param mixed $datetime

```

```

*/
public function setDatetime($datetime): void
{
    $this->datetime = $datetime;
}

/**
 * @param $Id
 * @param $employeeId
 * @param $employerId
 * @param $header
 * @param $description
 */
public function __construct($Id, $employeeId, $employerId, $header,
$description, $datetime, $status)
{
    $this->Id = $Id;
    $this->employeeId = $employeeId;
    $this->employerId = $employerId;
    $this->header = $header;
    $this->description = $description;
    $this->datetime = $datetime;
    $this->status = $status;
}

/**
 * @return mixed
 */
public function getId()
{
    return $this->Id;
}

```

```
/**
 * @param mixed $Id
 */
public function setId($Id): void
{
    $this->Id = $Id;
}

/**
 * @return mixed
 */
public function getEmployeeId()
{
    return $this->employeeId;
}

/**
 * @param mixed $employeeId
 */
public function setEmployeeId($employeeId): void
{
    $this->employeeId = $employeeId;
}

/**
 * @return mixed
 */
public function getEmployerId()
{
    return $this->employerId;
}

/**
 * @param mixed $employerId
```

```
*/  
public function setEmployerId($employerId): void  
{  
    $this->employerId = $employerId;  
}  
  
/**  
 * @return mixed  
 */  
public function getHeader()  
{  
    return $this->header;  
}  
  
/**  
 * @param mixed $header  
 */  
public function setHeader($header): void  
{  
    $this->header = $header;  
}  
  
/**  
 * @return mixed  
 */  
public function getDescription()  
{  
    return $this->description;  
}  
  
/**  
 * @param mixed $description  
 */  
public function setDescription($description): void
```

```

{
    $this->description = $description;
} // Description of the task (text)

public function __toString()
{
    return "Task: (id: " . $this->Id . ", employee: "
        . $this->employeeId . ", employer: " . $this->employerId . ", header: "
        . $this->header . ", description: " . $this->description . ", datetime: " . $this-
>datetime
        . ", status: " . $this->status . ")";
}
}
?>

```

Файл entity/User.php

```

<?php

Class User {

    protected $Id; // Unique numeric value
    protected $username; // String of username
    protected $status; // Just numeric value

    /**
     * @param $Id
     * @param $username
     * @param $status
     */
    public function __construct($Id, $username, $status)
    {
        $this->Id = $Id;
        $this->username = $username;
    }
}

```

```
$this->status = $status;
}

/**
 * @return mixed
 */
public function getId()
{
    return $this->Id;
}

/**
 * @param mixed $Id
 */
public function setId($Id): void
{
    $this->Id = $Id;
}

/**
 * @return mixed
 */
public function getUsername()
{
    return $this->username;
}

/**
 * @param mixed $username
 */
public function setUsername($username): void
{
    $this->username = $username;
}
```



```

/**
 * @return mixed
 */
public function getStatus()
{
    return $this->status;
}

/**
 * @param mixed $status
 */
public function setStatus($status): void
{
    $this->status = $status;
}

public function __toString()
{
    return "User: (id: " . $this->Id . ", username: " . $this->username . ", status: " .
$this->status . ")";
}
}
?>

```

Файл forms/addReport.php

```

<div class="centred contentBlock">

    <div class="half-sized grid-vertical">
        <form enctype="multipart/form-data" action="/showTask.php?id=<?=$task_id?> ", method="post">
            <div class="grid-vertical-item from-fields">

```

```

        <label for="report">Додати файл звіту:</label><br>
        <input type="file" name="report" id="report">
    </div>

    <div class="grid-vertical-item from-fields">
        <input type="submit" value="Відправити звіт">
    </div>
</form>
</div>
</div>

```

Файл forms/addTaskForm.php

```

<div class="centred contentBlock">

    <div class="half-sized grid-vertical">
        <form action="/addTask.php">
            <div class="grid-vertical-item from-fields">
                <label for="caption">Заголовок задачі:</label><br>
                <input type="text" id="caption" name="caption" value="<?php
                    if(isset($_GET["caption"])) {echo ($_GET["caption"]);}
                    ?>">
            </div>

            <div class="grid-vertical-item from-fields">
                <label for="employee">Виконавець:</label><br>
                <select name="employee" id="employee">
                    <?php foreach ($employees as $employee) : ?>
                        <option value="<?= $employee->getId() ?>"><?= $employee-
>getUsername() ?></option>
                    <?php endforeach ?>
                </select>
            </div>

