МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____Аліна САВЧЕНКО

"____"_____2021 р.

# ДИПЛОМНА РОБОТА

**(ПОЯСНЮВАЛЬНА ЗАПИСКА)**

*ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ*
**"МАГІСТРА"**

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ "ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ"

**Тема: "Програмна система task-менеджер для керування IT проектами"**

**Виконавиця:** Возниця Анастасія Сергіївна

**Керівник:** к.т.н., доцент Савченко Аліна Станіславівна

**Нормоконтролер:** _____ Ігор РАЙЧЕВ

**Київ - 2021**

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра Комп'ютерних інформаційних технологій

Галузь знань, спеціальність, освітньо-професійна програма: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології".

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Аліна САВЧЕНКО
«____»_____2021р.

# ЗАВДАННЯ

**на виконання дипломної роботи студентки**

Возниці Анастасії Сергіївни
(прізвище, ім'я, по батькові)

**1. Тема роботи:** «Програмна система task-менеджер для керування IT проектами» затверджена наказом ректора від 12.10.2021 за № 2228/ст.

**2. Термін виконання роботи:** з 12.10.2021 по 31.12.2021.

**3. Вихідні дані до роботи:** основи проектного менеджменту, ключові підходи та методології керування IT проектами, розробка програмної системи керування задачами.

**4. Зміст пояснювальної записки**: вступ, аналітичний огляд і постановка завдання, розгляд особливостей керування проектами, дослідження технологій та засобів, розробка програмного продукту керування задачами, оцінка якості технології, висновки.

**5. Перелік обов'язкового ілюстративного матеріалу:** слайди, презентація.

**6. Календарний план-графік**

| *№ n/n* | ***Завдання*** | ***Термін виконання*** | ***Підпис керівника*** |
|---|---|---|---|
| 1. | Отримання завдання на дипломну роботу та побудова плану-графіку виконання робіт | 12.10.21 – 15.10.21р. | |
| 2. | Огляд та аналіз відомих програмних архітектур і патернів | 16.10.21 – 19.10.21р. | |
| 3. | Аналіз літератури та джерел за темою дипломного проекту | 20.10.21 – 24.10.21р. | |
| 4. | Проведення консультації з науковим керівником щодо плану розділів. | 25.10.21 – 31.10.21р. | |
| 5. | Розробка розділу 1 | 01.11.21 – 07.11.21р. | |
| 6. | Розробка розділу 2 | 08.11.21 – 17.11.21р. | |
| 7. | Розробка розділу 3 | 18.11.21 – 01.12.21р. | |
| 8. | Оформлення та друк пояснювальної записки | 02.12.21 – 11.12.21р. | |
| 9. | Оформлення супровідних документів. Створення презентації, доповіді та підготовка до захисту дипломної роботи | 12.12.21 – 20.12.21р. | |

**7. Дата видачі завдання:** 12.10.2021р.

**Керівник дипломної роботи** _____ Аліна САВЧЕНКО
(підпис керівника)

**Завдання прийняла до виконання** _____ Анастасія ВОЗНИЦЯ
(підпис випускника)

# РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Програмна система task-менеджер для керування IT проектами» складається із вступу, трьох розділів, загальних висновків, списку використаних джерел і містить 100 сторінок тексту та 47 рисунків. Список використаних джерел містить 15 найменувань.

***Метою*** дипломної роботи є дослідження особливостей та практик керування проектами та розробка безкоштовної спрощеної програмної системи базового функціоналу task-менеджер для керування IT проектами на компанії з розробки ПЗ.

***Об'єктом дослідження*** є процес керування IT проектами, супроводження програмних систем та task-менеджери.

***Предметом дослідження*** є автоматизація роботи проектного менеджера та його команди за допомогою найкращих практик та ряду програмних систем.

**Ключові слова:** PM, WORKFLOW, КЕРУВАННЯ ПРОЕКТАМИ, ЕФЕКТНИВНІСТЬ, TEAM WORK PRINCIPALS, ДЕЛІВЕРІ-ПРОЦЕС, SDLC, МОДЕЛІ КЕРУВАННЯ, АРХІТЕКТУРА.

**ЗМІСТ**

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

***ПЗ*** – Програмне забезпечення

***ТЗ*** – Технічне завдання

***PMBOK*** – Основний довідник PM

***PM*** – Проектний менеджер

***BA*** – Бізнес аналітик

***DM*** – Департмент менеджер

**PRG.M** – Програмний менеджер

***SDLC*** – Життєвий цикл програмного забезпечення

***KPI*** – Ключові показники ефективності

***ARCH.L*** – Архітектор проекту

***DEV.L*** – Розробник програмного забезпечення

***AGILE*** – Філософія гнучкої розробки програмного забезпечення

***SCRUM*** – Підхід управління проєктами для гнучкої розробки

***KANBAN*** – Метод управління розробкою

***APF*** – Адаптивні рамки проекту

***QRM*** – Швидкореагуюче виробництво

***SQL*** – Декларативна мова програмування для взаємодії користувача з базами даних

***CSRF*** – Міжсайтова підробка запиту

***XSS*** – Міжсайтовий скриптинг

***FLASK*** – Мікрофреймворк для веб-додатків

***DJANGO*** – Високорівневий відкритий Python-фреймворк для розробки веб-систем

# ВСТУП

В дипломній роботі проведено дослідження особливостей та практик керування проектами, а також запропоновано проект безкоштовної спрощеної програмної системи базового функціоналу task-менеджер для керування IT проектами на компанії з розробки ПЗ.

Зважаючи на постійно зростаючий рівень навантаження на керівників та проектні команди доцільною є розробка програмної системи, яку буде легко адаптувати під більшість проектів та завдяки ній буде можливо сформувати для всіх учасників процесу оптимальні графіки робіт. Це дозволить підвищити ефективність, якість роботи та зменшить час на розробку програмного забезпечення.

Використання сучасних програмних технологій дозволить сформувати оптимальний розклад навантаження, враховуючи специфічні особливості діяльності кожної людини. Реалізація даного проекту – це крок у напрямку тотального впровадження інформаційних технологій в усі сфери людського буденного та професійного життя.

Теоретичні дослідження роботи зосереджені на аналізі методів формування розкладу навантаження при виконанні проекту, які були використані при розробці програмних аналогів. Основна мета теоретичного дослідження полягає у розробленні алгоритму формування плану зменшення навантаження на проекті та врахуванню можливих ризиків.

Експериментальні дослідження полягають у перевірці адекватності розробленого алгоритму формування розподілу проектних задач з урахуванням специфіки діяльності для продукту, шляхом тестування системи та її аналізу.

Наукова новизна полягає у створенні універсального продукту, який буде легко адаптувати під будь-який проект, звести до мінімуму витрати на організаційну діяльність та зменшити навантаження на виконавців.

Практична значимість полягає у розробленні програмної системи керування з урахуванням специфіки проектної діяльності.

Використання розробленої системи дозволить оптимальним чином розподілити задачі людини для контролю енергетичних витрат та уникнення "вигорання".

В основі розробки полягає каркас, який містить різні шари, причому кожен шар включає в себе декілька модулів, а саме модулі складаються з патернів і використовують конкретні патерни. Для реалізації модулів необхідно побудувати підсистему подальшого редагування вже сгенерованих архітектур для створення зв’язків між модулями і шарами, причому потрібні можливості редагувати власне архітектури, дублювати, добавляти, видаляти та створювати нові архітектури.

Система повинна мати такий базовий функціонал: сторінки входу та контролю доступів, панель приладів та підсистема навігації, графа проектів та інструментів, метрики для планування проектів, канбан дошка, календар подій, контакт лист та чат.

За час існування ІТ-галузі було створено безліч методів  задля ефективного керування проектами. Більша частина з цих методів заснована на технічних аспектах роботи команди. Для того, щоб створити дійсно практичний продукт та недорогий продукт, треба сформувати та розкрити бачення на створення універсального підходу до розробки системи з погляду керування і цінності, яку може принести цей процес проекту і клієнту, а також на сам фреймворк як управлінський інструмент.

В результаті проведеної роботи буде отримано систему, яку можна буде повноцінно використовувати в якості програми для планування власного часу чи часу проектної команди.

# РОЗДІЛ 1
# ПРОЕКТНИЙ МЕНЕДЖМЕНТ. ОСОБЛИВОСТІ РОБОТИ З ІТ ПРОЕКТАМИ

## 1.1. Особливості ІТ проектів. Характеристики та класифікації.

Основною відмінністю ІТ-проектів від проектів, що реалізуються в інших сферах людської діяльності, наприклад, у будівництві чи виробництві є те, що проектне керування в ІТ має справу з невловимими результатами в інформаційному просторі. Крім того, ІТ-проекти мають ряд властивих лише їм факторів, що впливають на успішність виконання проекту.

У зв'язку з тим, що результат ІТ-проекту (за впровадження ІВ) невловимий – його не можна виміряти в загальноприйнятих одиницях виміру (кілограми, метри, секунди), представити у просторі, фізично відчувати – вимоги до результатів проекту та планування робіт мають бути максимально детальними. На відміну від будівельних та виробничих проектів у ІТ-проектів немає нормативів витрат для типових операцій, та й самі типові операції є лише дуже укрупнено і в рамках подібних проектів [1].

Враховуючи те, що в даний час інформаційні технології в принципі є елементом отримання конкурентної переваги, багато організацій намагаються впровадити інформаційні системи в якомога стисліші терміни, не проводячи детального планування та постановки завдання з інформатизації, що вкрай негативно позначається на результатах проекту. Можна сказати, що особливість ІТ-проектів щодо проектів, що реалізуються в інших сферах діяльності, у їх підвищеній складності та вищому ступені ризику. Складність ІТ-проекту залежить від наступних факторів:

**Організаційний обсяг проекту**

Організаційний обсяг проекту стосовно IT-проектів з впровадження інформаційних систем визначає кількість відокремлених організаційних одиниць, в яких буде здійснено впровадження/тиражування інформаційної системи. Під відокремленою організаційною одиницею у разі розуміється і окрема, як юридична особа організація, і окремий підрозділ у межах конкретної організації – юридичної особи. Показник практично лінійно впливає вартість робіт. Розмір і структура організаційного обсягу впливає як вибір методів керування організаційним обсягом, способів координації робіт, і інші аспекти проекту.

***Функціональний обсяг проекту***

Даний показник характеризує набір функціональних можливостей інформаційної системи, що входять до складу рішення, що впроваджується. Також можна сказати, що функціональний обсяг – сукупність бізнес-процесів, функцій та операцій, інформатизація/автоматизація яких передбачається рахунок впровадження інформаційної системи. Функціональний обсяг проекту впровадження інформаційної системи – найважливіший з метою оцінки складності проекту показник, т. до. він становить основну частину змісту робіт проекту, визначає об'єкт автоматизації – бізнес-процеси організації. Його зміна тягне за собою пропорційну зміну інших показників проекту, таких як методологічний обсяг, інтеграційний обсяг, вартість проекту тощо.

***Методологічний обсяг проекту***

Методологічний обсяг проекту описує набір нормативно-регламентуючої документації, яку потрібно розробити чи доопрацювати під час реалізації проекту. До такої документації можна віднести регламенти виконання процесів і операцій, що автоматизуються, інструкції користувачів, методики виконання конкретних операцій і процесів та ін.

***Забезпечення інформаційної безпеки***

Інформаційна система, що впроваджується, може містити дані, що представляють комерційну таємницю організації, або дані, які відповідно до законодавства повинні бути захищені, наприклад фінансові дані або персональні дані.

Перелічені основні фактори впливають на складність IT-проектів щодо впровадження інформаційних систем та присутні у всіх проектах цього виду, проте остаточний набір факторів, їх вага та критичність залежать від специфіки кожного конкретного проекту.

### 1.2. Підсистеми, функції та методи керування IT проектами

Для того, щоб отримати стабільні позитивні результати, проект повинен бути не складним у керуванні. Створення таких проектів потребує певного впливу:

- До нього повинні бути залучені замовники проекту. Тобто люді, для яких він має найвищу цінність.
- Необхідно розрахувати вірогідність успішності даного проекту з економічної точки зору.

Тобто з цього отримаємо, що керування проектом – це процес координації та передбачення якості роботі людських та матеріальних ресурсів у єдиному тандемі за допомогою певних технік та практик протягом всього його життєвого циклу. Завершенням можна вважати досягнення поставленої мети чи визнання всіма ведучими керуючими органами проекту, що це не можливо [1].

Контроль проекту заснований на системному ставленні до цілей. Це може буде зроблене за допомогою команди проекту. Методі у цьому випадку виступають як складова у процесах керування.

Для того, щоб краще зрозуміти процес керування гарним вибором може слугувати системно орієнтована модель координації.

Розглянемо такі секції:

- Керування цілями проекту (стратегічне);
- Керування діяльністю (оперативне);
- Керування існуючою системою (інструментальне);

Етапи керування:

- Початок та побудова;

- Процес керування та узгодження;
- Криза;
- Завершення проекту.

Ознаки проекту: мета, засобі досягнення, час на виконання та бюджет (лімітованій чи ні).

Проектний аналіз – це об’єднання методів та прийомів для розробки та оцінки проекту.

Досвід розвинених країн каже про те, що система керування проектами – гарний метод, щоб вийти з економічної кризи та знайти відповіді для вирішення наукових, виробничих та соціальних складнощів.

Люді, що напряму пов’язані з керуванням проектами згодом змогли об’єднати всі необхідні для їх роботи національні та міжнародні організації, все що може мати свій особистий ринок ПЗ [3].

### 1.2.1. Функції керування ІТ проектами

Управлінські функції мають певну важливу діяльність, яку треба робити лише спираючись, що вони відомі всім учасникам процесів . Якщо це не так, то потрібно звернути увагу та виправити отримане непорозуміння.

До функцій керування проектом відносяться:

- Створення плану;
- Робота з контролюванням процесів;
- Рахування вірогідності та аналіз;
- Отримання та винесення рішень;
- Обробка моніторингу;
- Оцінка результатів;
- Створення звітів;
- Експертність;
- Тестування результатів;

- Адміністрування процесів та людей.

### 1.2.2. Підсистеми керування IT проектами

Можна впевнено сказати, що підсистеми керування проектами залежать від складу предметних областей та елементів певного проекту. Предметні області та керовані елементи зазвичай мають: терміни виконання, ресурси завдяки яким це буде здійснено, кошторис, надання послуг, правки, ризики та ін. Висвітлені підсистеми мають ведуче місце практично у будь-якому проекті. Деколи використовуються специфічні підсистеми.

Є дуже велика різниця між підсистемами та функціями керування проектами. Вона полягає у тому, що перші напрямлені на предметну область, а другі - на процеси. Керування підсистемою виконує всі потрібні за ТЗ функції. Підсистеми мають розділення [3].

***Підсистема керування змістом***

Керування предметною областю визначає ті процеси, що відповідають за включення до проекту критичних активностей.

Предметна область проекту визначається, коли є:

- Цілі;
- Властивості та структура продукту (послуги);
- Роботи, які мають бути виконані для отримання продукту (послуги).

Напрями діяльності (процеси):

- Розробка концепції;
- Визначення предметної галузі проекту;
- Розподіл робіт;
- Встановлення звітності;
- Введення системи контролю;
- Завершення проекту.

***Підсистема керування часом***

Керування часом включає процеси, які забезпечують своєчасне завершення проекту на основі розробленого розкладу, а саме:

- Визначення складу операцій для здобуття різних результатів проекту;
- Визначення взаємозв'язку операцій;
- Оцінка тривалості операцій;
- Складання розкладу виконання проекту з урахуванням тривалості операцій та потреби в ресурсах;
- Керування розкладом – керування змінами розкладу проекту.

Напрями діяльності (процеси):

- Планування часу у проекті;
- Оцінка тривалості;
- Календарне планування;
- Контроль часом у проекті.

***Підсистема керування вартістю***

Керування вартістю включає процеси, які забезпечують завершення проекту в рамках встановленого бюджету.

Основні методи та інструменти керування – методи визначення ефективності інвестицій, мережевий графік

Напрями діяльності (процеси):

- Оцінка та прогнозування вартості;
- Бюджет та кошториси;
- Контроль вартості;
- Функціонально-вартісний аналіз.

***Підсистема керування якістю***

Керування якістю включає ті процеси, які забезпечують задоволення потреби, заради якої було розпочато проект.

Основні методи та інструменти – стандарти, нормативи, вимоги, система контролю та підтримки.

Напрями діяльності (процеси):

- Планування якості – визначення стандартів якості, що відповідають проекту, та засобів реалізації цих стандартів.
- Підтвердження якості – регулярна загальна оцінка виконання проекту з метою підтвердження того, що проект задовольняє прийняті стандарти якості.
- Керування якістю – контроль з метою перевірки відповідності проміжних результатів прийнятим стандартам якості та визначення шляхів усунення причин незадовільного виконання.

***Підсистема керування людськими ресурсами***

Керування людськими ресурсами включає процеси, необхідні найефективнішого використання людей проекті.

Основні методи та інструменти керування – неформальні (психологічні та соціологічні) у таких аспектах: службові відносини, оцінка та оплата праці, трудове законодавство та регулювання, зовнішні учасники проекту, члени команди проекту, команда проекту.

Напрями діяльності (процеси):

- Організаційне планування – визначення, документування та призначення проектних ролей, відповідальності та відносин звітності.
- Призначення персоналу – залучення необхідних людських ресурсів до роботи у проекті.
- Розвиток команди – розвиток індивідуальної та групової компетентності з метою покращення виконання проекту.

***Підсистема керування комунікаціями***

Керування комунікаціями (взаємодіям) має у своїй основі процеси, що є обов’язковими для якісної підготовки, об’єднання, розподілу, збереження та кінцевої обробки проектної інформації. Можливо досягти завдяки зв'язкам між колегами, ідеями, матеріалами, які вимагаються для досягнення успіху.

Головними методами та інструментами є:

- Формальні (інформаційні технології);
- Неформальні (міжособистісні контакти).

Напрями діяльності (процеси):

- Взаємодія - відокремлення вимог учасників та кураторів проекту: кому, коли, у якому вигляді і яка інформація необхідна
- Розподіл – забезпечення для всіх учасників проекту отримання потрібної інформації
- Звітність – отримання та розподіл інформації з приводу виконання роботи
- Завершення – підготовка та видача інформації з метою формування завершальної фази чи проекту.

***Підсистема керування ризиком***

Однією з найважливіших є підсистема керування ризиком (вплив на проект непередбачених подій) – це процес визначення та створення плану реагування на ризики. Він бере за основу розрахунок вірогідності успіху та мінімізацію ймовірних проблем з планом закриття у випадку складнощів.

Ризик має такі характеристики: вірогідність появи негативних подій та оцінка пошкоджень.

Головні методі – формальні (розрахунки, передбачення).

Напрями діяльності (процеси):

- Визначення плану по усуненню;
- Ідентифікація вірогідності та характеристик;
- Оцінка якості впливу;
- Отримання даних щодо ймовірності та наслідків;
- Створення процедур реагування;
- Моніторинг та контроль ризиків, що залишилися, отримання інформації щодо нових.

## 1.3. Життєвий цикл та організація IT проекту

Кожна організація, чи то стартап, чи міжнародна корпорація, зацікавлена в успішній реалізації проектів. І хоча кожному з них потрібен індивідуальний підхід, всі проекти мають схожу структуру.

Іншими словами, незалежно від того, яку методологію вибрано для керування проектом, у кожного з них завжди буде початок, середина і завершення. Це життєвий цикл проекту [6].

Життєвий цикл проекту — це послідовність етапів, якими проходять проекти від ініціації до завершення незалежно від своїх специфіки.

Чітке розуміння цих фаз дозволяє менеджерам та керівникам максимально ефективно контролювати проекти. Метою життєвого циклу є створення простої у використанні структури для керівництва та керування проектами.

Життєвий цикл проекту допомагає:

- Поліпшити комунікацію між командою та замовниками;
- Бути впевненим, що мета доступна за допомогою доступних ресурсів;
- Керувати ризиками та мінімізувати їх.

***Фази життєвого циклу проекту***

За даними Project Management Body of Knowledge, або PMBOK, життєвий цикл складається з п'яти фаз [6]:

- Ініціація;
- Планування;
- Виконання;
- Контроль;
- Завершення.

Як правило, етапи циклу йдуть послідовно, один за одним. Але буває інакше. Якщо під час реалізації з'являються зміни, можна повернутися на стадію планування, щоб скоригувати роботу команди проекту у майбутньому.

Розглянемо кожну фазу життєвого циклу окремо.

***Ініціація***

Ініціація - це старт роботи над концепцією, підготовка до її планування та реалізації. Для початку визначте, яке завдання стоїть перед командою та чи допоможе ідея вирішити проблему. Якщо відповідь позитивна, приступайте до написання концепції та економічного обґрунтування, а також пошуку партнерів.

Завданням цього етапу є визначення загальних цілей, реалізація яких приведе кожну зі сторін до бажаного результату.

Після досягнення угод необхідно зафіксувати основні тези та домовленості у статуті проекту. Статут — це формальний, короткий документ, який описує проект. Він є важливою складовою планування, оскільки використовується протягом усього життєвого циклу проекту та допомагає вирішити всі спірні моменти протягом робочого процесу.

***Планування***

Коли документи підписано та умови фінально затверджені зацікавленими сторонами, починається стадія планування

На етапі планування менеджер розбиває робочий процес на дрібні завдання, створює команду, розподіляє ролі, щоб, зокрема, успішно керувати командою, розробляє покрокову послідовність виконання завдань і терміни. Щоб завдання мали більше шансів на успіх, важливо переконатись, що для кожної з них вистачає ресурсів.

Коли готовий календарний графік, позначені ключові ролі та зони відповідальності, виявлено можливі ризики та шляхи їх запобігання, а бюджет сплановано, настає час для організаційної наради.

У ході цієї зустрічі менеджер представляє проект та його цілі, обговорює з командою найважливіші етапи плану, відповідає на запитання, а також знайомить усіх із інструментами, які команда використовуватиме у процесі роботи. Після загальних зборів кожен із членів команди повинен мати чітке уявлення про проект у цілому, його етапи та їх реалізацію.

Для цього після наради рекомендується надати учасникам постійний доступ до проекту. Так, члени команди зможуть постійно знати справи та зміни в ході робочого процесу.

***Виконання***

Тепер, коли проект затверджено, команда сформована та готова розпочинати справу, робочий процес переходить до фази виконання.

Завдання менеджера на цьому етапі – проконтролювати синхронний запуск роботи всіх відділів та переконатися, що кожен виконує своє завдання.

***Контроль***

Ця фаза, зазвичай, збігається з фазою ініціації. Тому що для досягнення поставленої мети та максимально успішного завершення проекту недостатньо лише благополучно запустити робочий процес. Керівнику необхідно постійно стежити, щоб команда слідувала початковому плану.

Тому у цій фазі життєвого циклу менеджер контролює ресурси та своєчасне виконання завдань, координує учасників команди, оперативно вносить правки до плану проекту у разі непередбачених обставин.

Зміни щодо цієї стадії — абсолютно нормальне явище. Гнучкість у разі навіть грає на руку. Адже що раніше виявлено проблему, то швидше вона вирішиться.

***Завершення***

Це останній етап, що означає його офіційне закінчення. Але не варто відразу ж вмивати руки та перемикатися на такі завдання. Для того, щоб залишити приємне враження від професіоналізму та роботи команди, досвідчений менеджер:

Здасть проект разом із документацією клієнту чи команді, яка його вестиме в майбутньому.

Проведе "роботу над помилками". На цій фінальній зустрічі команда здобуде корисні уроки з успіхів та невдач під час роботи над проектом.

Повідомить про успіх заходу клієнтам та керівникам, які були зацікавлені результатами команди.

Розмістить документацію у централізованому сховищі для легкого доступу до неї у майбутньому.

***Наразі проект офіційно завершено***

Часом керування проектами видається справою непосильною. Однак, поділ проекту на п'ять окремих фаз, які називаються життєвим циклом проекту, допоможе команді розумно витрачати час та ресурси, що підвищує шанси на успіх проектів будь-якої величини та складності.

### 1.4. Робота проектного менеджера. Процес керування проектом

Проектний менеджер – це спеціаліст, який керується головним завданням: загальний контроль проекту. Його рутинними завданнями є розподіл пріоритетів, планування виконання роботи, розділення та розподілення завдань, контроль взаємодії між працівниками, налагодження комунікації та швидке вирішення проблем [4].

Основним завданням та зоною відповідальності менеджера є доведення ідеї клієнта до описаної реалізації з витримкою часу та використанням обраних ресурсів.

Після отримання завдання PM'у необхідно налагодити процеси розробки, розподілити задачі по команді, налагодити процес розробки, створити правильний настрій, налагодити зв’язок між командами і клієнтом, передбачити та запобігти знищенню перешкод для команд, слідкувати за часом, якістю та систематичністю поставок.

Обов'язки PM'а:

- Робота з проектною документацією;
- Створення та аналіз плану проекту;
- Обговорення часу на виконання задач та регулярність дзвінків-презентацій для клієнта;
- Аналіз, усунення або створення плану нейтралізації можливих ризиків;
- Проведення співбесід з потенціальними кандидатами на ролі в команді;
- Розбивка проекту на задачі та розподіл між виконавцям;
- Відокремлення так контроль необхідних ресурсів та розподіл для команди;
- Налагодження робочих процесів (розробка, тестування, робота з вимогами);
- Розстановка пріоритетів виконання задач (для відповідності термінам та якості);
- Контроль роботи команди;
- Нотування та контроль стану проекту та виконання окремих завдань;
- Відстеження правильної пріоритетності виконання роботи над задачами;

- Відстеження стану кожного окремого розробника;
- Стабільна мотивація та надихання команди на роботу;
- Утворення довірливих відносин з кожним з учасників проекту;
- Відстеження задоволеності команди проектом та технологіями;
- Вирішення та зменшення конфліктних ситуацій всередині команди і в тандемі замовник-команда;
- Керування очікуваннями клієнта;
- Створення звітів для клієнта з метою надання статусу виконання проекту;
- Допомога команді з презентацією замовнику виконаного функціоналу або демо-версій;
- Інтерв'ювання та оновлення команди.

У рамках плану керування проектами робочий процес відноситься до серії кроків, які необхідно зробити для виконання завдань, і способів переміщення між ними.

Проект, як правило, є одноразовим підприємством, обмеженим часом, яке служить дуже конкретній меті і призводить до дуже конкретного результату [4].

Робочий процес керування проектом — це процес упорядкування завдань і дій між ключовими етапами в ефективну та значущу послідовність.

Успішне виконання окремих завдань зазвичай вимагає, щоб проекти проходили через серію етапів. Незалежно від розміру та складності, усі окремі проекти мають власні схеми робочого процесу.

Ефективні системи керування проектами повинні допомогти вам організувати та записувати етапи робочого процесу. Хороший робочий процес створить ряд кроків, які спрямовують команду до кінцевого пункту призначення. Найефективніша та найкраще освітлена доріжка забезпечить максимально плавну їзду.

### 1.5. Підхід до створення універсальної системи керування проектами

Для людини, яка керує проектами (PM/BA/DM/Prg.M тощо) система керування – це життєво необхідний інструмент ефективної організації роботи.

Система керування проектами розглядається як спосіб ефективної організації проектної діяльності, з цього випливає, що вона повинна відповідати на такі запитання [5]:

- Що нас об’єднує як команду? Це цінності (project core value) згідно з особливостями клієнту та проекту.
- Яким чином можливо втілити та використати для проектної мети ці цінності? Це планування головних компонентів командної роботи (framework core).
- Наскільки гнучкі у своїй проектній діяльності та наскільки готові до змін? Це про навичку адаптуватись до проектної роботи.

Розглянуті компоненти та характеристики — це базис ефективного керування проектною діяльністю задля успішності проекту.

Також існують проектні політики для управлінця — це інструмент задля затвердження базових домовленостей (basic agreements) з командою та клієнтом. Вони допомагають визначити межі співпраці та отримати базові правила для розробки та досягнення цілей. Саме завдяки ним знаходять рішення, роблять зовнішнє і внутрішнє оцінювання доречності виконання задач або алгоритму дій, до них також часто звертаються під час виникнення проблемних ситуацій. Саме через це вони є важливими.

### 1.5.1. Принципи командної роботи (teamwork principals)

У дійсно ефективної проектної команди завжди є робоча ідеологія. Основою напрацювань можуть слугувати пропаговані раніше принципи роботи у якості повсякденних звичок [7]. Інколи це допомагає визначити головні мотиваційні тригери команди. Люди – це головний капітал компанії. Можна сказати, що фактично команда виступає не лише носієм надання проектних послуг, а й складовою для створення плану та культури майбутньої взаємодії на роботі. Успішність співпраці з замовником напряму залежить від єдності цілей та поглядів команди [2].

Взаємозв’язок об’єктів проектної роботи (project context diagram)

Для визначення проектної специфіки треба зрозуміти поведінку об’єктів, які буде використано в роботі. Саме через це команді необхідна така модель у постійно актуальному стані. Це подарує розробникам можливість мати:

- Актуальну інформацію щодо проектного руху всього проекту та подальшого його розвитку, що також потрібне для контролю змін стратегії.
- Можливості створення звітів заснованих на актуальних даних для клієнта.

***Показники ефективності, моделі делівері-процесу (Delivery model based on SDLC type and SDLC KPI’s)***

Дана модель заснована на базі гнучкого для клієнтської специфіки життєвого циклу розробки ПЗ, що значно спрощує компанії та клієнту планування, надає можливість скоріше розуміти та менеджити поставки функціоналу. На ефективність та загальну діяльність команди будуть впливати структура та специфіка деталей кожного етапу делівері та моніторингу процесів [7].

Головні метрики ефективності делівері процесу:

- Якість роботи команди разом
- Правильність розподілу часу
- Ріст проекту та його членів
- Рівень задоволеності співпрацею клієнта
- Відповідність вимогам і стандартам якості.

### 1.5.2. Робота команди та її найкращі практики (Team workflow and Practices)

Опис деталей процесів роботи команди є необхідним, якщо потрібне чітке розуміння та правильне координування дій усіх учасників. Цей принцип можна вважати незамінним у якості координаційного операційного рівня для правильної робочої взаємодії. Описова модель використовується для забезпечення прозорості, чіткості та легкого прогнозування дій при розробці. Це важливо для всіх учасників проекту: від менеджерів до клієнта [8].

Структурна модель проектних активностей і їх артефактів (Project activities structure model)

Проектні активності – це головний компонент для структурування дій для всіх членів проектної команди. Базисом можна вважати формування блоків, що надають перелік деталей та артефактів. Тобто результат стає очевидним ще на етапі онбордингу.

Найкращі командні практики роботи напрямлені на результат та ефективне налаштування процесу. Прозорість та контрольованість лежать у основі будь-яких активностей.

***Створення імплементаційної групи***

Для введення в роботу фреймворку перш за все треба створити імплементаційну групу. В її складі будуть: програм-менеджер (Prg.M), деліверi-менеджер (DM), проєктний менеджер (PM), бізнес-аналітик (BA), архітектор (Arch.L), девелопмент-лід (Dev.L).

Даний концепт забезпечує:

- Широту експертних думок;
- Створення і рух необхідних ініціатив;
- Контроль утворення необхідного базису досвіду керування проектами;
- Роботу з джерелом синергії взаємодії у проєктній команді.

Але треба розуміти, що підтримка проектного фреймворку (project framework maintenance) має у своїй основі певні особливості [8]:

- При впроваджені фреймворку має бути наявним спеціаліст, який може бути проектним методологом.
- Чітке планування дій – це базис для впровадження нового підходу. Це повинні пам’ятати і команда, і клієнт. Потрібно з самого початку виділити час на цей етап.
- Обов’язково треба врахувати гнучкість реакції на зміни. Так як у будь-якому проекті така активність є обов’язковою.
- З самого початку операції впровадження нового фреймворку треба врахувати, що повинні бути створені інструкції щодо змін та поширені з командою.

- Фреймворк-методолог повинен забезпечити, щоб вся команда мала стабільний зворотній зв’язок з замовником для коригування дій із впровадження підходу.

***Складнощі при впровадженні фреймворку***

Робота з проектами – це зазвичай виклик для команди та управлінця, бо не існує двох ідентичних проектів. З цього випливає, що просто не можливо створити проектний фремворк, що стане панацеєю для будь-яких задач. Впровадження стає головним викликом на початку роботи з проектом. Його можна поділити на частини:

- Обов’язковим є навчання проектних команд. Здебільшого фахівці мають різноманітний досвід роботи з проектами та ризиками.
- Потрібне гарне формування єдиного бачення у команди та клієнта майбутніх задач, які будуть занесені до цього фреймворку.
- Фреймворк повинен бути гнучким, щоб у випадку вдосконалень та змін робота не ставала на паузу. Такі процеси вимагають тотальної уваги до деталей та повної залученості імплементаційної групи. Але це є не простим завданням з погляду навантаження. Головні складові – це витрачений час та отримане на команду навантаження.

## 1.6. Постановка задачі

Для забезпечення будь-якого проекту системою керування задачами потрібно зробити базову модель з відкритим для вдосконалень кодом, з чого випиває, що це практично апробована модель.

Для реалізації ідеї вдосконалення правильним буде рішення керування такими принципами:

- В пріоритеті повинно знаходитись забезпечення комплексної управлінської візії
- Базисом повинен бути аналіз ланцюжка каскаду наслідків реалізації змін.
- Інструменти розробки повинні бути універсальними.
- Повинні бути альтернативні погляди на імплементацію фреймворку.

Серед всіх можливих питань з приводу універсального фреймворку найважливішими можна вважати:

- Розгляд впровадження та створення фреймворку в різних доменних експертизах.
- Ідеї та досвід впровадження проектного фреймворку.
- Визначення можливих шляхів розвитку отриманого фреймворку.

Саме такі теми можуть допомогти у побудові та обиранні методів застосування за принципом єдності сприйняття  запропонованої практичної моделі.

Враховуючи наведене вище, актуальною є дослідження особливостей та практик керування проектами.

Метою дипломної роботи є розробка безкоштовної спрощеної програмної системи базового функціоналу task-менеджер для керування IT проектами на компанії з розробки ПЗ. Для досягнення поставленої мети необхідно проаналізувати та вирішити наступні задачі:

- Провести аналітичний огляд особливостей та практик керування IT проектами;
- Визначити критерії оцінки їх ефективності;
- Розробити архітектуру програмної системи task-менеджер;
- Розробити інфтерфейс системи.

Функціонал програмної системи task-менеджер для керування IT проектами має реалізує такі можливості:

- Додавати всі необхідні файли, посилання на матеріали, зауваження, інструкції, коментарі до поставлених завдань.
- Нагадування про закінчення терміну проекту чи окремих його етапів розсилається автоматичною системою кожному з учасників процесу та зокрема відповідальним особам персонально.
- Зберігати історію роботи над поставленими завданнями: кожен із учасників процесу може публікувати коментарі та коригування в ході виконання робочих завдань, а також простежити історію внесення змін та осіб, які їх вносили.

- Відстежити трудовитрати кожного з працівників та оцінити ефективність роботи всього персоналу.
- Можливість обмінюватися миттєвими повідомленнями під час роботи над проектом.
- Розсилання інформації про закінчення чергового етапу роботи електронною поштою.
- Зберігання інформації та історії щодо кожного з проектів, включаючи співробітників, задіяних у роботі, їх трудовитрати, час на реалізацію, загальна кількість проектів у яких брав участь кожен окремий співробітник, динаміка виконання проекту, відсоток завершеності, порушення термінів, проблеми та складності у роботі, комплекс заходів вжитих їх усунення, звітність за всіма проектами, повна історія взаємодії з клієнтами.
- Встановити обмежений доступ до інформації для окремих користувачів, якщо ці дані не мають безпосереднього відношення до їхньої роботи.

***Переваги проектованої системи***

Компанія, яка використовує у роботі Систему керування проектами, значно економить свій час і налаштовується на максимально ефективний результат роботи за допомогою оптимізації взаємодії працівників. Керівник отримує у своє розпорядження інструмент, який дозволяє керувати процесом та здійснювати контроль за виконанням завдань на кожному етапі проходження замовлення, включаючи кожного працівника, який бере участь у робочому процесі, миттєво реагувати та вносити коригування у хід виконання роботи.

Все це можна робити незалежно від місцезнаходження співробітників. Система керування завданнями дозволяє здійснювати спільну роботу над проектом, навіть якщо люди працюють у різних підрозділах, перебувають у бізнес-поїздці чи працюють віддаленим методом. Це забезпечує додаткову зручність, оскільки не завжди є можливість або необхідність швидко зібрати весь персонал разом.

Програмну систему керування завданнями зможуть використовувати розробники програмного забезпечення, керівники IT-відділів та будь-які інші

профільні фахівці, в діяльності яких передбачається спільна робота групи співробітників над проектами.

## Висновки до розділу 1

Для забезпечення будь-якого проекту системою керування задачами потрібно зробити базову модель з відкритим для вдосконалень кодом, з чого випиває, що це практично апробована модель.

Для реалізації ідеї вдосконалення правильним буде рішення керування такими принципами:

- В пріоритеті повинно знаходитись забезпечення комплексної управлінської візії.
- Базисом повинен бути аналіз ланцюжка каскаду наслідків реалізації змін.
- Інструменти розробки повинні бути універсальними.
- Повинні бути альтернативні погляди на імплементацію фреймворку.

Серед всіх можливих питань з приводу універсального фреймворку найважливішими можна вважати:

- Розгляд впровадження та створення фреймворку в різних доменних експертизах.
- Ідеї та досвід впровадження проектного фреймворку.
- Визначення можливих шляхів розвитку отриманого фреймворку.

Саме такі теми можуть допомогти у побудові та обиранні методів застосування за принципом єдності сприйняття  запропонованої практичної моделі.

# РОЗДІЛ 2
# ЗАСОБИ ТА ТЕХНОЛОГІЇ КЕРУВАННЯ ІТ ПРОЕКТАМИ

Підходів до керування проектами дуже багато, тому перш ніж зробити вибір, потрібно мати уявлення про найбільш використовувані. При цьому потрібно розуміти переваги та недоліки кожного з них.

## 2.1. Введення в архітектуру програм

Важливо розуміти, кожна методологія керування проектами пропонує різні стратегії, сприяють успішної реалізації проекту. Ретельно підібрана методологія враховує особливості проекту та дає правильні принципи керування, командної роботи, інструкції з контролю, перевірки та оцінки результату та багато інших переваг [9].

### 2.1.1. Типи методологій керування проектами

Як і будь-яка бізнес-модель чи підхід, методології керування проектами мають низку переваг та недоліків. Деякі методології спрямовані на швидкість реалізації проекту. Інші більше орієнтуються на охоплення складових проекту чи керування співробітництвом.

Більшість керівників проектів вважають такі методології, як Kanban, Agile і Waterfall, основою розробки проектів, інші розглядають лише як відправну точку.

Розглянемо декілька підходів та методологій:

- Водоспадна модель
- Agile

- SCRUM
- Kanban

Перш ніж розпочати огляд, хочемо нагадати — вибір методології залежить від особливостей проекту та його команди. І, природно, будь-яка методологія має бути адаптована під конкретний проект. Універсальних методик керування проектами немає.

#### 2.1.1.1. Водоспадна модель

Водоспадна модель, можливо, найстаріша з існуючих. Вінстон У. Ройс представив її ще 1970 року. Він вигадав цю методологію як у відповідь швидкий розвиток галузі розробки програмного забезпечення. Модель є однією з найбільш традиційних і, можливо, найпоширеніших. Вважається, що водоспадна модель – найпростіша методологія для розуміння та оптимальна відправна точка для вивчення методологій керування проектами.

Водоспадна модель була основою керування IT-проектами протягом багатьох десятиліть, але й багато інших галузей, таких як виробництво чи будівництво, також її використовують.

В основі водоспадної моделі лежить послідовне планування етапів робіт з подальшим розбиттям етапів на завдання. Початком проекту, відповідно до даної методології, є збір вимог до проекта, формування підходів, рішень та етапів реалізації.