            <div class="grid-vertical-item from-fields">

```

```

<label for="description">Опис задачі:</label><br>
<textarea id="description" name="description"><?php
    if(isset($_GET["description"])) {echo ($_GET["description"]);}
    ?></textarea>
</div>

<div class="grid-vertical-item from-fields">
    <input type="submit" value="Додати задачу">
</div>
</form>
</div>
</div>

```

Файл forms/feedbackForm.php

```

<div class="centred contentBlock">

<div class="half-sized grid-vertical">
    <form action="/feedback.php" method="post">
        <div class="grid-vertical-item from-fields">
            <label for="email">E-mail:</label><br>
            <input type="text" id="email" name="email" value="<?php
                if(isset($_POST["email"])) {echo ($_POST["email"]);}
                ?>">
        </div>

        <div class="grid-vertical-item from-fields">
            <label for="comment">Ваше питання/побажання:</label><br>
            <textarea id="comment" name="comment"></textarea>
        </div>

        <div class="grid-vertical-item from-fields">
            <input type="submit" value="Відправити форму">
        </div>
    </form>

```

```
</div>
</div>
```

Файл forms/sortingForm.php

```
<div class="centred contentBlock">
  <div>
    <form action="/myProjects.php" method="post">
      <div class="lineral filter-form-fields">
        <select name="filter" id="filter">
          <option value="newer">Спочатку нові</option>
          <option value="older">Спочатку старі</option>
          <option value="completed">Виконані</option>
          <option value="new">Нові</option>
          <option value="review">Потребують правок</option>
        </select>
        <input type="submit" value="Сортувати/фільтрувати">
      </div>
    </form>
  </div>
</div>
```

Файл res/tables.sql

```
CREATE TABLE `user_status` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `status` varchar(45) CHARACTER SET utf8 DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `users` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `login` varchar(45) NOT NULL,
  `password` varchar(64) DEFAULT "",
  `status_id` int(10) unsigned NOT NULL,
```

```

PRIMARY KEY (`id`,`login`),
UNIQUE KEY `id_UNIQUE` (`id`),
UNIQUE KEY `login_UNIQUE` (`login`),
CONSTRAINT `status_id` FOREIGN KEY (`status_id`) REFERENCES
`user_status` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `task_status` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `status` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `task` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `employee_id` int(10) unsigned NOT NULL,
  `employer_id` int(10) unsigned NOT NULL,
  `status_id` int(10) unsigned NOT NULL DEFAULT '1',
  `header` varchar(240) DEFAULT "",
  `description` mediumtext,
  `datetime` datetime DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`),
  KEY `employee_id_idx` (`employee_id`),
  KEY `employer_id_idx` (`employer_id`),
  KEY `status_id_idx` (`status_id`),
  CONSTRAINT `employee_id` FOREIGN KEY (`employee_id`) REFERENCES
`users` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `employer_id` FOREIGN KEY (`employer_id`) REFERENCES
`users` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `task_status_id` FOREIGN KEY (`status_id`) REFERENCES
`task_status` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=19 DEFAULT CHARSET=utf8;

```

```
CREATE TABLE `reports` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `task_id` int(10) unsigned NOT NULL,
  `filename` varchar(80) CHARACTER SET utf8 DEFAULT NULL,
  `date` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`),
  CONSTRAINT `task_id` FOREIGN KEY (`task_id`) REFERENCES `task`
(`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `comments` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `report_id` int(10) unsigned DEFAULT NULL,
  `task_id` int(10) unsigned NOT NULL,
  `message` tinytext,
  `datetime` datetime DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`),
  KEY `report_id_idx` (`report_id`),
  KEY `task_id_idx` (`task_id`),
  CONSTRAINT `report_id` FOREIGN KEY (`report_id`) REFERENCES
`reports` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `comment_task_id` FOREIGN KEY (`task_id`) REFERENCES
`task` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `feedbacks` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `email` varchar(128) NOT NULL,
  `feedback` tinytext NOT NULL,
  `datetime` datetime DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
```

```

    UNIQUE KEY `id_UNIQUE` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;