***Стадії***

Етапи робіт у водоспадній моделі послідовні. Початок етапу 2 можливий тільки після завершення етапу 1. Для проекта розробки програмного забезпечення етапи виглядатимуть так (Рис. 2.1.):

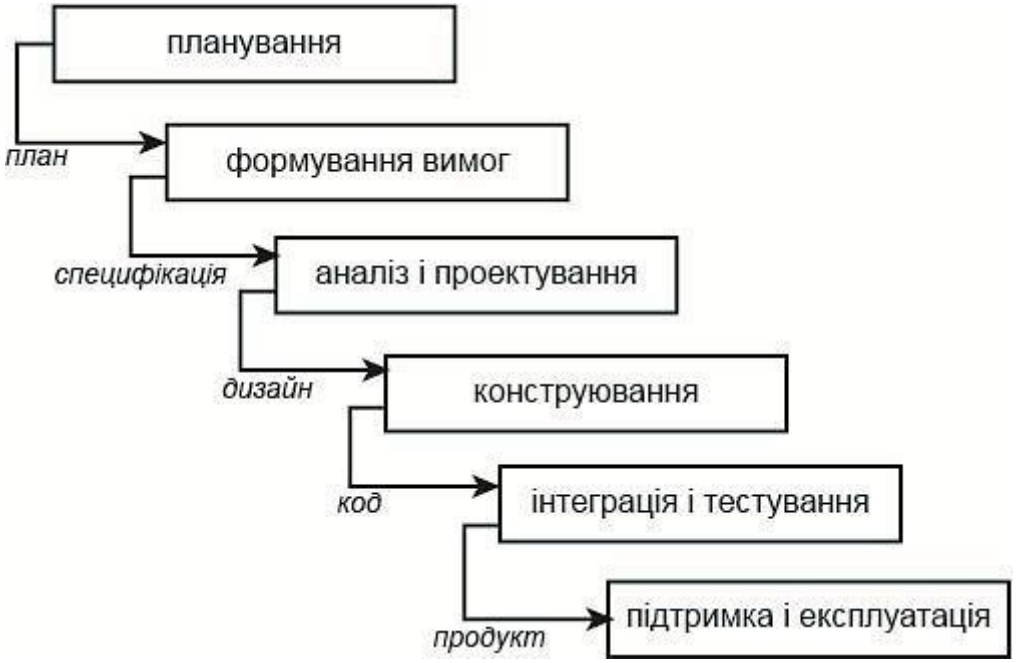



Рис. 2.1. Етапи розробки програмного забезпечення

Простіше кажучи, водоспад вимагає скласти список цілей та довести його до завершення. Тому цей метод добре підходить для галузей, де продукти вимагають точних та докладних інструкцій.

***Сильні і слабкі сторони***

Крім простоти, головна перевага водоспадної моделі полягає в тому, що практично будь-який план з минулих проектів може бути оптимізований і повторно використаний з невеликими коригуваннями. Крім того, його лінійна природа гарантує, що ви відповідаєте всім вимогам із самого початку – етап може бути завершений лише тоді, коли виконано всі вимоги. І так щодо кожного етапу проекту.

Ця методологія керування проектами значною мірою спирається на документацію та записи у процесі розробки. Таким чином, цей метод також спрямований на полегшення проблем, пов'язаних із унікальними знаннями працівників. Наприклад, у разі відходу старого співробітника, новий може продовжити роботу миттєво та без будь-яких труднощів завдяки наданій документації. Ще однією перевагою методу є посилення нагляду та контролю на кожному етапі.

Але як тільки етапи проекту встановлені, методологія вимагає, щоб вони залишалися незмінними. Таким чином, метод виявляється «упертим» та має серйозні обмеження. Якщо діапазон та масштаб проекту змінюються, план не зможе адаптуватися до такої зміни.

Деякі критики стверджують, що зниження рівня гнучкості та креативності робить цю модель застарілою порівняно з сучасними стандартами. Ще один недолік полягає в тому, що метод не спирається на зворотний зв'язок від зацікавлених сторін, який можна використовувати для покращення результатів проекту.

***Застосування методології***

Незважаючи на свої слабкі сторони, основною перевагою методології є її підхід до планування проекту. Цей підхід легко адаптується до будь-якого процесу розробки, що включає об'ємні і складні завдання. Наприклад: промислове виробництво, будівництво, розробка нових продуктів тощо. Також водоспадна модель полегшує майбутню реалізацію схожих проектів: проект будівництва типової будівлі може бути легко адаптований до нового об'єкту будівництва.

Методологія відмінно підходить для складних проектів зі строгими термінами реалізації. А також для проектів, які були раніше реалізовані з низьким рівнем ризиків і помилок.

### 2.1.1.2. Agile

Методологія Agile – засіб обходу слабких місць водоспадної моделі. Хоча деякі концепції Agile використовуються вже давно, цю методологію керування проектами було офіційно представлено в 2001 році в «Agile Manifesto». Публікація, написана провідними експертами з розробки програмного забезпечення, закликала до альтернативи, яка б протистояла великоваговим, орієнтованим на документацію, підходам до керування проектами.

Методологія фокусується на проектах, які потребують швидкості та адаптивності. Гнучке керування проектами ґрунтується на короткострокових «спринтах», а також високого ступеня інтерактивності та співробітництва.

***Цінності agile***

Методологія Agile заснована на чотирьох основних цінностях (Рис. 2.2.):


Рис. 2.2. Цінності Agile

Причина, через що Agile відкрила нові горизонти в керування проектами, у її революційній ідеології. Проекти, що гнучко керувалися, зуміли створити віху, з точки зору адаптивності, співпраці з клієнтами та надання цінності.

Відверто кажучи, Agile це не методологія. Це ідеологія та підхід, яка знаходить своє відображення у таких методах як: Scrum, Adaptive Project Framework (APF) та Extreme Programming.

Основний зміст полягає в наступному: проект може розвиватися і змінюватися з часом, відповідно продукт, рішення або результат проекту також можуть змінюватися разом з ним.

***Ключові принципи***

У маніфесті Agile перераховано дванадцять основних принципів:

1. Найвищим пріоритетом для нас є задоволення потреб замовника завдяки регулярній та ранній поставці цінного програмного забезпечення.
2. Зміна вимог вітається навіть на пізніх стадіях розробки. Agile процеси дозволяють використовувати зміни для забезпечення замовнику конкурентної переваги.
3. Працюючий продукт слід випускати якнайчастіше, з періодичністю від кількох тижнів до кількох місяців.
4. Протягом усього проекту розробники та представники бізнесу мають щоденно працювати разом.
5. Над проектом мають працювати вмотивовані професіонали. Щоб роботу було зроблено, створіть умови, забезпечте підтримку та повністю довіртеся їм.
6. Безпосереднє спілкування є найбільш практичним та ефективним способом обміну інформацією.
7. Діючий продукт - основний показник прогресу.
8. Процес розробки повинен бути сталим.
9. Якість та технічна досконалість у пріоритеті, завдяки чому проект стає гнучким.
10. Команда не потребує мікро менеджменту та здебільшого само організована.
11. Всі учасники процесу аналізують варіанти покращення ефективності проекту та змінюють порядок своєї роботи відповідно цьому.

***Сильні і слабкі сторони***

У порівнянні з водопадною моделлю методологія Agile забезпечує більш високу гнучкість. Однак реалізація сильно змінюватиметься залежно від потреб проекту. Основний плюс методології керування проектами полягає в тому, що вона

дозволяє здійснювати співпрацю, зміну та переоцінку у процесі розробки. Тому вона дуже популярна у світі інформаційних технологій.

Методологія підтримує безперервне вдосконалення, залишаючи простір експериментів. Таким чином, вона неймовірно добре працює для проектів, які потребують високого рівня інновацій та креативності.

Інші його переваги включають швидкий цикл, постійний зворотний зв'язок та підвищену продуктивність.

І навпаки, деякі з недоліків – відсутність фіксованих планів та оцінок, труднощі фінансового керування та збої у плануванні. І хоча деякі віддають перевагу інтерактивності, в інших може не вистачити часу на постійну співпрацю. Більше того, ця методологія керування проектами не спирається на передбачуваність, що означає, що витрати, зусилля та час не так просто оцінити, порівняно з іншими методологіями.

***Застосування методології***

Гнучкість Agile гарантує, що ви зможете впровадити цю методологію у різноманітні проекти, галузі та продукти. Він неймовірно добре підходить для проектів із значним рівнем невизначеності та взаємодії.

Підхід широко поширений у маркетингу, розробці продуктів та виробництві, будівництві, фінансах, плануванні заходів, а також в автомобільній промисловості. В цілому Agile може сприяти швидкому виробництву при розширеному співробітництві, і він дуже уважно ставиться до зростання попиту та еволюції ринку.

### 2.1.1.3. Scrum

Багато хто вважає Scrum найпростішим і найбільш широко використовуваним методом, пов'язаним із Agile. Він заснований на спринтах, які тривають від двох тижнів до тридцяти днів. Цей підхід служить визначення пріоритетів завдань, наголошуючи на команді. Під час спринтів учасники команди проводять 15-хвилинні зустрічі, щоби переглянути свої результати. Ось як працюють спринти:

- Команда використовує дошку для відстеження списку завдань.

- Завдання розділені на кілька списків.
- Люди можуть отримувати завдання з дошки та приступати до роботи.
- Кожне завершене завдання перевіряється якість.
- Після власником товару завдання позначається як виконана.

Мета – уникнути перевтоми та виснаження. Методологія Scrum спрямована на стимулювання гнучкості, креативності та надання високоякісних результатів.

На відміну від керівника проекту, цей підхід використовується поняття «scrum master». Ця роль полягає в тому, щоб усунути будь-яку перешкоду і тим самим підвищити продуктивність для самоврядної команди. Scrum запозичує багато процесів і принципів Agile, але використовує особливий набір тактик і цінностей.

Ось п'ять основних цінностей Scrum, які фокусуються на людині та команді (Рис. 2.3.):


Рис. 2.3. Цінності Scrum

Відповідно до цих принципів, методологія Scrum спрямована на сприяння співпраці, успішній реалізації та стійкості складних продуктів. І так само, як інші підходи Agile, вона заохочує ітеративний розвиток.

***Сильні і слабкі сторони***

Орієнтований на спринт робочий процес Scrum полегшує комунікації та керування. Протягом певного періоду спринту команда може крок за кроком аналізувати свої цілі. Цей підхід фокусується на практичних завданнях, швидкій співпраці та ефективної координації. Це одна з найлегших методик із чітким визначенням обов'язків та ролей.

Підхід ґрунтується на постійному вдосконаленні та оптимізації. Однак гнучкість, що він забезпечує, також є серйозним обмеженням. Наприклад, самоврядна Scrum-команда може бути не в змозі виконати вимоги організацій та галузей, які вимагають фіксованого обсягу, встановлених бюджетів, суворих термінів та інших серйозних обмежень. Ось чому методологія краще працює у складі гібридних підходів. Багато організацій та агентств використовують методи самоврядування та оцінки, що надаються Scrum, у поєднанні з іншими підходами.

***Застосування методології***

Теоретично Scrum був би корисним для невеликих команд (десять чи менше), які вимагають високого рівня гнучкості.

Цей підхід найкраще підходить для галузей з високим рівнем невизначеності, що змінюється навколишнім середовищем та тих, які щоразу вимагають абсолютно нових продуктів чи рішень. Деякі приклади, де Scrum може бути корисним поза розробкою програмного забезпечення – це маркетинг, юриспруденція, реалізація стратегії та дизайн продукту. Однак галузі, які покладаються на повторення та передбачуваність, повинні використовувати інший метод.

### 2.1.1.4. Kanban

Ключова відмінність Kanban полягає у його сильному акценті на візуалізацію у процесі розробки.

Цей термін відноситься до 1940-х років і означає «сигнальна карта». Спочатку карти Kanban використовувалися Toyota для реорганізації ресурсів та підвищення ефективності. В даний час Kanban використовує методи віртуальної візуалізації за допомогою систем керування та інших інструментів керування проектами.

***Візуальні сигнали***

Методологія керування проектами Kanban спрямовано візуалізацію робочого процесу виявлення вузьких місць і підвищення продуктивності. Прогрес відображається візуально за допомогою різних сигналів:

- Дошка Kanban. Дошка (цифрова чи фізична) відображає етапи розробки чи керування.
- Мапи. Карти Kanban видаються кожному за завдання чи предмета. Вони можуть відслідковувати прогрес і співробітництво та надаючи дані (керування крайніми термінами, статус тощо).
- Доріжки. Є особливу форму відстеження та класифікації завдань. Зазвичай вони мають горизонтальне планування та потоки.

Крім візуалізації, Kanban також слідує іншим принципам, таким як керування потоками, цикли зворотного зв'язку, спільне поліпшення і т. д. Крім цих принципів, Kanban не має фіксованого набору правил і етапів, або запропонованих ролей.

***Сильні і слабкі сторони***

Методологія керування проектами сприяє візуальній залученості. Це може усунути проблеми, пов'язані з керуванням робочим навантаженням та розміщенням пріоритетів. Метод призводить до підвищення продуктивності, оскільки дозволяє керівникам ефективніше контролювати робоче навантаження.

Понад те, методологія та її інструменти прості розуміння. Це також знижує рівень незавершеного виробництва та призводить до оптимізованого та більш швидкого робочого процесу. Команди, що використовують Kanban, також можуть адаптуватися до змін набагато швидше. Що стосується мінусів, то у команди можуть виникнути проблеми з навігацією та обслуговуванням дошки Kanban та інших інструментів. При поганій обробці сигнали візуалізації можуть призвести до надмірного ускладнення та безладного робочого процесу.

***Застосування методології***

Команди та проекти, які забезпечують стабільний результат або працюють у стислий термін, можуть використовувати Kanban для підвищення своєї ефективності. Також методологія зручна для реалізації типових проектів з передбачуваними етапами робіт. Крім цього, метод може принести користь організації, яка значною мірою залежить від виконання обсягу різношерстих завдань. Його можна використовувати для моніторингу та вимірювання технічної роботи, термінів виконання, термінових завдань, а також роботи, що повторюється (наприклад, нарад, звітів тощо). До речі, дошки канбан дуже зручні для керування операційними процесами.

### 2.1.2. Вибір методології

Методологія, яка найкраще підходить для певного проекту, може запропонувати вам ефективне покрокове керівництво роботою. Однак, як уже згадувалося раніше, методологія може бути ефективною лише в тому випадку, якщо вона підходить для цього проекту. То як знайти найвідповідніший підхід?

***Зосередженість на результаті***

Найчастіше, щоб знайти ідеальну методологію, було б корисно розглянути кінцевий результат. Результати та вигоди, яких ви плануєте досягти, можуть наблизити до вибору правильного підходу.

***Оцінюйте команду***

Вкрай важливо перевірити процеси, які вже були впроваджені, та оцінити сильні та слабкі сторони команди. Визначення того, чи покладається команда на співпрацю як на свою творчу лінію життя чи жорстку структуру, також може допомогти знайти оптимальну модель.

***Розгляньте весь проект***

Після оцінки команди та цілей проекту варто проаналізувати обмеження проекту, його терміни, необхідні інструменти та всіх залучених осіб. Однак майте на увазі, що

навіть якщо певна методологія здається правильним рішенням, вона не завжди може працювати і може вимагати подальших коригувань.

Більше того, вкрай важливо завершити подальші дослідження та йти в ногу з ландшафтом керування проектами та його методологіями, що постійно змінюється.

***Заключні думки***

Методологія керування проектами є незамінними інструментами, які допоможуть вам у реалізації успішних проектів з високою цінністю. І хоча кожен підхід дає багато переваг, слід зазначити, що методологи не є універсальними.

Ефективність методу змінюватиметься залежно від галузі та конкретного проекту. У свою чергу, їх сильні сторони та переваги працюють найкраще, коли ви застосовуєте їх правильно, дотримуючись типу проекту та його вимог. Це включає тип команди, зміст проекту, цілі результатів, а також можливості керування проектом.

Ні для кого не є секретом, що керування проектами часто стикається з невизначеностями, які знаходяться за межами його розуміння. Це означає, що важливо вибрати ідеальну методологію, яка може доповнити та покращити проект.

## 2.2. Підходи до впровадження системи керування проектами

Коли організації починають впровадження системи керування проектами, вони не чекають на невдачі, незважаючи на ризики, пов'язані зі складністю проекту. Швидше, вони розраховують на успіх, у межах їхніх бюджетів, вимог до результатів, очікувань керівників та дедлайнів [10]. Проте, незважаючи на всі зусилля, спрямовані на керування проектами впровадження, частка невдач таких проектів залишається високою.

Впровадження системи керування проектами закінчується невдачею з безлічі причин, що включають: брак підтримки з боку вищого керівництва, нереалістичні очікування, погане визначення вимог, неправильний вибір пакетів, відмінності між програмними вимогами та бізнес-вимогами та недостатні ресурси. Крім цього можна виділити нереалістичні бюджети та графіки, погане керування проектом, відсутність

методології керування проектами, недооцінку впливу змін, відсутність навчання, і - остання, але не менш важлива – погана взаємодія.

З таким переліком факторів, здатних викликати провал проекту впровадження системи керування проектами, підвищення шансів на успіх здається неможливим, але його можна досягти.

Для початку необхідно скласти план стратегічного забезпечення проекту в критичних точках застосування. Такий план визначає очікування всіх людей, які беруть участь у проекті впровадження – від керівництва компанії та керівництва IT-відділу до партнерів вендора та кінцевих користувачів.

Методологія проектного керування дає впевненість, що проект завершиться вчасно, у рамках бюджету та результати будуть прийняті клієнтом. Якщо включити цю методологію до великомасштабного проекту впровадження системи керування проектами, то можна:

- Контролювати та скоротити вартість проекту;
- Переконатись, що віхи виконуються;
- Мінімізувати незаплановані фактори;
- Забезпечити об'єктивний аналіз;
- Забезпечити спокій та довіру серед керівників та членів команди проекту.

***Використання системи керування проектами. Проектний підхід***

Можна виділити такі найкращі практики проектного керування для впровадження системи керування проектами:

1. Перш за все треба визначити реальні проблеми.
2. На керівному рівні необхідно організувати діалог з керівництвом, який дозволить визначити та проаналізувати організаційні та бізнес питання без емоцій. Продовжувати цей діалог потрібно протягом усього процесу впровадження. Обов’язково повинні бути усунені організаційні бар'єри як з боку організації, так і з боку вендора. Усі сторони мають бути приведені у відповідність із загальною метою – успішною реалізацією проекту.
3. Потрібно ставити реалістичні часові рамки.

4. Не варто покладатись на розклад. Багато організацій встановлюють оптимістичні дати запуску, незважаючи на реальну ситуацію та обмеження проекту.
5. Наприклад, фаза розробки проекту зростає, а терміни реалізації проекту немає. Потрібно відслідковувати прогрес реалізації проекту в ході всього процесу впровадження та запустити обговорення ключових дат проекту на початку життєвого циклу проекту, щоб уникнути факторів, які можуть негативно вплинути на строки.
6. Привести у відповідність потоки робіт.
7. Важливо визначити, привести у відповідність та постійно контролювати потоки робіт для гарантії безперервного прогресу у всій організації. Зрозуміти залежність між роботами та тимчасовими рамками проекту під час розробки плану проекту для забезпечення належного розподілу ресурсів. Продовжувати відстежувати взаємозалежність протягом усього проекту.
8. Не покладатись лише на індикатори.
9. Всупереч поширеній думці зелений насправді може бути червоним. Реалістичний моніторинг та аналіз прогресу впровадження може показати, що навіть незважаючи на те, що всі показники проекту зелені, попереджувальні знаки вказують на проблемні елементи. Якщо показники торкатимуться лише минулих етапів проекту, але не показуватимуть готовність до майбутніх завдань та діяльності, вони, безумовно, не заслуговують на довіру в прогнозах на майбутнє.
10. Керувати очікуваннями.
11. Найважливішим для контролю проектів є необхідність знаходити компроміс між надмірно оптимістичними термінами запуску системи та зовнішніми факторами, реалістичними очікуваннями та обмеженнями, такими як доступні ресурси. Потрібно ставити реалістичні терміни та нагадувати про них проектній команді, щоб вони не втратили уявлення про ліс, коло навколо дерева.
12. Досягнути об'єктивності.