INSERT INTO `task_status` VALUES (1,'new'),(2,'review'),(3,'completed');
INSERT INTO `user_status` VALUES (1,'employee'),(2,'employer');
Insert into `users` (login, status_id) values ("Alex" , 1);
Insert into `users` (login, status_id) values ("Valentin" , 2);
Insert into `users` (login, status_id) values ("Kate" , 2);
Insert into `users` (login, status_id) values ("Vadim" , 1);

```

Файл utils/Constants.php

```

<?php

$DATABASE_HOST = "localhost";
$DATABASE_PORT = "3306";
$DATABASE_USER = "root";
$DATABASE_PASSWORD = "";
$DATABASE_NAME = "worker";
$COOKIE_LOGIN_TIME = 60*60*24*30;
$COOKIE_TEMPORY_TIME = 10;
$EMPLOYEE_USER_STATUS = "employee";
$EMPLOYER_USER_STATUS = "employer";
$TASK_STATUS_NEW = "new";
$TASK_STATUS_REVIEW = "review";
$TASK_STATUS_COMPLETED = "completed";
$FILE_SAVE_PATH = "files/";

$SORT_NEWER = "newer";
$SORT_OLDER = "older";
$SORT_NEW = "new";
$SORT_COMPLETED = "completed";
$SORT_REVIEW = "review";

?>

```

Файл utils/logger.php

```

<?php

function my_log($data){ // сама функція
    if(is_array($data) || is_object($data)){
        echo("<script>console.log('php_array: '".json_encode($data)."'");</script>");
    } else {
        echo("<script>console.log('php_string: ".$data."");</script>");
    }
}

?>

```

Файл addTask.php

```

<?php include_once 'header.php'; ?>

<?php
    include_once 'entity/Block.php';
    include_once 'entity/User.php';
    include_once 'database/MySQL.php';
    include_once 'utils/Constants.php';

    global $EMPLOYER_USER_STATUS, $EMPLOYEE_USER_STATUS,
    $COOKIE_TEMPORY_TIME;

    $db = new MySQL();
    $blocks = array();
    $isLoggedIn = isset($_COOKIE["login"]) && isset($_COOKIE["user_status"]);

    if (!$isLoggedIn) {
        $blocks[] = new ContentBlock("Увійдіть у систему", "Для подальшої
роботи та перегляду ваших робіт,
        будь ласка, увійдіть у систему", true);
    } else {
        if ($_COOKIE["user_status"] === $EMPLOYER_USER_STATUS) {

```



```

        if(!empty($_GET) && isset($_GET["caption"]) && $_GET["caption"] !=
        ""

            && isset($_GET["employee"])
            && isset($_GET["description"]) && $_GET["description"] != "") {
                //setcookie("tmp_is_task_just_created", true, time() +
                $COOKIE_TEMPORY_TIME);

                $employer = $db->get_user($_COOKIE["login"]);
                $task = new Task(Null, $_GET["employee"], $employer->getId(),
                $_GET["caption"], $_GET["description"], Null, 1);
                $_GET["caption"]="";
                $_GET["description"]="";
                try{
                    $db->create_task($task);
                    $blocks[] = new CustomBlock("Задачу " . $task->getHeader() . "
                успішно додано", "contentBlock blue color-black");

                } catch (Exception $ex) {
                    $blocks[] = new CustomBlock("Вибачте, під час додавання задачі
                сталася помилка,
                    будь ласка, спробуйте знову\n" . $ex , "contentBlock pink");
                    var_dump($_GET);
                }
            }
        else {
            if (!empty($_GET)) {
                $blocks[] = new CustomBlock("Вибачте, одне або більше поле
                форми було незаповнено,
                    будь ласка, спробуйте знову", "contentBlock pink");
            }
        }

        try {

```

```

        $employees = $db-
>get_users_by_status($EMPLOYEE_USER_STATUS);
        include_once 'forms/addTaskForm.php';
    } catch (Exception $ex) {
        $blocks[] = new CustomBlock("Вибачте, під час отримання списку
користувачів,
        будь ласка, спробуйте знову\n" . $ex , "contentBlock pink");
    }
}
else {
    $blocks[] = new CustomBlock("Вибачте, тільки замовники можуть
додавати нові завдання", "contentBlock pink");
}
}

?>

<?php foreach ($blocks as $block) : ?>
    <?= $block ?>
<?php endforeach ?>

<?php include_once 'footer.php'; ?>

```

Файл feedback.php

```

<?php include_once 'header.php'; ?>

<?php
    include_once 'entity/Block.php';
    include_once 'database/MySQL.php';
    $blocks = array();
    $db = new MySQL();
    if (isset($_POST["email"])) {

```

```

if (strlen($_POST["email"]) > 5) {
    if (isset($_POST["comment"]) && strlen($_POST["comment"]) > 5) {
        try {
            $db->add_feedback($_POST["email"], $_POST["comment"]);
        } catch (Exception $ex) {
            $blocks[] = new CustomBlock("Вибачте, ми не можемо додати
відгук,
            будь ласка, спробуйте пізніше: " . $ex, "contentBlock pink");
        }
    } else {
        $blocks[] = new CustomBlock("Вибачте, ми не можемо додати відгук,
поле відгуку повинно містити щонайменше 5 символів
        будь ласка, спробуйте знову", "contentBlock pink");
    }
} else {
    $blocks[] = new CustomBlock("Вибачте, ми не можемо додати відгук,
будь ласка вкажіть свій e-mail та спробуйте знову"
    , "contentBlock pink");
}
}

try {
    $feedbacks = $db->get_feedback();
    foreach ($feedbacks as $feedback) {
        $blocks[] = new ContentBlock($feedback->getEmail(), $feedback-
>getFeedback() . "<br />
        <div class='righten'>".$feedback->getDatetime()."</div>");
    }
} catch (Exception $ex ) {
    $blocks[] = new CustomBlock("Вибачте, ми не можемо отримати відгуки,
    будь ласка, спробуйте пізніше: " . $ex, "contentBlock pink");
}
}
?>

```

```
<?php include_once 'forms/feedbackForm.php'; ?>
```

```
<?php foreach ($blocks as $block) : ?>
```

```
    <?= $block ?>
```

```
<?php endforeach ?>
```

```
<?php include_once 'footer.php'; ?>
```

Файл footer.php

```
<div class="footer">
```

```
    <h3>© Worker 2022</h3>
```

```
    <i>Розроблено Голуб Д. у рамках курсового проекту з технологій  
комп'ютерного проектування</i>
```

```
</div>
```

```
</body>
```

```
</html>
```

Файл getReport.php

```
<?php include_once 'header.php'; ?>
```

```
<div class="contentBlock important">
```

```
    <div class="contentHeader">
```

```
        Будьте обережні
```

```
    </div>
```

```
    <div class="contentContent">
```

```
        Завантажені файли можуть завдати школи вашому ком'ютеру, будь  
ласка, перевірте, що отриманий від виконавця звіт не містить вірусів
```

```
    </div>
```

```
</div>
```

```

<?php
    include_once 'utils/Constants.php';
    include_once 'entity/Block.php';
    global $FILE_SAVE_PATH;
    $isLoggedIn = isset($_COOKIE["login"]) && isset($_COOKIE["user_status"]);
    $is_file_required = isset($_GET["filename"]);
    $blocks = array();

    if ($is_file_required && is_readable($FILE_SAVE_PATH .
$_GET["filename"])) {
        $blocks[] = new CustomBlock("
            <a href = \"\" . $FILE_SAVE_PATH . $_GET["filename"] . \"\"
download>Завантажити файл</a>
            <a href = \"\" . $FILE_SAVE_PATH . $_GET["filename"] . \"\"
target=\"_blank\">Переглянути файл у новій вкладці</a>
            ", "contentBlock bigButton");

    } else {
        $blocks[] = new CustomBlock("Вибачте, ми не можемо знайти звіт,
        будь ласка, спробуйте пізніше: ", "contentBlock pink");
    }

?>

<?php foreach ($blocks as $block) : ?>
    <?= $block ?>
<?php endforeach ?>

<?php include_once 'footer.php'; ?>

```

Файл header.php

```

<!DOCTYPE html>
<html>
<head>

```

```

<meta charset="utf-8">
<title>Worker</title>
<link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Open+Sans:300,400">
<link rel="stylesheet" href="css/style.css">
</head>
<body>

<div class="logo">
  <h1><a href="index.php">Worker</a></h1>
  <i>The task distribution service</i>
</div>

<div class="navigation">
  <nav>
    <ul>
      <li class="navItem"><a href="index.php" >Головна</a></li>
      <li class="navItem"><a href="login.php" >Увійти</a></li>
      <li class="navItem"><a href="myProjects.php" >Мої задачі</a></li>
      <li class="navItem"><a href="feedback.php">Зворотній зв'язок</a></li>
    </ul>
  </nav>
</div>

```

Файл index.php