Оцінки, які може дати зовнішній експерт, допоможуть досягти як більшої цінності проекту впровадження системи керування проектами, так і гарантій, що проект не зазнає дорогої невдачі. Експертиза дозволяє отримати дорожню карту реалізації проекту та об'єктивну оцінку, необхідну для подолання організаційних перешкод. Також вона допоможе залишатися спокійними всіх учасників процесу.

Оцінки повинні проводитися або виконавчим керівником проекту, або експертом з впровадження системи керування проектами, який мав досвід успішного керування достатньою кількістю проектів, щоб зрозуміти, як розпізнати індикатори, що важко розрізняти, втручатися, щоб керувати ситуацією, і відповідно регулювати очікування.

Використання методології гарантованої реалізації проектів дозволяє вийти за рамки традиційного керування проектами і дає відповіді, необхідні для підвищення ймовірності успіху проекту. Ця методологія дозволить визначити та вирішити стратегічні, тактичні та нематеріальні питання та керувати людськими факторами, перш ніж проблеми стануть нерозв'язними. А найкраще те, що проектне керування дає всім учасникам впевненість, що проект знаходиться на правильному шляху.

## 2.3. Аналіз програмних систем Task менеджерів для керування IT проектами

Таск-менеджер - програмне рішення, призначене для керування проектами. Його функціонал дозволяє керівнику роздавати персоналізовані та групові завдання, розділяти їх на етапи, контролювати виконання всіх заданих процесів. Такі послуги досить давно користуються популярністю, адже дозволяють здійснювати загальне керування та контроль над проектами з меншими витратами часу. Не потрібно ходити між столами та кабінетами – все можна побачити на своєму екрані будь-якої миті [11].

Впровадження та використання таск менеджерів було корисним та актуальним раніше. Але зараз в умовах масового переходу на формат віддаленої роботи їхнє застосування стає життєво необхідним. Адже набагато зручніше ставити та

коригувати завдання, контролювати їх виконання, керувати командою в єдиному інтерфейсі, ніж окремо спілкуватися з кожним учасником проекту.

Загалом, перехід керування проектами в таск-менеджері – логічний етап побудови ефективної командної віддаленої роботи. Але важливо обрати правильний сервіс, який відповідатиме всім вимогам, а також працюватиме на всіх популярних дистрибутивах Windows, Mac і Linux.

Таск-менеджер для роботи над проектами з участю віддалених співробітників – сервіс необхідний, навіть незамінний. Але для ефективної побудови процесу лише його використанням не обійтися. Він дозволяє ставити завдання, обмінюватися інформацією, спілкуватися та отримувати звіти про виконану роботу. Але реально контролювати співробітників жодна з вище перерахованих систем та їх менш успішних аналогів не вміє. Просто нема відповідного функціоналу.

Повноцінно керувати колективом можна тільки маючи змогу контролювати сам робочий процес – знати, чим, скільки часу та з якою інтенсивністю займається кожен працівник. Потрібно не лише ставити завдання, а й перевіряти, як вони виконуються, бачити, хто працює, а хто відбуває номер. Для того, щоб врахувати всі критерії, потрібно порівняти вже існуючі таск менеджери [11].

### 2.3.1. Trello

Trello – найбільш відомий серед конкурентів. Сервіс орієнтований на невеликі стартапи та колективи з обмеженою кількістю учасників у проекті. Але в ньому є необхідний базовий функціонал для організації керування та внутрішньої взаємодії. Приклад Trello дошки (Рис. 2.4.):

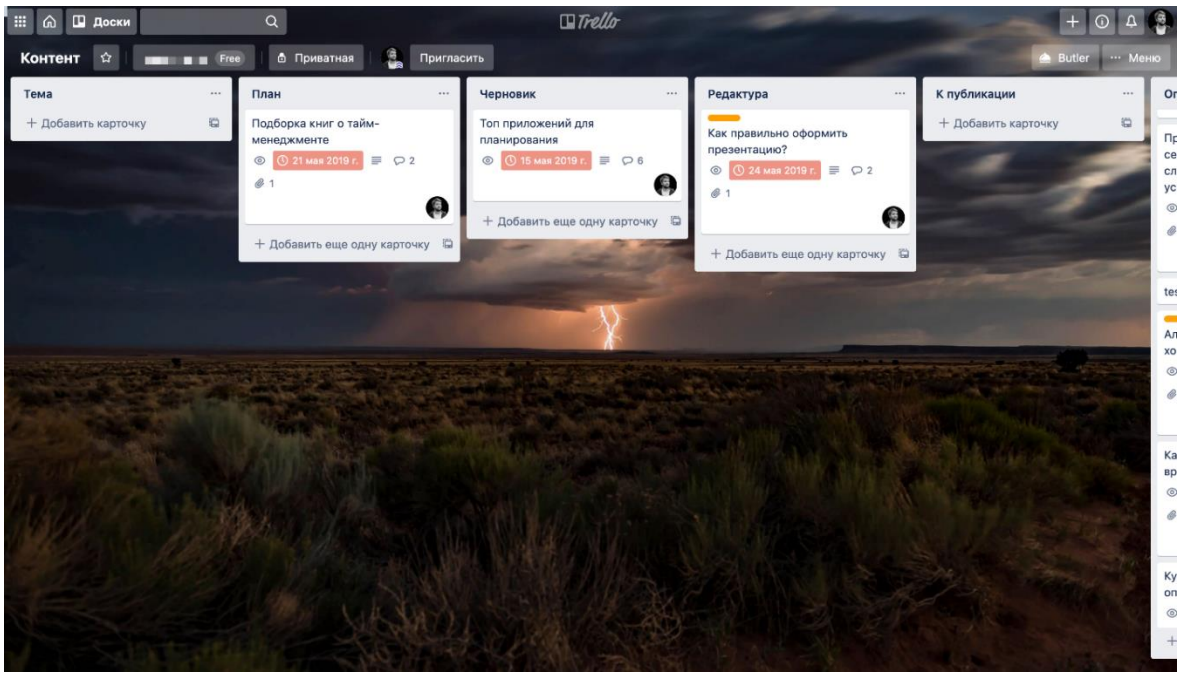



Рис.2.4. Приклад Trello дошки

Система керування – Канбан. Завдання ставляться на дошці з картками – персоніфікованими та груповими. Підтримується інтеграція з Google Drive, Dropbox, OneDrive та іншими сторонніми сервісами.

Інструмент вкрай популярний завдяки зрозумілій ідеї з канбан-дошками, яка лежить в основі всього. Начебто картки-завдання мають свої дедлайни, а начебто їх можна і між колонками переміщати, таким чином відстежуючи прогрес роботи.

Систему легко впровадити, інтерфейси візуально прості та барвисті, зрозумілі навіть тим, кому за 60 років. Проте, вона не підходить для масштабних проектів. Примітно, що в мобільних додатках Trello від своєї core-концепції не відходить - жодного списку завдань там побачити не можливо, і завдання доведеться перетягувати між стовпцями.

Переваги:

- Логічне побудова системи постановки завдань;
- Зручна карткова система;
- Основні операції із завданнями вимагають не більше 2 кліків;
- Інтеграція із сторонніми сервісами;

- Функціональна система фільтрації за кольорами.

Недоліки:

- Малі можливості кастомізації;
- Нема аналітики;
- Немає врахування часу за завданнями у базовому функціоналі;
- Інтерфейс швидко захаращується, довго шукати потрібну дошку;
- На телефоні не зручно переміщати картки між колонками;
- Завдання з різних проектів не видно на одному екрані;
- Підходить лише для найпростіших лінійних завдань.

Ціна:

- Базовий функціонал безкоштовний;
- Можливість керування правами доступу та вивантаженням даних - 25 $ на місяць.

### 2.3.2. Jira

Оригінальний таск-менеджер, в якому дошки Канбан використовуються одночасно з технологією Scrum. Це дозволяє охопити максимальну кількість учасників проекту, що виконують різні завдання, що володіють різною значимістю в його рамках. Є можливість зворотної організації процесу: співробітник сам планує проект, а уповноважений контролер перевіряє та спрямовує його виконання. Діє система оцінок.

Jira доступна в десктопній версії, а також у вигляді програм під iOS та Android. Підтримує інтеграцію із популярними сторонніми сервісами.

Jira - сервіс, в якому проекти складаються із завдань-тикетів. Все зроблено під реєстрацію та виконання запитів на обслуговування, тому його часто використовують IT-команди. Скрін сервісу: Jira (Рис. 2.5.):

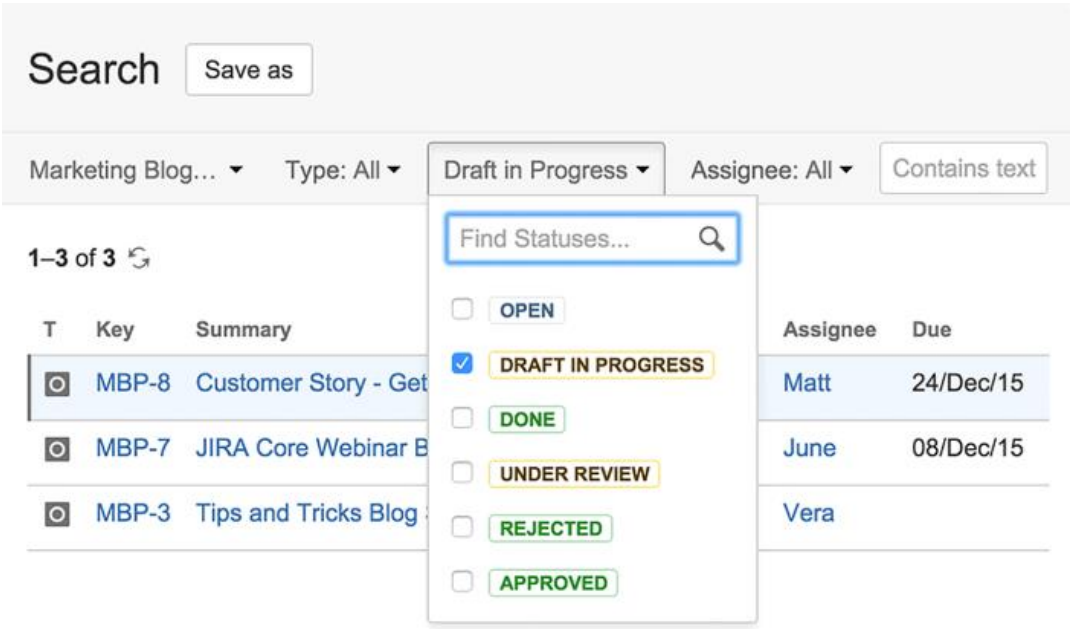



Рис.2.5. Скрін сервісу Jira

Перевага і водночас недолік Jira – високий рівень кастомізації. На основі API та плагінів можна налаштувати Jira, щоб відправляти нагадування в корпоративний чат, приховувати неактуальні коментарі до завдання та багато іншого. Але для цього знадобиться допомога фахівця.

Система заточена під відділ розробки, користуватися нею може будь-яка кількість співробітників, у тому числі великі команди понад 100 осіб. Найчастіше використовується як баг-трекер. Впровадження вимагає значного часу. Якщо команда не технічна, доведеться довго й болісно навчати співробітників.

Переваги:

- Функціональний мікс із канбан-дощок та scrum;
- Величезний набір функціоналу та можливостей для кастомізації;
- Система оцінювання виконавців;
- Потужний api;
- Більше 3000 додатків jira software;
- Зручне планування спринтів;
- Інтеграція із хмарними сервісами;
- Багаторівнева побудова проектів.

Недоліки:

- Система складна у освоєнні;
- Інтерфейси часто змінюються;
- Складно налаштовувати власні робочі процеси;
- Мало можливостей для автоматизації;
- Повільно підвантажуються сторінки.

Ціна:

- Якщо учасників трохи більше 10, потрібно оплачувати 10$ на місяць;
- Якщо учасників від 11 до 100, оплата – 7$ на місяць за кожного з них.

### 2.3.3. Redmine

Система із розширеними можливостями керування. Для кожного завдання можна призначати окремий список учасників-виконавців. Кожному виконавцю можна відкривати доступ до функціонала індивідуально. Є можливість створювати одразу кілька проектів та підпроекти у їх рамках. Реалізовано систему відстеження помилок. Скрін сервісу Redmine (Рис. 2.2.):

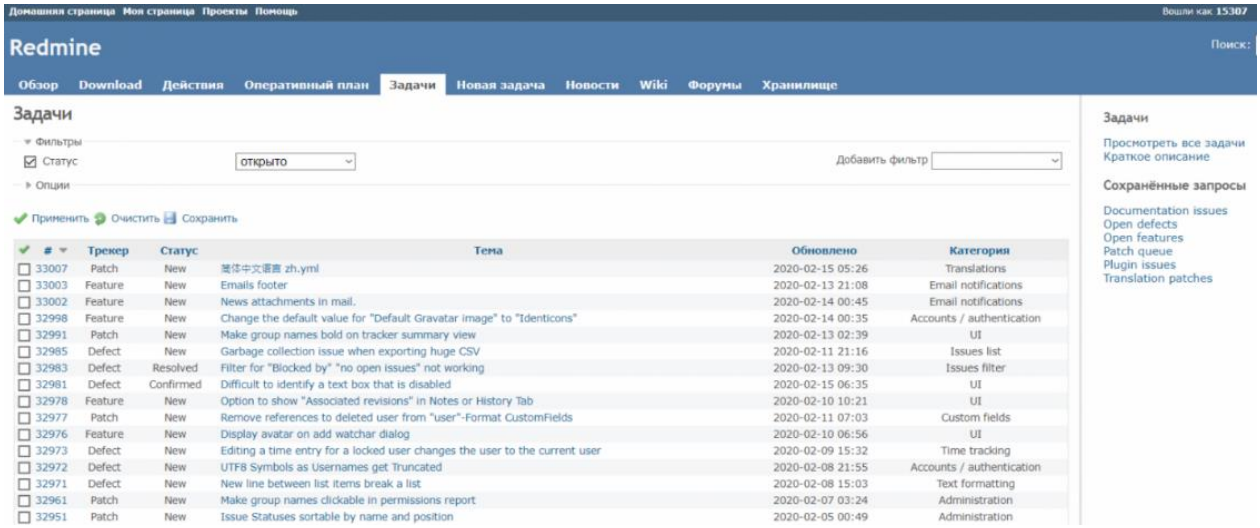



Рис.2.6. Скрін сервісу Redmine

Redmine дозволяє кожному користувачеві задавати конкретну роль у межах кожного проекту. Це зручно, оскільки дозволяє структурувати ієрархію щодо доступу, обов'язків та інших критеріїв. Усім є лише те, що їм потрібно для роботи.

Унікальна програма з відкритим вихідним кодом, на 100% безкоштовна, створена ентузіастами ще в середині 2000-х. Система керування проектами не для всіх, орієнтована на технічні команди. Заточена під відстеження багів та керування проблемами у відділі розробки.

Переваги:

- Сервіс повністю безкоштовний;
- У систему інтегровано все необхідне керування усіма етапами розробки;
- Мультимовність;
- Зручні пошук;
- Фільтрація та сортування карток;
- Гарні діаграми;
- Календар та форум;
- Робота із кількома проектами;
- Має у своїй конфігурації інструменти для обміну та обговорення коду;
- Створення підпроектів;
- Система обліку часу;
- Система повідомлень.

Недоліки:

- Застарілий інтерфейс;
- Деякі функції заховані у неочевидних місцях;
- Немає інструмента для відстеження часу та активності користувача за завданням;
- Важко встановити на власному сервері.

Ціна:

- Повний доступ розробниками надається безкоштовно.

## 2.4. Найбільш поширені проблеми з системами керування IT проектами

Спочатку кілька слів про те, навіщо потрібне керування проектами. Знамените опитування CHAOS Chronicles, проведене The Standish Group показало, що у світі лише 32% IT-проектів завершуються успішно, а 23% IT-проектів у світі повністю провалюються. У 2009 році картина була вже дещо кращою, ніж у 1994, але покращень явно не достатньо. Статистика успішності IT-проектів CHAOS (Рис. 2.7.):

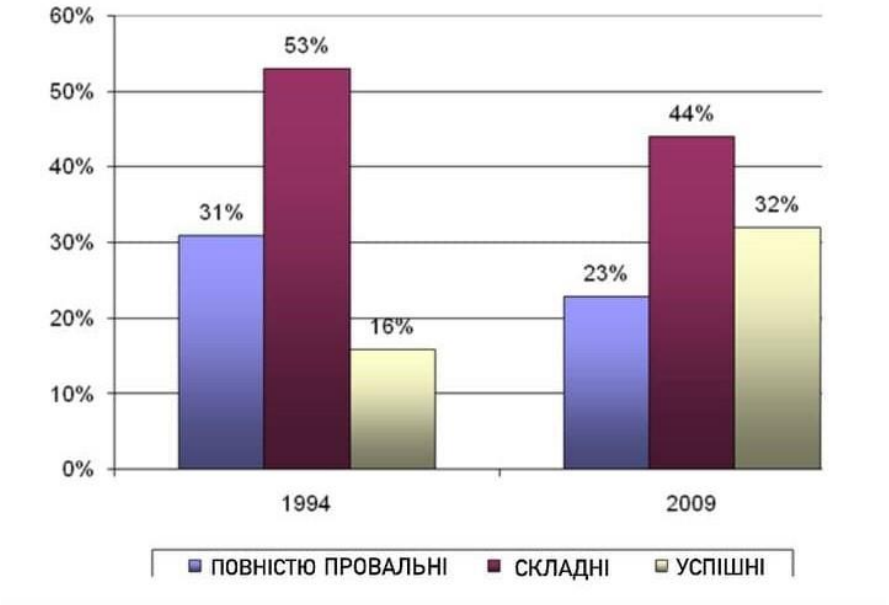



Рис 2.7. Статистика успішності IT-проектів CHAOS Chronicles (Standish Group)

Результати дослідження PMI 2004 року, в якому аналізувалися 23 тисячі проектів із розробки додатків, аналогічні: лише 26% IT-проектів виконується вчасно та в рамках бюджету, 46% спізнюються або виходять за рамки бюджету, а 28% провалюються. Загальної статистики щодо російських проектів, на жаль, немає. Існує єдине дослідження Hewlett-Packard та Economist Intelligence Unit згідно з яким, лише 5% російських IT-проектів завершуються вчасно.

До речі, ті, хто стверджує, що провали це специфіка саме IT-проектів, не мають рації – та ж проблема існує, наприклад, і для дуже великих інфраструктурних проектів. Згідно з проведеним дослідженням 258 інфраструктурних проектів із

загальним бюджетом понад 90 млрд. дол. 9 із 10 проектів стикаються з перевищенням бюджету.

Крім наведених вище проблем є три серйозні причини, які підштовхують компанії до впровадження проектного керування та керування програмою проектів:

- Зміни в організації стають все більш складними та комплексними.
- Досягнення цілей проектів вимагають тісної взаємодії функцій та залучення безлічі зовнішніх сторін.
- Існуюча організація, процеси та системи не підтримують такий вид діяльності, як проекти.

Керування проектом із використанням напрацьованих стандартних інструментів дозволяє відчутно підвищити ймовірність успішного завершення проекту, щоправда в обмін на додаткові витрати (зарплата проектного менеджера, вартість створення планів, документації, звітності тощо). Додатковим бонусом йде скорочення термінів та витрат проекту за рахунок уникнення непродуктивної, не потрібної роботи.

З цього можна визначити найбільш слабкі сторони систем керування проектами:

- Що більше проектів ведеться, то складніше стає орієнтуватися;
- Інтерфейс швидко захаращується;
- Немає врахування часу за завданнями у базовому функціоналі;
- Мало можливостей для класифікації карток;
- Вести роботу у великих командах важко;
- Безладдя у завданнях;
- Для великих проектів не достатньо стандартних звітів;
- Немає ієрархії співробітників;
- Надто демократична система прав;
- Важко розставляти пріоритети для проектів;
- Бракує автоматизації процесів;
- Немає можливості кастомізувати звіти під себе;

- Збої під час виконання завдань;
- Дані не синхронізуються різних пристроях та друге.

## Висновки до розділу 2

Кожна організація в умовах жорсткої конкуренції прагне оптимізації робочих процесів, значного скорочення витрат у процесі виконання замовлення, максимально ефективно застосовувати трудові ресурси, ліквідувати втрату та спотворення актуальної інформації, яка застосовується в роботі. Однак не кожна компанія здатна втілити в життя всі перелічені зміни якнайкраще. Ідеальним помічником у цій ситуації може стати Система керування завданнями (Task management software).