```

<?php include_once 'header.php'; ?>

<div class="contentBlock">
  <div class="contentHeader">
    Про проект
  </div>
  <div class="contentContent">
    Цей проект розробив Дмитро Голуб у рамках курсового проекту з
    технологій комп'ютерного проектування.
  </div>
</div>

```

Веб-додаток реалізує функціонал розподілу задач між виконавцями, а саме:

Керівник видає роботу виконавцям, про що у кожного виконавця в особовому кабінеті

з'являється персональне завдання. Виконавець виконує роботу та відправляє звіт

до цієї роботи. Керівник приймає або відхиляє звіт. Якщо звіт відхилено робітник

зобов'язаний виправити недоліки, та відправити звіт знову.

</div>

</div>

<div class="contentBlock important">

<div class="contentHeader">

Перелік функцій, що були реалізовані

</div>

<div class="contentContent">

Функції усіх користувачів

Вхід у систему (заглушка)

Форма зворотнього зв'язку

Перегляд загальної інформації

Функції керівника

Створювати завдання та вказувати виконавця

Переглядат надісланих виконавцем звітів

Прийняття або відхилення звітів

Коментування звітів

Функції виконавця

```

        <ul>
            <li>Перегляд отриманих завдань</li>
            <li>Надсилення звіту про виконання завдання</li>
            <li>Перегляд коментарів керівника</li>
        </ul>
    </li>
</ul>
</div>
</div>

<?php include_once 'footer.php'; ?>

```

Файл init.php

```

<?php

include_once 'database/MySQL.php';

$mySql = new MySQL();
$mySql->create_tables();

?>

```

Файл login.php

```

<?php include_once 'header.php'; ?>

<?php
include_once 'entity/Block.php';
include_once 'entity/User.php';
include_once 'database/MySQL.php';
include_once 'utils/Constants.php';

$db = new MySQL();
$blocks = array();
$is_just_logged_id = false;

```



```

$sis_unlogging = false;
global $COOKIE_LOGIN_TIME;

if (isset($_GET["login_as"]) && !strcmp($_GET["login_as"], "noUser")) {
    unset($_COOKIE['login']);
    unset($_COOKIE['user_status']);
    setcookie('login', null, -1, '/');
    setcookie('user_status', null, -1, '/');
    header("Refresh: 0; url=" . $_SERVER['PHP_SELF']);
    $sis_unlogging = true;
}

if(isset($_GET["login_as"]) && !$sis_unlogging) {
    try {
        $user = $db->login($_GET["login_as"], "");
        $user = $db->set_user_statuses(array($user))[0];
        setcookie("login", $user->getUsername(), time() +
$COOKIE_LOGIN_TIME);
        setcookie("user_status", $user->getStatus(), time() +
$COOKIE_LOGIN_TIME);
        $sis_just_logged_id = true;
        header("Refresh: 0; url=" . $_SERVER['PHP_SELF']);
    } catch (Exception $ex) {
        $blocks[] = new ContentBlock("Помилка",
            "Нажаль, сталась помилка під час отримання користувачів системи,
            будь-ласка, спробуйте пізніше: " . $ex->getMessage());
    }
}

$sisLoggedIn = isset($_COOKIE["login"]) && isset($_COOKIE["user_status"])
&& !$sis_unlogging;

if ($sisLoggedIn || $sis_just_logged_id) {
    $blocks[] = new ContentBlock("Оберіть обліковий запис",

```

```
"Ви вже увійшли у систему як " . $_COOKIE["login"] . ". Тип вашого  
облікового запису: "
```

```
$_COOKIE["user_status"].". Ви можете увійти у систему під іншим  
ім'ям");
```

```
} else {
```

```
$blocks[] = new ContentBlock("Увійдіть у систему", "Для подальшої  
роботи та перегляду ваших робіт,
```

```
будь ласка, увійдіть у систему", true);
```

```
}
```

```
try {
```

```
$user_arr = $db->get_all_users();
```

```
$user_arr = $db->set_user_statuses($user_arr);
```

```
foreach ($user_arr as $user) {
```

```
$blocks[] = new CustomBlock(
```

```
"<a href=\"?login_as=" . $user->getUsername() . "\">
```

```
Увійти як " . $user->getUsername() . " зі статусом " . $user->getStatus()  
 . "</a>", "bigButton"
```

```
);
```

```
}
```

```
$blocks[] = new CustomBlock(
```

```
"<a href=\"?login_as=noUser\">
```

```
Вийти з системи </a>", "bigButton"
```

```
);
```

```
} catch (Exception $e) {
```

```
$blocks[] = new ContentBlock("Помилка",
```

```
"Нажаль, сталась помилка під час отримання користувачів системи,  
будь-ласка, спробуйте пізніше");
```

```
}
```

```
?>
```

```

<?php foreach ($blocks as $block) : ?>
    <?= $block ?>
<?php endforeach ?>

<?php include_once 'footer.php'; ?>

```

Файл myProjects.php

```

<?php include_once 'header.php'; ?>

<?php
    include_once 'entity/Block.php';
    include_once 'database/MySQL.php';
    include_once 'entity/Task.php';
    include_once 'entity/User.php';
    include_once 'utils/Constants.php';

    $isLoggedIn = isset($_COOKIE["login"]) && isset($_COOKIE["user_status"]);
    $blocks = array();
    $db = new MySQL();
    $filter = Null;
    if (isset($_POST["filter"])) {
        $filter = $_POST["filter"];
    }

    if ($isLoggedIn) {
        include_once 'forms/sortingForm.php';
        global $EMPLOYER_USER_STATUS;
        if($_COOKIE["user_status"] === $EMPLOYER_USER_STATUS) {
            $blocks[] = new CustomBlock("
                <div class='flex_item_right'>Ви можете <a href = \"addTask.php\">додати
завдання</a></div>
                ", "contentBlock smallButton right_flex");
        }
    }

```

```

try {
    $user = $db->get_user($_COOKIE["login"]);
    $projects = $db->get_tasks($user, $filter);
    foreach ($projects as $task) {
        $blocks[] = new TaskBlock($task->getHeader(),
            $task->getDescription(), $task->getDatetime(), $task->getStatus(),
            $task->getId());
    }
    if (count($projects) == 0) {
        $blocks[] = new ContentBlock("В вас немає проєктів", "На даний час
для вашого облікового
запису нема жодного проєкту");
    }
}
catch (Exception $ex) {
    $error_block = new ContentBlock("Помилка", "Нажаль, під час обробки
запиту сталася помилка,
будь-ласка, спробуйте повторити спробу пізніше. " . $ex-
->getMessage());
    $error_block->setIsPink(true);
    $blocks[] = $error_block;
}
// add tasks to array
} else {
    $blocks[] = new ContentBlock("Увійдіть у систему", "Для подальшої
роботи та перегляду ваших робіт,
будь ласка, увійдіть у систему", true);
}
?>

<?php foreach ($blocks as $block) : ?>
    <?= $block ?>

```

```
<?php endforeach ?>
```

```
<?php include_once 'footer.php'; ?>
```

Файл showTask.php

```
<?php include_once 'header.php'; ?>
```

```
<?php
```

```
    include_once 'entity/Block.php';
```

```
    include_once 'database/MySQL.php';
```

```
    include_once 'entity/Task.php';
```

```
    include_once 'entity/User.php';
```

```
    include_once 'utils/Constants.php';
```

```
    global $EMPLOYER_USER_STATUS, $TASK_STATUS_COMPLETED,
    $FILE_SAVE_PATH, $TASK_STATUS_COMPLETED,
    $TASK_STATUS_REVIEW;
```

```
    $isLoggedIn = isset($_COOKIE["login"]) && isset($_COOKIE["user_status"]);
```

```
    $isApproved = isset($_GET["approve"]) && $_GET["approve"] === "true";
```

```
    $isChangesRequired = isset($_GET["approve"]) && $_GET["approve"] ===
    "false";
```

```
    $task_id = $_GET["id"];
```

```
    if(isset($_COOKIE["approve"])) { header('Location: /showTask.php'); }
```

```
    $isEmployer = $_COOKIE["user_status"] === $EMPLOYER_USER_STATUS;
```

```
    $blocks = array();
```

```
    $db = new MySQL();
```

```
    $isTaskActive = false;
```

```
    if (isset($_POST["comment"])) {
```

```
        try {
```

```
            $db->add_comment_raw($task_id, $_GET["reportId"],
    $_POST["comment"]);
```

```
        } catch (Exception $ex) {
```

```

        $blocks[] = new CustomBlock("Вибачте, неможливо додати коментар,  

        будь ласка, спробуйте пізніше: ". $ex->getMessage(), "contentBlock  

pink");
    }
}

if ($isApproved) {
    try {
        $db->set_task_status($task_id, $TASK_STATUS_COMPLETED);
        $blocks[] = new CustomBlock("Статус завдання змінено на ".  