Система керування завданнями (Task management software) допомагає модернізувати роботу над поставленими завданнями, зробивши її максимально ефективною, незалежно від обсягу штату компанії та місця перебування працівників. Кожному з учасників цього проекту надається перелік доручень та завдань, які ставляться безпосередньо перед ним, а також ступінь виконання та пріоритетність завдання в окремий проміжок часу порівняно з іншими завданнями.

Task management software є зручною системою з організації групової роботи над завданнями та керівництва проектами.

Весь перелік завдань та вчинені для вирішення дії розміщено в одному місці. До системи можна легко отримати доступ через глобальну мережу практично будь-коли.

Керівництво компанії завжди бачить дійсний стан справ. Розподіл відповідальності та постановка пріоритетів з програмою Task Management Software відбувається максимально швидко.

Керівник ставить завдання, передаючи свої повноваження та призначаючи відповідальних осіб, розставляє перед ними пріоритетні завдання, позначає конкретні терміни роботи над проектом.

Усі учасники трудового процесу перебувають у єдиному інформаційному полі: завдання, доручення, документація, дії. База даних розташована в одному місці, адаптованому для максимально зручного пошуку.

Вже існуючі сисстеми керування проектами мають ряд проблем, над виправленням яких планується робота у третьому розділі дипломного проекту:

- Нема аналітики;
- Немає ієрархії співробітників;
- Незручно для великих командах від 20 осіб;
- Важко розставляти пріоритети для проектів;
- Відсутня навіть умовно безкоштовна версія;
- Система складна у освоєнні;
- Інтерфейси часто змінюються;
- Складно налаштовувати власні робочі процеси;
- Мало можливостей для автоматизації;
- Повільно відвантажуються сторінки;
- Застарілий інтерфейс;
- Деякі функції заховані у неочевидних місцях;
- Немає інструмента для відстеження часу та активності користувача за завданням;
- Дрібні літери.

# РОЗДІЛ 3
# ТЕХНОЛОГІЇ РОЗРОБКИ СУЧАСНОЇ TASK-MANAGER СИСТЕМИ

Впровадження та використання таск менеджерів було корисним та актуальним раніше. Але в умовах масового переходу на формат віддаленої роботи їх застосування стає життєво необхідним. Адже набагато зручніше ставити та коригувати завдання, контролювати їх виконання, керувати командою в єдиному інтерфейсі, ніж окремо спілкуватися з кожним учасником проекту.

Загалом, перехід керування проектами в таск-менеджері – логічний етап побудови ефективної командної віддаленої роботи. Але важливо вибрати правильний сервіс, який відповідатиме всім вимогам, а також працюватиме на всіх популярних дистрибутивах Windows, Mac і Linux. Їх багато, зокрема популярних. Але неможливо знайти якийсь ідеальний для всіх проектів, тому можна спробувати розробити свій особистий, щоб зрозуміти складнощі та покрити саме свої питання.

## 3.1. Мова програмування Python. Обґрунтування вибору мови

Оскільки процес розробки веб-додатків радикально розвивався за останні кілька років, це те, що робить Python провідним конкурентом або вибором серед розробників програмного забезпечення. З його допомогою можна будувати практично все.

Як мова загального призначення, Python — це єдина мова, яка може бути слушною для розробки майже чого завгодно.

Python — це майстер на всі руки серед мов програмування. І оволодіння ним потенційно може дозволити розробнику програмного забезпечення бути експертом у всіх типах програмування [12].

***Переваги використання Python***

- Python багатоцільовий. Це мова програмування загального призначення з широким набором додатків.
- Python є оптимальним вибором для операцій безпеки. Інформаційна безпека, безпека веб-сайтів та кібербезпека — це всі функції, які можна реалізувати за допомогою Python.
- Підвищує продуктивність розробника. Ідеологія Python полягає в дотриманні умов та уникнення повторюваних завдань, які займають багато часу. Повторне використання шаблонів і модулів дуже економить час, а простий у використанні синтаксис оптимізує весь процес розробки.
- Масова підтримка бібліотеки. Найбільшою перевагою Python є кількість підтримуваних бібліотек сторонніх розробників. Бібліотека Python — це фрагмент попередньо написаного коду, який можна включити у проект або колекцію модулів.
- Дуже переносимий код. Python розроблено, щоб бути дуже портативним. Він підтримується всіма операційними системами Windows, Linux, UNIX та macOS.
- Python використовується як клей для багатьох веб-додатків. Програмісти пишуть важливий програмний код на C/C++ або Java і використовують модулі Python для з’єднання між собою різних компонентів.
- Легко вивчати та використовувати. Читабельність коду та простий зручний дизайн є важливими аспектами мови програмування.
- Гнучкість, читабельність і потужний інтерпретатор роблять його однією з найпростіших об’єктно-орієнтованих мов у використанні. Автоматичне керування пам'яттю — ще одна перевага, яку пропонує Python. Він також підтримує численні парадигми програмування, такі як функціональні, об’єктно-орієнтовані, імперативні та декларативні, що дозволяє легко реалізувати.
- Частина LAMP Stack. LAMP – це дуже відома абревіатура для набору веб-технологій, які використовуються для створення веб-сайтів та веб-додатків [12].

***Недоліки використання Python***

- Продуктивність Python не досить висока. Це пов'язано з тим, що Python інтерпретується. Але це легко можна нівелювати за допомогою C реалізацій тієї чи іншої проблемної ділянки коду. В умовах сьогоднішніх потужностей це не дуже помітно.
- Динамічна типізація — через динамічну типізацію Python споживає більше ресурсів, ніж міг би, але це часто компенсується внутрішнім кешуванням.
- Global Interpriter Lock. На даний момент це є основною проблемою продуктивності Python, а також цим обумовлена погана реалізація багатопоточності.
- Високий рівень залежність від системних бібліотек. В результаті ускладнюється перенесення на інші системи. Так, проблема вирішується за допомогою Virtualenv, проте цей інструмент має свої недоліки: милиці, надмірність повних методів ізоляції, дублювання системних бібліотек [12].

Незважаючи на зазначені недоліки, переваги мови Python для розробки програмної системи task-менеджер очевидні.

***Обґрунтування фінального рішення щодо мови***

Python є інтерпретованою мовою програмування, яка не компілюється. Таким чином, до запуску він є звичайним текстовим файлом. Відповідно, програмувати можна майже на всіх платформах, а сама мова логічна і добре спроектована.

Кода в ньому менше, ніж при використанні інших мов програмування, тому розробка здійснюється швидше. Ось, наприклад, як виглядає код виведення тексту Hello, Otus! мовою програмування Java:

```
class Main {
    public static void main(String[] args) {
        System.out.println("Hello, Otus!");
    }
}
```

У Python ж для виконання того ж завдання досить всього одного рядка:

```
print("Hello, Otus!")
```

Можна зробити висновок, що для більшості завдань: для веб-розробки, для скриптів, прототипування, машинного навчання та роботи з великими даними Python — одна з найкращих мов.

## 3.2. Вибір фреймворку для програмної системи. Порівняння Django та Flask.

Згідно з опитуванням розробників Python JetBrains 2020 року, Django і Flask на сьогоднішній день є двома найпопулярнішими веб-фреймворками Python.

Обидві платформи використовуються для розробки веб-програм. Ключова відмінність полягає в тому, як вони досягають цієї мети. Про Django краще думати як про машину, а про Flask як про велосипед. Обидва можуть доставити з точки А до точки Б, але їх підходи абсолютно різні. У кожного є свої найкращі варіанти використання. Те ж саме і з Django та Flask [13].

### 3.2.1. Філософія Django та Flask

Django та Flask – це безкоштовні веб-фреймворки на основі Python з відкритим вихідним кодом, призначені для створення веб-додатків.

Порівняно з Flask, Django пропонує стабільність, а також підхід «з увімкненими батареями», коли ряд елементів (наприклад, інструменти, шаблони, функції та функціональність) надаються «з коробки». Що стосується стабільності, у Django зазвичай більш довгі та жорсткі цикли випуску. Отже, випуски Django містять менше блискучих нових функцій, але мають більшу зворотну сумісність.

Flask добре справляється із основними завданнями. За замовченням можна отримати маршрутизацію URL-адрес, обробку запитів і помилок, створення шаблонів, файли cookie, підтримку модульного тестування, налагоджувач і сервер розробки [13]. Незалежно від того, чи використовуються сторонні розширення або налаштовується код розробником, Flask для цього не заважає. Він набагато гнучкіший, ніж Django. Також набагато менше поверхні, відкритої для атаки, і менше коду для перевірки, якщо потрібно зламати капот і переглянути вихідний код.

### 3.2.2. Особливості Flask і Django. Розгляд функцій

Порівняємо Flask і Django на основі функцій, які постачаються з основним фреймворком.

***База даних***

Django включає просту, але потужну ORM (об'єктно-реляційне відображення), яка підтримує ряд готових реляційних баз даних — SQLite, PostgreSQL, MySQL та Oracle. ORM надає підтримку для створення та керування міграцією бази даних. Також досить легко створювати форми, уявлення та шаблони на основі моделей даних, що ідеально підходить для типового веб-додатка CRUD.

Flask не робить припущень про те, як зберігаються дані, але існує багато бібліотек і розширень, які допоможуть у цьому.

На завершення, якщо використовується реляційна база даних, Django значно спрощує початок роботу, оскільки має вбудований ORM та інструмент керування міграцією. Однак якщо використовується нереляційна база даних або планується використовуватись інший ORM, наприклад SQLAlchemy, Django буде «боротися» з розробником майже на кожному кроці.

Flask дарує свободу вибирати ORM (або ODM), який найкраще підходить для програми. Однак свобода має свою ціну: вища крива навчання і більше можливостей для помилок, оскільки розробник сам керує цими частинами.

***Django***

Коли запит відповідає шаблону URL-адреси, об’єкт запиту, який містить інформацію HTTP-запиту, передається у представлення, і це подання потім викликається. Щоразу, коли потрібно надати доступ до об’єкта запиту, треба явно передати його.

***Flask***

У своїй основі Flask використовує Werkzeug, який забезпечує маршрутизацію URL-адрес і обробку запитів/відповідей.

Об’єкт запиту є глобальним у Flask, тому можна отримати до нього доступ набагато простіше. URL-адреси зазвичай визначаються разом із представленням (через декоратор), але їх можна розділити в централізоване розташування, подібно до шаблону Django.

***Форми***

Форми, ще одна важлива частина більшості веб-додатків, постачаються разом із Django. Це включає обробку вводу та перевірку на стороні клієнта та сервера, а також вирішення різних проблем безпеки, таких як підробка міжсайтових запитів (CSRF), міжсайтові сценарії (XSS) та ін’єкція SQL. Їх можна створювати з моделей даних (через ModelForms) і добре інтегрувати з панеллю адміністратора.

Flask не підтримує форми за замовчуванням, але потужне розширення Flask-WTF інтегрує Flask з WTForms. WTForms-Alchemy можна використовувати для автоматичного створення форм на основі моделей SQLAlchemy, подолавши розрив між формами та ORM так само, як ModelForm від Django.

***Багаторазові компоненти***

Що стосується структури проекту, оскільки програми ускладняються, обидві фреймворки полегшують їх розділення, групуючи пов’язані файли разом, які мають схожу функціональність. Таким чином, можна, наприклад, згрупувати всі пов’язані з користувачами функції разом, які можуть включати маршрути, подання, форми, шаблони та статичні активи.

Django має концепцію програми, а Flask має креслення.

Програми Django складніші, ніж креслення Flask, але, як правило, з ними легше працювати та повторно використовувати після налаштування. Крім того, завдяки умовам urls.py, models.py і views.py (послідовна структура проекту) можна досить легко додавати нових розробників до проекту Django. Тим часом креслення простіше і легше запустити.

***Шаблони та статичні файли***

Механізми шаблонів дозволяє динамічно вводити інформацію на сторінку з серверної частини. Flask за замовчуванням використовує Jinja2, тоді як Django має власний механізм шаблонів. Вони досить схожі з точки зору синтаксису та наборів функцій. Також можна використовувати Jinja2 з Django.

### 3.2.3. Обов'язкові фактори для успішного вибору фреймворку

***Тестування***

Обидві фреймворки мають вбудовану підтримку для тестування.

Для модульного тестування обидва вони використовують фреймворк Python unittest. Кожен з них також підтримує тестовий клієнт, якому можна надсилати запити, а потім перевіряти та перевіряти частини відповіді.

***Безпека***

Django має вбудований захист від ряду поширених векторів атак, таких як CSRF, XSS та SQL-ін'єкція. Ці заходи безпеки допомагають захистити від вразливостей у коді. Команда розробників Django також активно розкриває та швидко виправляє відомі вразливості безпеки.

Flask, з іншого боку, має набагато меншу кодову базу, тому є менша площа поверхні, відкрита для атаки. Однак потрібно буде усунути та виправити вразливі місця безпеки у створеному вручну коді програми, коли вони з'являться.

Оскільки Flask набагато більше залежить від сторонніх розширень, програми будуть настільки безпечними, як і найменш безпечне розширення. Це створює більший тиск на команду розробників, щоб підтримувати безпеку шляхом оцінки та моніторингу сторонніх бібліотек і розширень. Це не означає, що Django за своєю суттю більш безпечний, ніж Flask; це просто простіше захищати наперед та підтримувати протягом усього терміну служби програми.

***Гнучкість***

Flask за своєю конструкцією набагато гнучкіший, ніж Django, і його потрібно розширювати. Через це налаштування Flask зазвичай займає більше часу, оскільки доведеться додати відповідні розширення на основі бізнес-потреб, наприклад, ORM, дозволи, аутентифікацію тощо. Ця початкова вартість призводить до більшої гнучкості в майбутньому для програм, які не підходять до стандартної моделі Django.

Але гнучкість дає розробникам більше свободи та контролю, а це може сповільнити розробку, особливо для великих команд, оскільки потрібно приймати набагато більше рішень.

Оскільки Flask не надає багато обмежень або думок щодо того, як розробляється додаток, розробник може представити свої власні. В результаті дві програми Flask, які є функціонально взаємозамінними в порівнянні пліч-о-пліч, будуть структуровані по-різному. Таким чином, потрібна більш зріла команда, яка розуміє шаблони проектування, масштабованість і важливість тестування для такої гнучкості.

***Винесення фінального рішення***

- Обидві громади дуже активні
- Django старший і має набагато більше учасників
- Flask використовується в більшій кількості проектів
- На Django є більше вмісту

Щоб по-справжньому порівняти ці фреймворки (або екосистеми) з точки зору відкритих кодів, доведеться врахувати Jinja2 та Werkzeug разом із деякими основними бібліотеками та розширеннями Flask, такими як SQLAlchemy / Flask-SQLAlchemy, Alembic / Flask-Alembic та WTForms / Flask-WTF.

Оскільки основна функціональність Flask розподілена між кількома проектами, спільноті важче створити та розвинути необхідну синергію між проектами для підтримки імпульсу.

***Тип програми***

Django чудово створює повнофункціональні веб-додатки із серверними шаблонами. Якщо розробляється статичний веб-сайт або веб-сервіс RESTful, який живить SPA або мобільний додаток, Flask — надійний вибір. Django у поєднанні з Django REST Framework також добре працює в останньому випадку.

***Продуктивність***

Flask працює трохи краще, оскільки вона менша і має менше шарів. Хоча різниця тут незначна, особливо якщо взяти до уваги введення-виведення.

### 3.2.4. Вибір фреймворку для розробки task-менеджеру

Django є повнофункціональним, тому вимагає менше рішень. Таким чином можна рухатися швидше. Однак, якщо є унікальні вимоги до програми, які обмежують кількість можливостей, краще звернутися до Flask.

Зрештою, обидві фреймворки знизили бар’єр для створення веб-додатків, зробивши їх набагато простішими та швидшими у розробці.

З цього випливає, що найкращим вибором буде саме Django. Це надійна структура, яка чудово підходить для запуску програм для всіх – від стартапів до великих компаній; він відмінно реалізує можливості Python і надає всі інструменти, необхідні для достатньої роботи програми.

Навпаки, Flask працює більше як пісочниця для розробників, де вони можуть відточувати свої навички та швидко тестувати рішення, використовуючи різні модулі та бібліотеки. Його краще використовувати для тестування та роботи з менш структурованими об’єктами, а Django – для забезпечення надійного продукту, який відповідає та перевершує очікування.

### 3.3. Послідовність етапів розробки. Робота з бібліотеками для створення програмної системи

Розробка програмної системи Task-менеджер – є головним завданням даного дипломного проекту. Мета роботи даного продукту – це створення зрозумілого та простого об’єднання найбільш часто використовуємого функціоналу на проектах компанії, з якою я на даний момент співпрацюю.

Стартом можна вважати сторінку входу з можливостями авторизації через інші мережі та відновленням паролю за допомогою системи смс сповіщень. В ідеалі у фінальній версії продукту потрібно розглянути варіант 3-D Secure протокол або Двофакторну аутентификацию без можливості використання соціальних мереж. Панель приладів отримає базовий функціонал типовий до Slack та Trello. Навігація повинна бути такою, щоб будь-яка дія у програмній системі була можлива за 2-3 кліки [15]. Проекти повинні займати головний екран та мати простий інтерфейс, який у випадку необхідності буде доповнений статистиками. Головними керувальними та планувальними елементами можна вважати: опис проекту, календар та систему контролю проектних задач. Task-менеджер як програмна система повинен мати можливості постановки завдань: контроль виконаних задач та створення Gantt Chart.

Важливим елементом щоденного контролю виконаних робіт є Kanban дошка. Обов’язково окремо повинні бути винесені календар та гарячі контакти. Гарним доповненням може стати чат для обміну короткими повідомленнями.

### 3.3.1. Імпорт модуля Tkinter

Tkinter - це стандартна бібліотека графічного інтерфейсу для Python. Python в поєднанні з Tkinter забезпечує швидкий і простий спосіб створення додатків з графічним інтерфейсом [14].

```
import tkinter
from tkinter import messagebox
```

***Модуль для головного вікна дизайну***

Перш за все потрібно спроектувати головний екран. На цьому екрані дисплея є мета внесення задач, які потрібно виконати, і час внесення (у секундах), а для кнопки це додати задачу та вийти.

Реалізація:

```
root.bind('&lt;Return&gt;', getting_tasks)


lbl.place(x=0, y=10, width=200, height=25)
lbl_hrs.place(x=235, y=10, width=200, height=25)
todo.place(x=20, y=40, width=160, height=25)
time.place(x=245, y=40, width=170, height=25)
post.place(x=62, y=80, width=100, height=25)
Exit.place(x=302, y=80, width=50, height=25)
WorkingList.place(x=20, y=120, width=395, height=300)
```

```
root = tkinter.Tk()
root.geometry("460x480")
root.title("Students to do List Reminder")
root.rowconfigure(0, weight=1)
root.config(bg="blue")


frame = tkinter.Frame(root)
frame.pack()


lbl = tkinter.Label(root, text="Enter Tasks To Do:", fg="white", bg="blue",
                    font=('Arial', 14), wraplength=200)
lbl_hrs = tkinter.Label(root, text="Enter time (Seconds)", fg="white",
                        bg="blue", font=('Arial', 14), wraplength=200)
todo = tkinter.Entry(root, width=30, font=('Arial', 14))
time = tkinter.Entry(root, width=15, font=('Arial', 14))
post = tkinter.Button(root, text='Add task', fg="white", bg='green',
                      font=('Arial', 16), relief="ridge", bd=5, height=3,
                      width=30, command=getting_tasks)
Exit = tkinter.Button(root, text='Exit', fg="white", bg='red', height=3,
                      font=('Arial Bold', 14), relief="ridge", bd=5, width=30, command=root.destroy)
WorkingList = tkinter.Listbox(root, font=('Arial', 12

                                          ))
if tasks != "":
    actual_time()
```

Щоб отримати повідомлення-попередження користувача про помилки або інформування користувача про успішне завершення завдання:

```
def proceed_time(task):
tkinter.messagebox.showinfo("Notification", "Its Now the Time for : " + task)
```

У наведеному вище коді, який призначений для відображення спливаючих повідомлень, вказується час в секундах.