$TASK_STATUS_COMPLETED, "contentBlock blue");
    } catch (Exception $ex) {
        $blocks[] = new CustomBlock("Вибачте, неможливо оновити статус  

задачі,  

        будь ласка, спробуйте пізніше: ". $ex->getMessage(), "contentBlock  

pink");
    }
}

if ($isChangesRequired) {
    try {
        $db->set_task_status($task_id, $TASK_STATUS_REVIEW);
        $blocks[] = new CustomBlock("Статус завдання змінено на ".  

$TASK_STATUS_REVIEW, "contentBlock blue");
    } catch (Exception $ex) {
        $blocks[] = new CustomBlock("Вибачте, неможливо оновити статус  

задачі,  

        будь ласка, спробуйте пізніше: ". $ex->getMessage(), "contentBlock  

pink");
    }
}

try{
    if (isset($_FILES["report"])) {

```

```

// Add file
$filename_postfix = substr($_FILES["report"]["name"], -4);
$filename_prefix = substr($_FILES["report"]["name"], 0,
strlen($_FILES["report"]["name"])-4);
$file_current_location = $_FILES["report"]["tmp_name"];
if ($_FILES["report"]["error"] != 0 || $_FILES["report"]["size"] == 0) {
    throw new Exception("File getting error");
}
$file_new_name = $filename_prefix . "-" . time() . $filename_postfix;
if (move_uploaded_file($file_current_location, $FILE_SAVE_PATH .
$file_new_name)) {

    $db->add_report_raw($task_id, $file_new_name);
    $blocks[] = new CustomBlock("Звіт ". $file_new_name .
        " успішно додано", "contentBlock blue");
    $db->set_task_status($task_id, $TASK_STATUS_REVIEW);
} else {
    throw new Exception("Cannot save file");
}

}
} catch (Exception $ex) {
    $blocks[] = new CustomBlock("Вибачте, ми не можемо додати звіт,
        будь ласка, спробуйте пізніше: ". $ex->getMessage(), "contentBlock
pink");
}

if ($isLoggedIn) {
    try { // load task by id
        $task = $db->get_task_with_status($task_id);

        if ($task->getStatus() != $TASK_STATUS_COMPLETED) {
            $isTaskActive = true;

```

```

    }

    $blocks[] = new TaskBlock($task->getHeader(),
        $task->getDescription(), $task->getDatetime(), $task->getStatus(), $task-
>getId());

    if ($task->getStatus() != $TASK_STATUS_COMPLETED) {
        if ($isEmployer) {
            $blocks[] = new CustomBlock("
                <div class='flex_item_right'>
                <a href = \" /showTask.php?id=\".$task_id."&approve=true\">Задачу
ВИКОНАНО</a>
                <a href =
                \" /showTask.php?id=\".$task_id."&approve=false\">Потребує правок</a>
                </div>
                ", "contentBlock smallButton right_flex");
        }

    } else {
        $blocks[] = new CustomBlock("Виконана задача перенесена в архів"
            , "contentBlock inactive");
    }

    $reports = $db->get_reports($task_id);

    foreach ($reports as $report) {
        $blocks[] = new ReportBlock
            ($report, $db->get_comments_for_report($report->getId()), $task_id,
            $isEmployer && $isTaskActive);
    }

    // TODO: load form for adding reports

```



```

        } catch (Exception $ex){
            $blocks[] = new CustomBlock("Вибачте, ми не можемо отримати дані
задачі,
            будь ласка, спробуйте пізніше: ". $ex->getMessage(), "contentBlock
pink");
        }
        // Load task from database
        // Load reports from database
        // Load comments for reports

    } else {
        $blocks[] = new ContentBlock("Увійдіть у систему", "Для подальшої
роботи та перегляду ваших робіт,
        будь ласка, увійдіть у систему", true);
    }

?>

<?php foreach ($blocks as $block) : ?>
    <?= $block ?>
<?php endforeach ?>

<?php
if($isLoggedIn && !$isEmployer && $isTaskActive) {
    include_once 'forms/addReport.php';
}
?>

<?php include_once 'footer.php'; ?>

```