Бібліотеку підключено та проведено тест базового функціоналу.

***Вікно програми створення списку справ***

Tk() — це віджет верхнього рівня, який використовується для створення головного вікна програми, в якому ми будемо створювати програму зі списком справ на Python.

Метод title() використовується, щоб дати назву програмі, яка в основному відображається вгорі.

Метод mainloop() в основному запускає цикл подій Tkinter, він запускає і відображає все, що написано в коді.

```
window=Tk()
#giving a title
window.title("DataFlair Python To-Do List APP")
window.mainloop()
```

### 3.3.2. Створення макету програми

Метод Frame() — це в основному віджет-контейнер у головному вікні, який використовується для зберігання різних віджетів. Потрібен аргумент, який є головним вікном.

Метод pack() — це менеджер геометрії, який правильно організовує віджет перед тим, як розмістити його в головному вікні в порядку рівня, доки не буде зазначено явно.

Віджет Listbox() буде зберігати всі завдання, які перераховано в програмі списку справ на python. Було наведено три аргументи, перший з яких — це те, де ми хочемо, щоб віджети були розміщені. Якщо висота та ширина не вказано, приймається значення за замовчуванням. Також буде прийняте значення за замовчуванням як вікно, якщо frame_task не згадується явно.

Віджет Scrollbar() використовується для розміщення смуги прокрутки, якщо загальна кількість списків перевищує заданий розмір вікна списку справ. Як і будь-який інший віджет, він потребує аргументів щодо того, де його слід розмістити. Пакується праворуч від віджета frame_task, потім встановлюється його на віджет listbox_task і після цього налаштовується на вертикальній осі, заданій командою listbox_task.yview.

Віджет Button() використовується для створення кнопки. Ми хочемо, щоб кнопки були у головному вікні списку справ, щоб було надано вікно введення. Потім вказується текст, який буде відображатися на кнопці, і, нарешті, у вводі команди задається функція, яка буде викликатися при натисканні кнопки [14].

```
#creating the initial window
window=Tk()
#giving a title
window.title("DataFlair Python To-Do List APP")
#Frame widget to hold the listbox and the scrollbar
frame_task=Frame(window)
frame_task.pack()
#to hold items in a listbox
listbox_task=Listbox(frame_task,bg="black",fg="white",height=15,width=50,font = "Helvetica")
listbox_task.pack(side=tkinter.LEFT)
#Scrolldown in case the total list exceeds the size of the given window
scrollbar_task=Scrollbar(frame_task)
scrollbar_task.pack(side=tkinter.RIGHT,fill=tkinter.Y)
listbox_task.config(yscrollcommand=scrollbar_task.set)
scrollbar_task.config(command=listbox_task.yview)
#Button widget
entry_button=Button(window,text="Add task",width=50,command=entertask)
entry_button.pack(pady=3)
delete_button=Button(window,text="Delete selected task",width=50,command=deletetask)
delete_button.pack(pady=3)
mark_button=Button(window,text="Mark as completed ",width=50,command=markcompleted)
mark_button.pack(pady=3)
window.mainloop()
```

***Визначення функції списку справ***

Хорошою практикою кодування є розміщення всіх функцій на початку коду. Ці функції будуть викликані при натисканні кнопок і виконуватимуть роботу відповідно.

entertask() використовується для додавання завдання до списку. При виклику цієї функції відкривається інше вікно, в яке можна написати завдання, і воно додається до списку. Функція add() викликається при натисканні кнопки Додати завдання у вікні, що відкрилося.

Введені дані зберігаються в input_text, якщо змінна порожня, то відображається попереджувальне повідомлення, інакше воно вставляється в список справ python.

deletetask() використовується для видалення вибраного елемента, функція listbox_task.curselection() повертає кортеж, у якому перший елемент є індексом вибраного елемента. Потім функція delete() використовується для видалення елемента, що відповідає індексу в програмі списку робіт на python

markcompleted() використовується, щоб позначити елемент у списку справ як виконаний. У цій функції зберігається вміст вибраного завдання у змінній під назвою temp_marked.

Оновлюється значення, додаванням «✔» в кінці. Після цього функція delete() видаляє вибране завдання, а функція insert() вставляє оновлене значення за цим конкретним індексом.

```
def entertask():
    #A new window to pop up to take input
    input_text=""
    def add():
        input_text=entry_task.get(1.0, "end-1c")
        if input_text=="":
            tkinter.messagebox.showwarning(title="Warning!",message="Please Enter some Text")
        else:
            listbox_task.insert(END,input_text)
            #close the root1 window
            root1.destroy()
    root1=Tk()
    root1.title("Add task")
    entry_task=Text(root1,width=40,height=4)
    entry_task.pack()
    button_temp=Button(root1,text="Add task",command=add)
    button_temp.pack()
    root1.mainloop()
```

```
#function to facilitate the delete task from the Listbox
def deletetask():
    #selects the selected item and then deletes it
    selected=listbox_task.curselection()
    listbox_task.delete(selected[0])
#Executes this to mark completed
def markcompleted():
    marked=listbox_task.curselection()
    temp=marked[0]
    #store the text of selected item in a string
    temp_marked=listbox_task.get(marked)
    #update it
    temp_marked=temp_marked+" ✔"
    #delete it then insert it
    listbox_task.delete(temp)
    listbox_task.insert(temp,temp_marked)
```

Tkinter та GUI бібліотеки налаштовані та працюють справно. Можна переходити до наступного етапу.

### 3.3.3. Покрокова реалізація

Крок 1. Створено папку з іменем «Диспетчер задач». Відкрито редактор коду.

Крок 2. Створено файл Python з ім'ям task_manager.py.

Крок 3: Тепер починається написання коду програмного забезпечення. Спочатку йде створенням функції реєстрації. Функція реєстрації буде приймати ім’я користувача, під яким користувач збирається створити свій початковий запис і запитувати пароль для цього початкового запису. Приведений нижче код проясняє ситуацію.

```
def signup():
    print("Please enter the username by which you \
    wanna access your account")
    username = input("please enter here:  ")
    password = input("Enter a password:  ")
```

Крок 4: Тепер створено функцію user_information, яка буде брати дані з функції реєстрації та створювати текстовий файл для збереження даних користувача. Наведений нижче код покаже, як все буде відбуватися:

```
# pssd means password, ussnm is username
def user_information(ussnm, pssd):
    name = input("enter your name please: ")
    address = input("your address")
    age = input("Your age please")
    ussnm_ = ussnm+" task.txt"

    f = open(ussnm_, 'a')
    f.write(pssd)
    f.write("\nName: ")
    f.write(name)
    f.write('\n')
    f.write("Address :")

    f.write(address)
    f.write('\n')
    f.write("Age :")
    f.write(age)
    f.write('\n')
    f.close()


def signup():
    print("Please enter the username by which you\
    wanna access your account")
    username = input("please enter here:  ")
    password = input("Enter a password:  ")
    user_information(username, password)
```

Крок 5: Після того, як текстовий файл буде створено функцією user_information, щоб закодувати функцію входу в систему. Функція входу в систему вказує ім'я користувача та запитує пароль, пов'язаний з ним. Як тільки користувач вводить пароль, функція перевіряє, совпадає з паролем, зберігається в текстовому файлі, з введеним.

```
def login():
    print("Please enter your username ")
    user_nm = input("Enter here: ")

    # Password as entered while logging in
    pssd_wr = (input("enterr the password"))+'\n'
    try:
        usernm = user_nm+" task.txt"
        f_ = open(usernm, 'r')

        # variable 'k' contains the password as saved
        # in the file
        k = f_.readlines(0)[0]
        f_.close()

         # Checking if the Password entered is same as
         # the password saved while signing in
        if pssd_wr == k:
            print(
                "1--to view your data \n2--To add task \n3--Update\
                task status \n4--View task status")
            a = input()
        else:
            print("SIR YOUR PASSWORD OR USERNAME IS WRONG , Plz enter Again")
            login()
    except Exception as e:
        print(e)
        login()
```

Крок 6: На цьому етапі робиться, щоб після входу користувача в систему можна було його запитати війти в свій початковий запис. Це можна зробити, викликавши функцію входу в систему в кінці функції входу. Далі функція входу буде виглядати так:

```
def signup():
    print("Please enter the username by which you wanna access your account")
    username = input("please enter here:  ")
    password = input("Enter a password:  ")
    user_information(username, password)
    print("Sir please proceed towards log in")
    login()
```

Крок 7: Треба виконувати чотири важливі функції, вказані в блоці «вхід у систему»: функція для перегляду даних, функція для додавання задач до даних, функція для оновлення статусу задач і функція для перегляду статусу задач. На цьому етапі треба виконати функцію входу в систему, виконавши частину if-else після введення запиту користувача:

```
def login():
    print("Please enter your username ")
    user_nm = input("Enter here: ")

    # Password as entered while logging in
    pssd_wr = (input("enterr the password"))+'\n'
    try:
        usernm = user_nm+" task.txt"
        f_ = open(usernm, 'r')

        # variable 'k' contains the password as
        # saved in the file
        k = f_.readlines(0)[0]
        f_.close()

         # Checking if the Password entered is same
         # as the password saved while signing in
        if pssd_wr == k:
            print(
                "1--to view your data \n2--To add task \n3--Update\
                task \n4--VIEW TASK STATUS")
            a = input()
```

```
            if a == '1':
                view_data(usernm)
            elif a == '2':
                # add task
                task_information(usernm)
            elif a == '3':
                task_update(user_nm)
            elif a == '4':
                task_update_viewer(user_nm)
            else:
                print("Wrong input ! ")
        else:
            print("SIR YOUR PASSWORD OR USERNAME IS WRONG")
            login()

    except Exception as e:
        print(e)
        login()


    def view_data(username):
        pass

    def task_information(username):
        pass

    def task_update(username):
        pass

    def task_update_viewer(username):
        pass
```

Команда пропускається, щоб дозволити записати ім'я та аргумент, а також запобігти появі повідомлення про помилку в коді редактора.

Крок 8: Закодуємо блок даних представлення:

```
def view_data(username):
    ff = open(username, 'r')
    print(ff.read())
    ff.close()
```

Крок 9: Щоб закодувати задачі інформаційного блоку, потрібно пам’ятати про основні концепції обробки тексту в Python. Запитаємо користувача, що він хоче додати, і, по його бажанню, повторимо цикл тільки раз і попросимо його ввести свою задачу і мету, по якій він хоче завершити задачу:

```
def task_information(username):
    print("Sir enter n.o of task you want to ADD")
    j = int(input())
    f1 = open(username, 'a')

    for i in range(1, j+1):
        task = input("enter the task")
        target = input("enter the target")
        pp = "TASK "+str(i)+' :'
        qq = "TARGET "+str(i)+" :"

        f1.write(pp)
        f1.write(task)
        f1.write('\n')
        f1.write(qq)
        f1.write(target)
        f1.write('\n')

        print("Do u want to stop press space bar otherwise enter")
        s = input()
        if s == ' ':
            break
    f1.close()
```

Було додано повідомлення про те, що якщо користувач хоче припинити додавання завдань раніше за кількість разів, що він хоче, тоді у нього буде шанс. Це робить програму дуже зручною для користувача.

Крок 10: Оновлення статусу завдань відбувається в аналогічній концепції обробки тексту в Python. Перше, що треба - зберегти номер задачі, номер якої завершено, який виконується, а яка ще не запущена, з відміткою дати та часу з ними.

```
def task_update(username):
    username = username+" TASK.txt"
    print("Please enter the tasks which are completed ")

    task_completed = input()
    print("Enter task which are still not started by you")

    task_not_started = input()
    print("Enter task which you are doing")

    task_ongoing = input()
    fw = open(username, 'a')
    DT = str(datetime.datetime.now())

    fw.write(DT)
    fw.write("\n")
    fw.write("COMPLETED TASK \n")
    fw.write(task_completed)
    fw.write("\n")
    fw.write("ONGOING TASK \n")
    fw.write(task_ongoing)
    fw.write("\n")
    fw.write("NOT YET STARTED\n")
    fw.write(task_not_started)
    fw.write("\n")
```

Крок 11: Тепер у залишилася функція перегляду оновлення задач. Ця функція так же проста, як функція view_data.

```
def task_update_viewer(username):
    ussnm = username+" TASK.txt"
    o = open(ussnm, 'r')
    print(o.read())
    o.close()
```

Найважливіша задача, яка все ще залишається, - це кодування основної функції та керування потоком команд програми з самої основної функції.

Крок 12: Основна функція.

```
if __name__ == '__main__':
    print("WELCOME TO ANURAG`S TASK MANAGER")
    print("sir are you new to this software")
    a = int(input("Type 1 if new otherwise press 0 ::"))

    if a == 1:
        signup()
    elif a == 0:
        login()
    else:
        print("You have provided wrong input !")
```

Це знаменує кінець блоку коду.

Далі логіка програмної системи будується ідентично.

За допомогою Python фреймворку Django вказаний каркас програми та інші розширення, засновані на даній логіці отримують гарний інтерфейс.

### 3.4. Опис основних функцій додатку. Інструкція користувача

Завершено роботу над створенням програмної системи task-менеджер. Для написання було обрано мову програмування Python, фреймворк Django та за основу для налаштування логіки використану серію бібліотек, з яких можна сміливо виділити Tkinter та GUI.

Розглянемо переваги та недоліки отриманої програмної системи.

Переваги:

- Логічне побудова системи постановки завдань;
- Зручна карткова система;
- Основні операції із завданнями вимагають не більше 3 кліків;
- Інтеграція із сторонніми сервісами;
- Функціональна система фільтрації за кольорами;
- Комунікація з колективом усередині сервісу;
- Поділ завдань із пріоритетності;
- Є різні можливості під різні бізнес-процеси;
- Система повідомлень;
- Зручне планування спринтів;
- Сервіс повністю безкоштовний;
- У систему інтегровано все необхідне керування усіма етапами розробки;
- Фільтрація та сортування карток;
- Гарні діаграми;
- Календар;
- Робота із кількома проектами;
- Система обліку часу;
- Перегляд історії змін;
- Сучасні інтерфейс.

Недоліки:

- Малі можливості кастомізації;
- Немає інтеграції із хмарними сервісами;
- Немає можливості експорту даних;
- Не можна оцінювання виконавців;
- Немає служби підтримки;
- Мультимовність відсутня;
- Немає розумного пошуку;

- Немає ієрархії співробітників;
- Демократична система прав;
- Незручно для великих командах від 20 осіб;
- Бракує дизайнерських фішок (э лише світла та темна теми).

Враховуючи, що система може стабільно підтримуватись самою командою – це гарний результат з великою кількістю переваг.

**Знайомство з програмною системою task-менеджер**

План системи:

- Login pages (Сторінки входу)
- Dashboard (Панель приладів)
- Navigation (Навігація)
- Projects (Проекти)
- Project Management (Керування проектами)
- Project Planner (Планування проекту)
- Tasks (Завдання)
- Kanban Desk (Канбан дошка)
- Calendar (Календар)
- Contacts (Контакти)
- Chat & Inbox (Чат&Вхідні)

### 3.4.1. Login pages (Сторінки входу)

Робота з програмною системою починається зі сторінки входу (Рис.3.1):

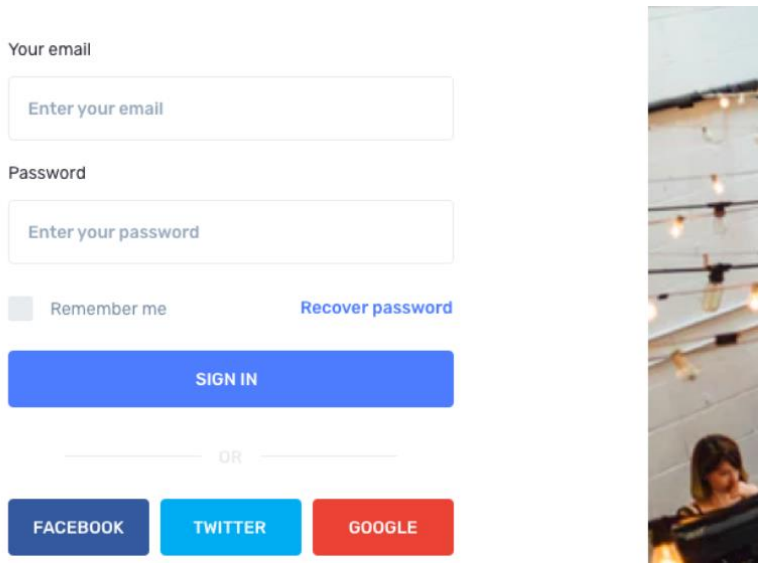
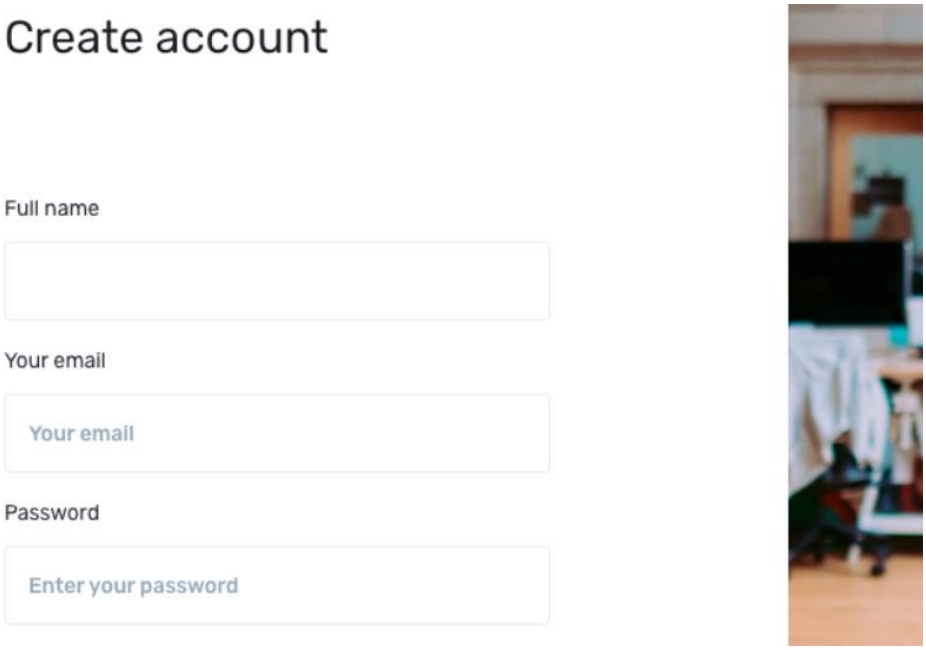



Рис. 3.1. Сторінка входу в систему

У випадку, якщо аккаунт відсутній, є можливість його створити (Рис.3.2):



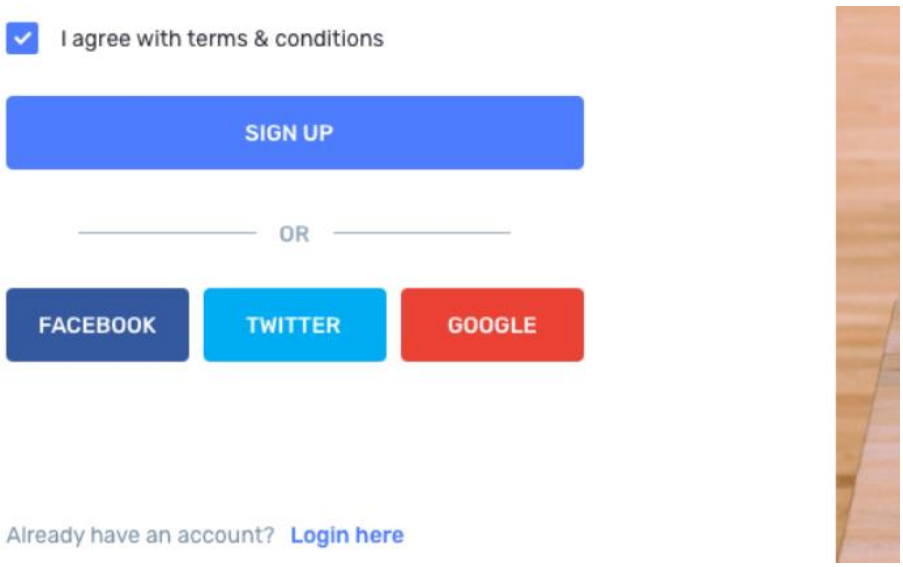



Рис. 3.2. Сторінка створення аккаунту

Для відновлення паролю на першій сторінці можна знайти відповідну кнопку (Рис.3.3):

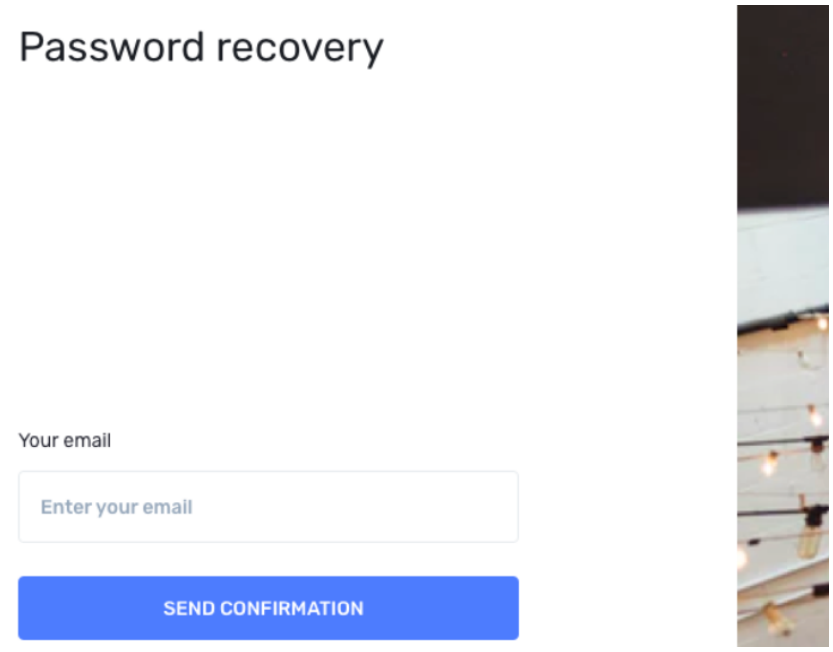



Рис. 3.3. Сторінка відновлення паролю

Щоб підтвердити зміни, потрібно ввести 6-значний код, який буде знаходитись на пошті (Рис.3.4):

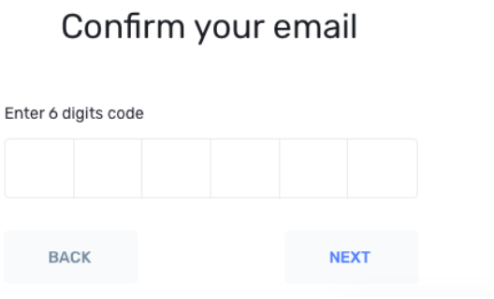



Рис. 3.4. Операція відновлення паролю за допомогою коду

### 3.4.2. Dashboard (Панель приладів)

Панель приладів має базовий функціонал (Рис.3.5):

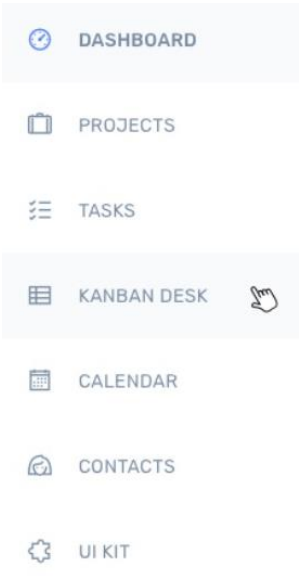



Рис. 3.5. Головна панель

Передбачено, що статистика нових та виконаних задач буде відображено за допомогою відповідних графіків (Рис.3.6):

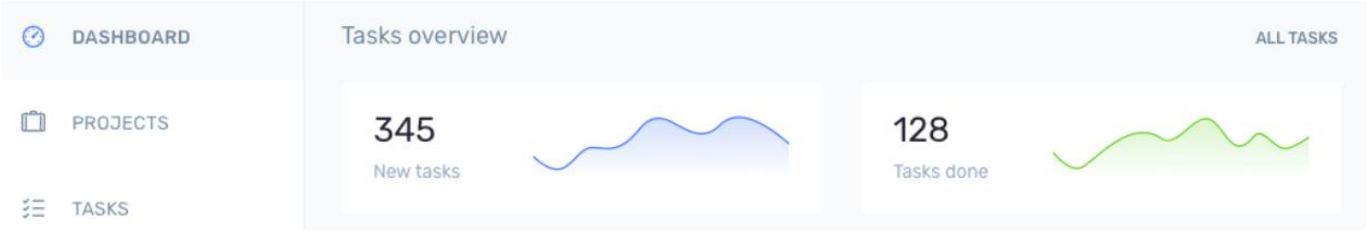



Рис. 3.6. Головна сторінка з кількісною статистикою задач

На головній панелі розташований перелік задач користувача з можливістю залишати помітки про виконання (Рис.3.7):

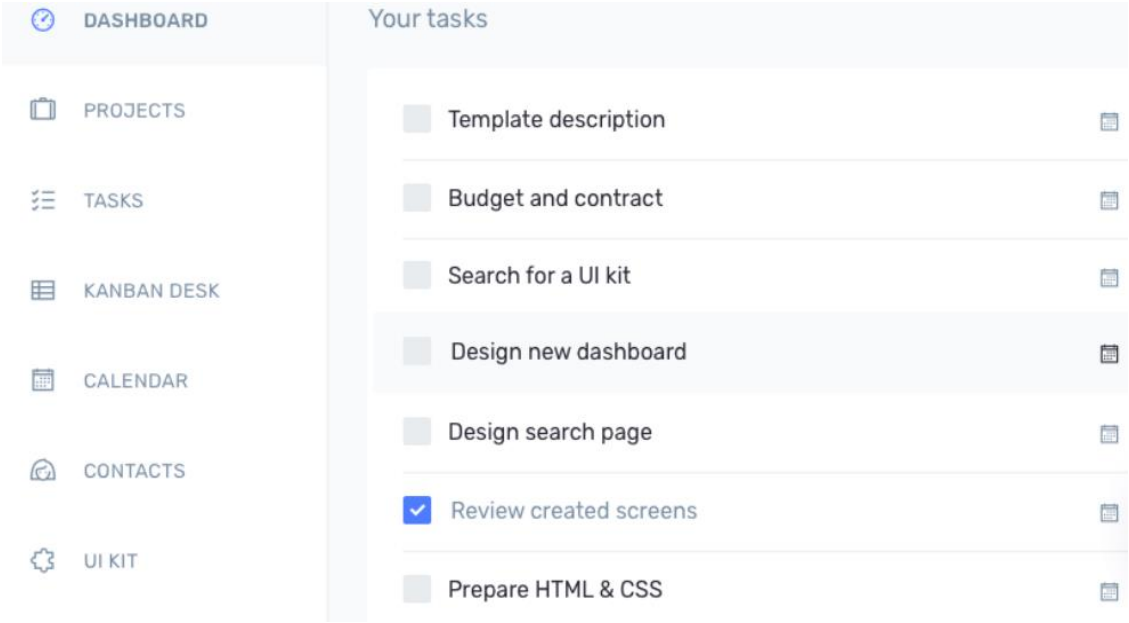



Рис. 3.7. Відображення задач та головна панель

Кожне завдання супроводжується кількісним відображенням саб-завдань та коментарів (Рис.3.8):

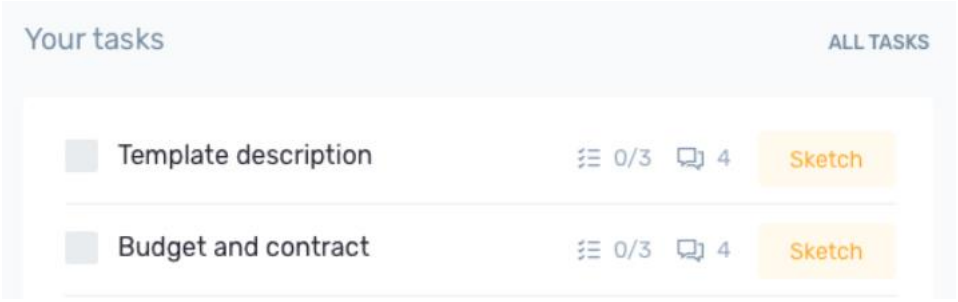



Рис. 3.8. Відображення задач у контексті сабтасків

В деяких проектах AdTech (напрям роботи мого департаменту) є необхідність ведення статистики витрат та прибутків від реклами (Рис.3.9):

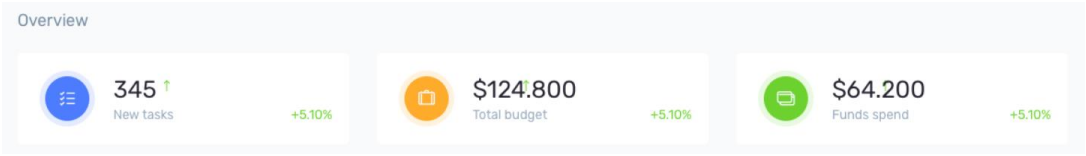



Рис. 3.9. Бюджет задач та статистика витрат

Готовність проекту до релізу відображається у процентному колі або графіку активностей на вибір (Рис.3.10):

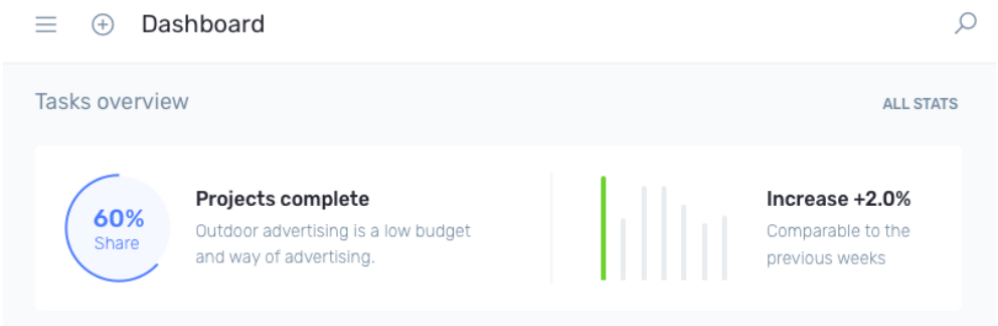



Рис. 3.10. Готовність проекту до релізу

Програмна система має у своєму складі календар з можливістю нотифікацій зустрічей (Рис.3.11):

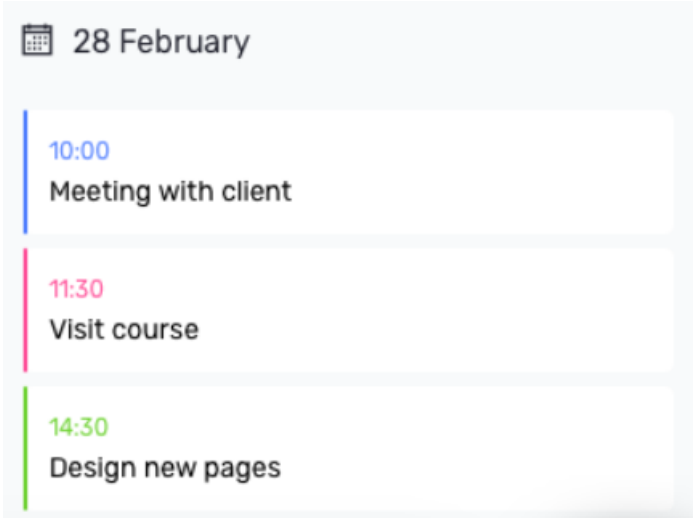



Рис. 3.11. Система нотифікацій та планування календарю

Програмну систему оснащено модульним пошуком від Google (Рис.3.12):

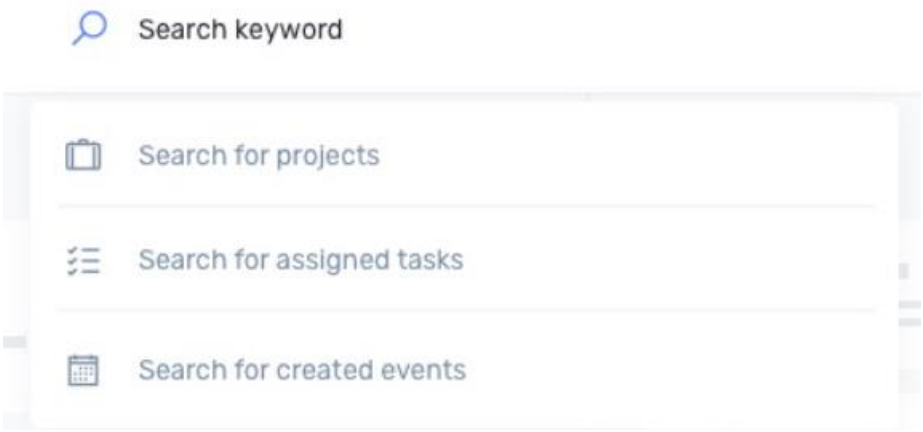



Рис. 3.12. Система пошуку

Головна панель зроблена максимально простою та має у своєму складі лише базовий необхідний функціонал (Рис.3.13):

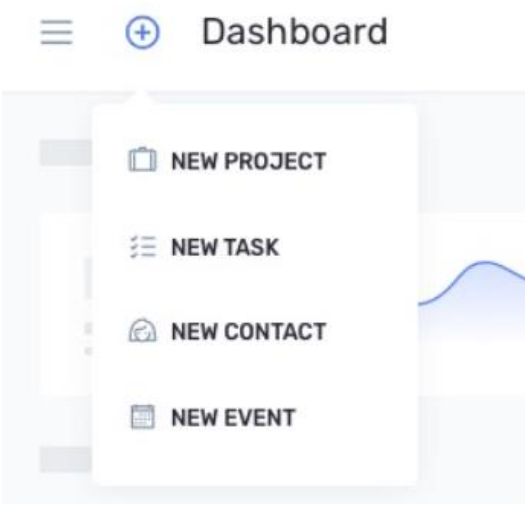



Рис. 3.13. Можливості головної панелі

Можливість замінити текст відповідними позначками (Dashboard):

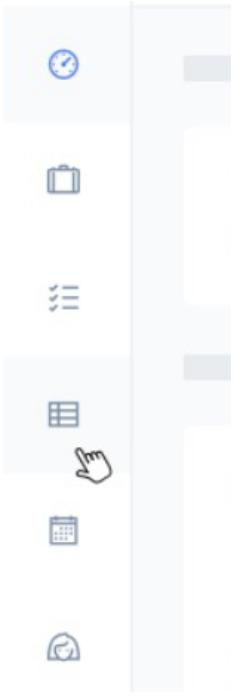


Рис. 3.14. Зміни інтерфейсу позначок

Можливість змінювати тему для головної панелі (Рис.3.15):

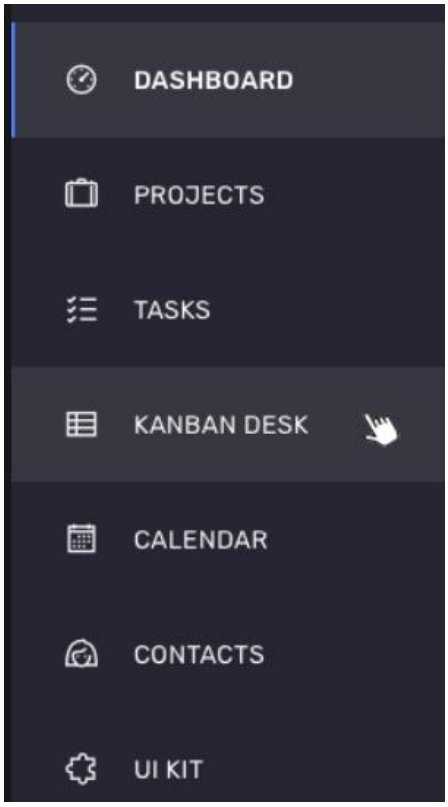



Рис. 3.15. Зміни кольору інтерфейсу

### 3.4.3. Navigation (Навігація)

Можливість прив’язки кольору до певних задач або категорій:

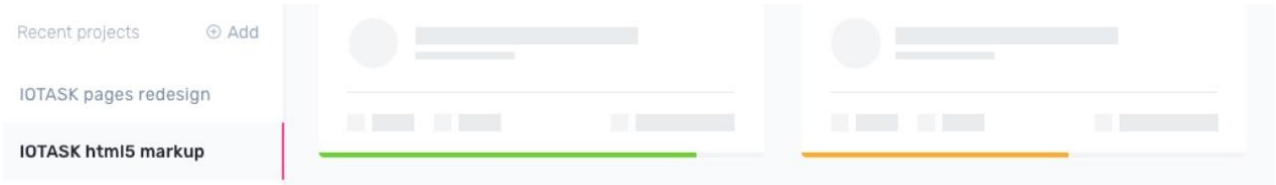



Рис. 3.16. Розподіл задач за кольором

Відображення статистики за допомогою діаграм або візуальних елементів (Рис.3.17):

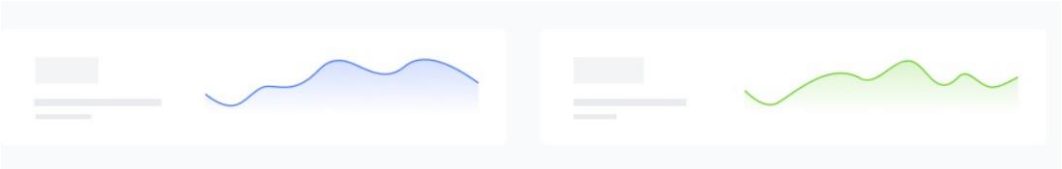


Рис. 3.17. Відображення діаграм

### 3.4.4. Projects (Проекти)

Проекти можуть бути відсортовані за ім’ям або статусом (Рис.3.18):

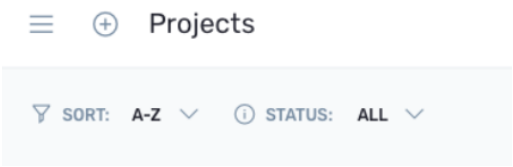



Рис. 3.18. Сортування проектів

Учасники проекту можуть бути запрошені спеціальним листом через пошту (Рис.3.19):

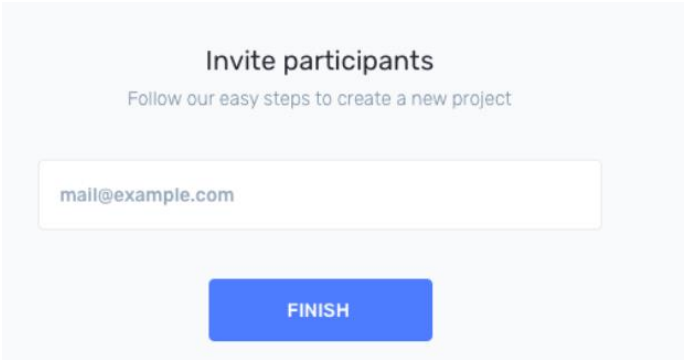



Рис. 3.19. Система запрошення через пошту учасників

Новий проект створюється за 3 кліки (Рис.3.20):

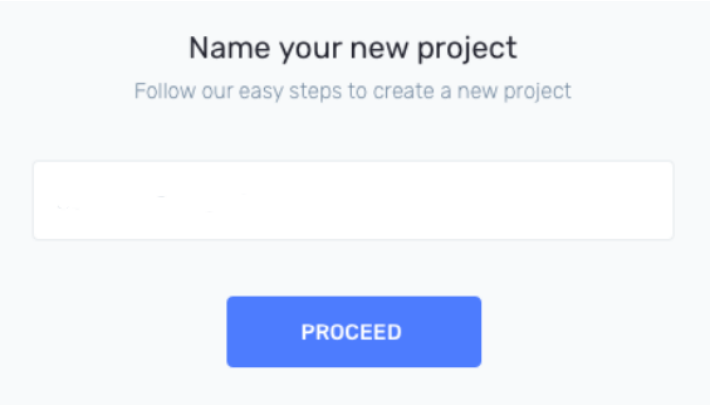



Рис. 3.20. Створення нового проекту

Всі проекти завжди відображено на головному екрані:

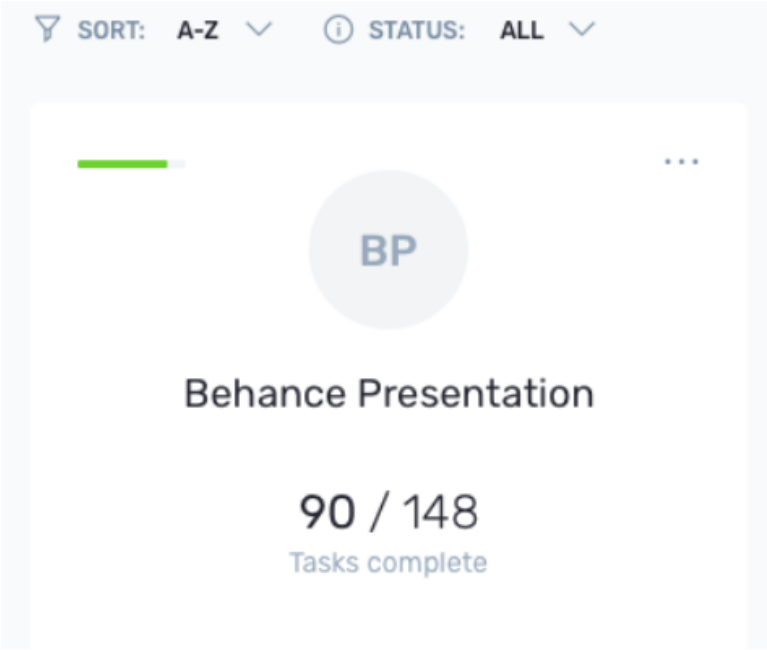



Рис. 3.21. Відображення проекту на головному екрані

Діаграма статистики виконаних задач, яка оновлюється в реальному часі:

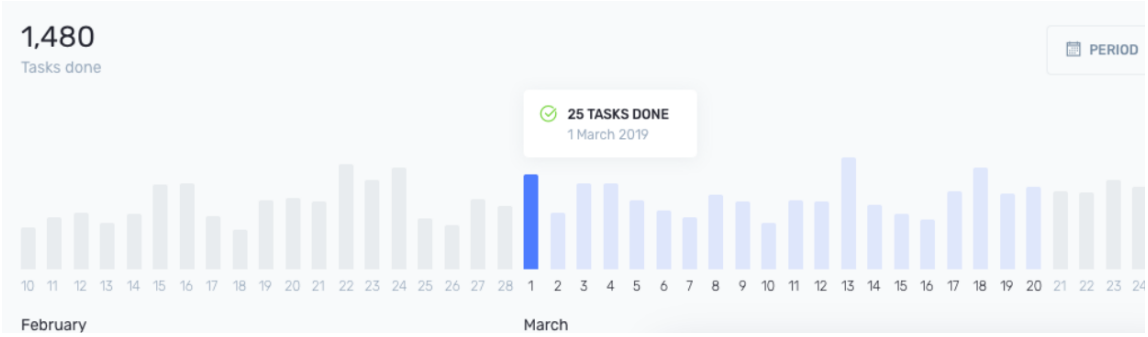



Рис. 3.22. Створення діаграми статистики за допомогою віджетів

### 3.4.5. Project Management (Керування проектами)

Кожен проект має місце для короткої анотації до нього:

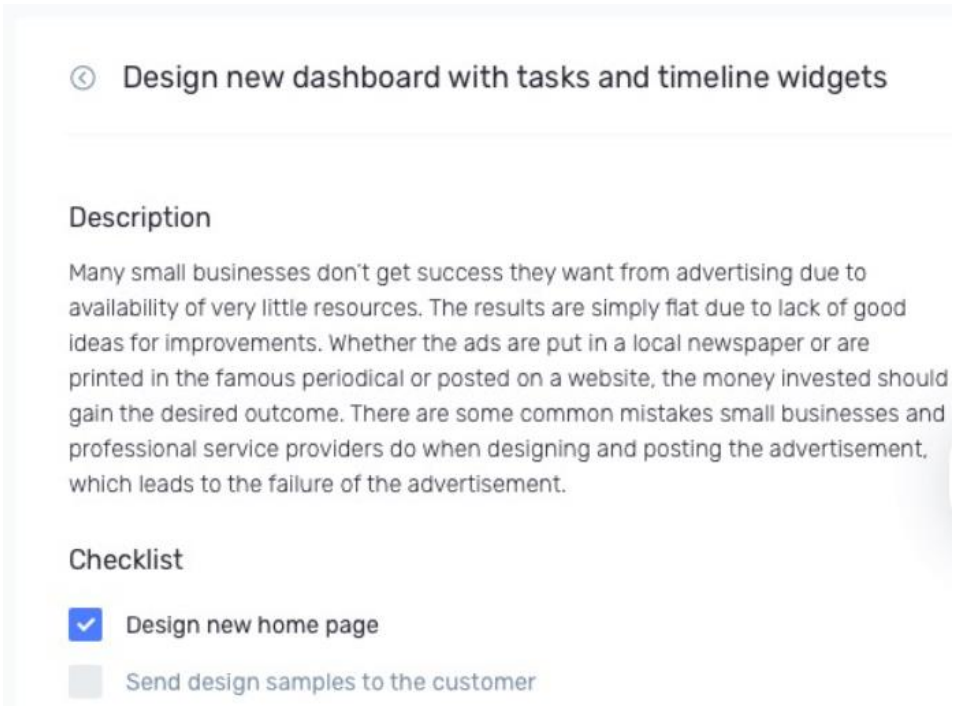



Рис. 3.23. Відображення опису проекту

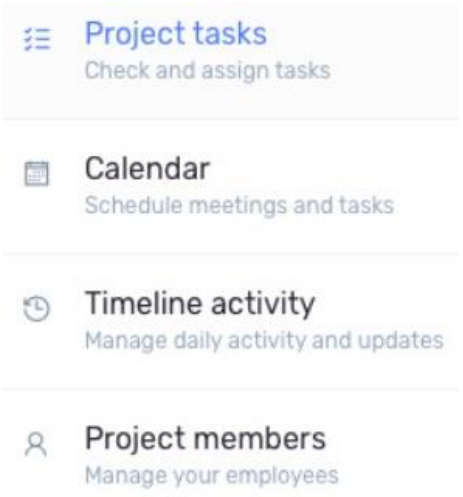



Рис. 3.24. Система контролю проектних задач

Календар задач та подій (Рис.3.25):

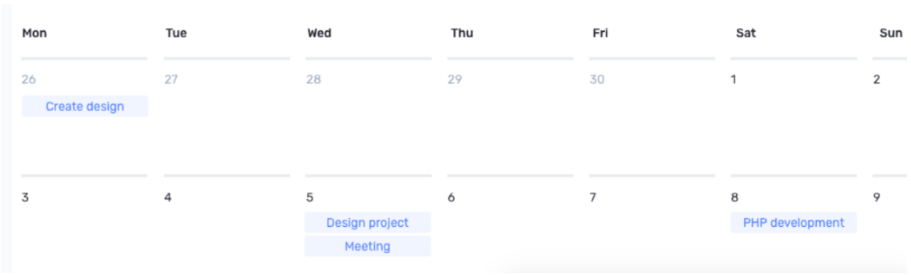



Рис. 3.25. Відображення календаря задач

### 3.4.6. Project Planner

Можливість додавати задачі до відповідного їй розташування (Рис.3.26):

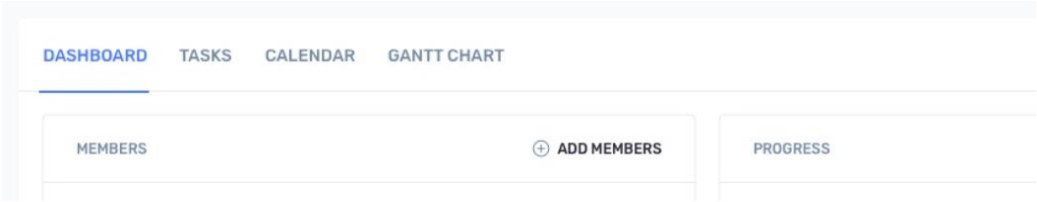



Рис. 3.26. Відображення календаря задач

Можливості розширення задач та розбиття на під задачі (Рис.3.27):

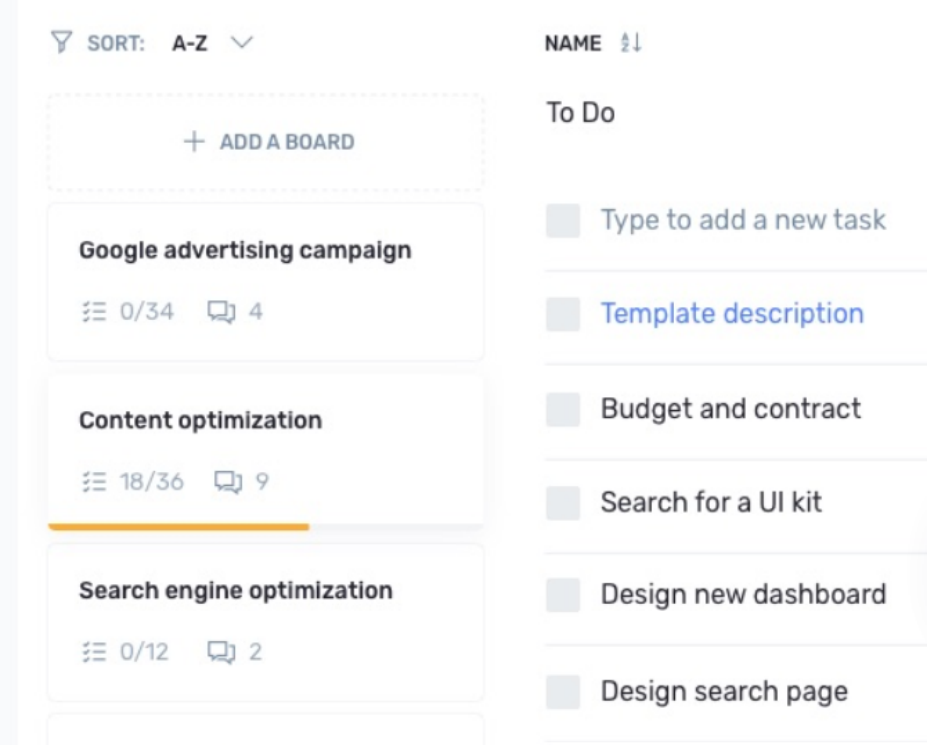



Рис. 3.27. Можливість додавати під задачі

Система виконано/не виконано (Рис.3.28):

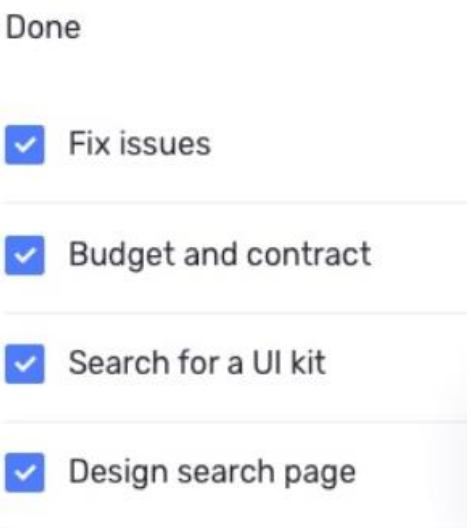



Рис. 3.28. Контроль виконаних задач

Відображення прогресу проекту або релізу на Gantt Chart (Рис.3.29):

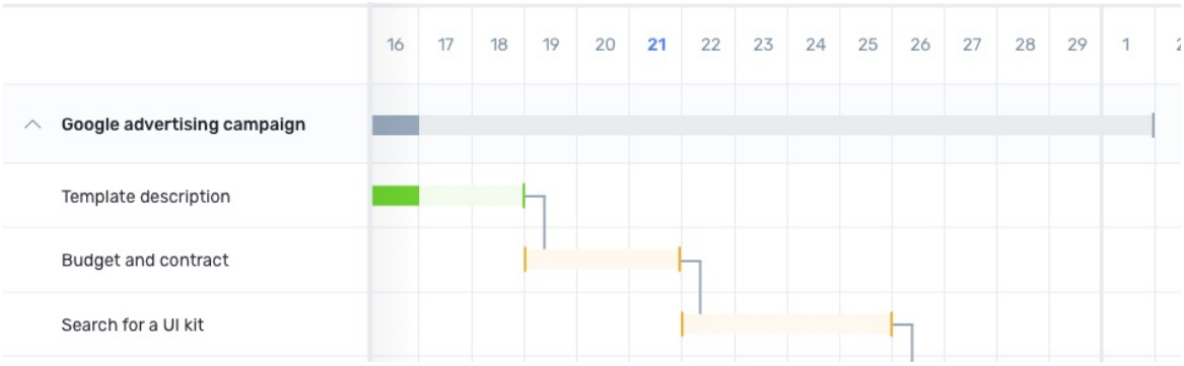



Рис. 3.29. Створення Gantt Chart

### 3.4.7. Tasks (Завдання)

Інструкції та формати відображення задач, що виконуються у даних проміжок часу (Рис.3.30):

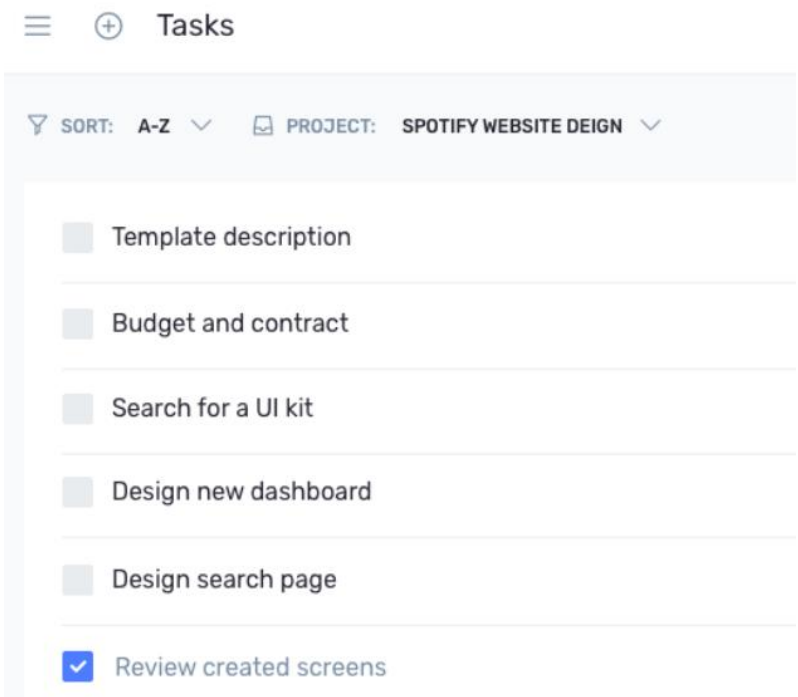



Рис. 3.30. Відображення списку задач

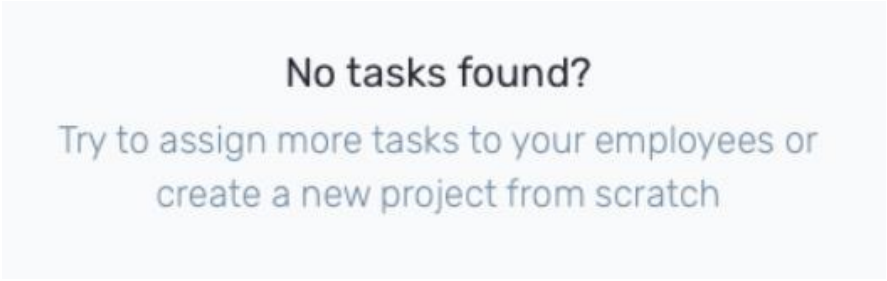



Рис. 3.31. Короткі інструкції для додавання задач

### 3.4.8. Kanban Desk (Канбан дошка)

Базова Kanban дошка для створення, обробки та фіналізування задач (Рис.3.32):

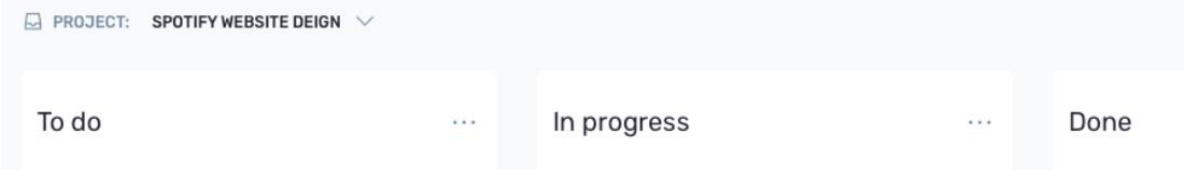



Рис. 3.32. Стандарт дошки для розміщення задач (Kanban Desk)

Варіанти налагодження Kanban дошки (Рис.3.33):

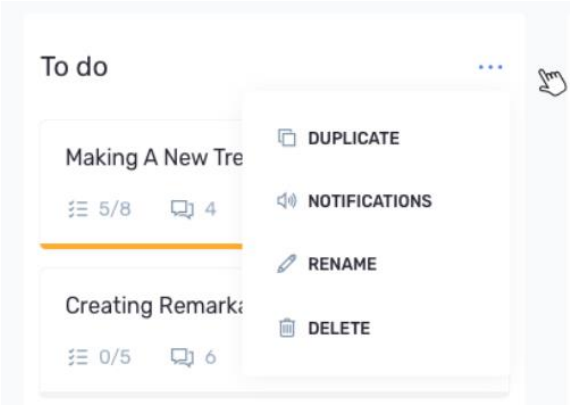



Рис. 3.33. Можливості змін для Kanban Desk

Функціонал для створення нових задач на Kanban дошці (Рис.3.34):

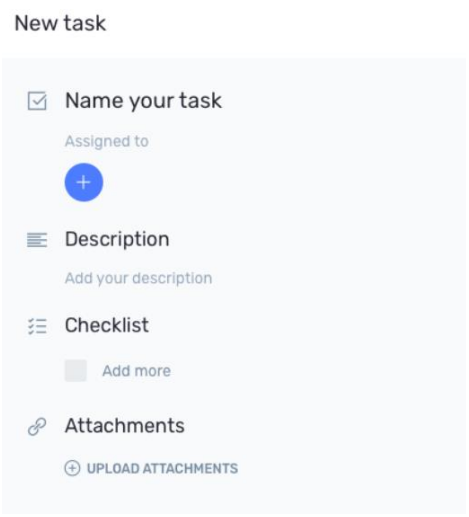



Рис. 3.34. Вікно створення нової задачі

### 3.4.9. Calendar (Календар)

Відображення календаря проектних задач та зустрічей (Рис.3.35):

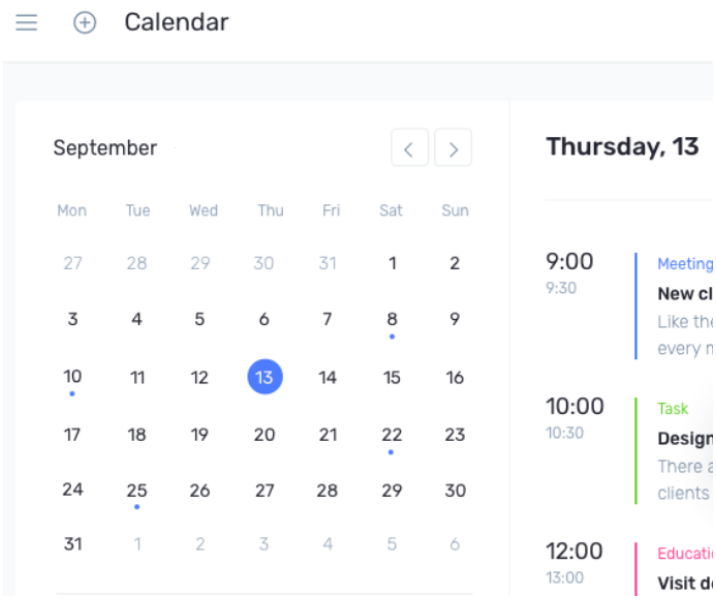



Рис. 3.35. Керування календарем: відображення зустрічей

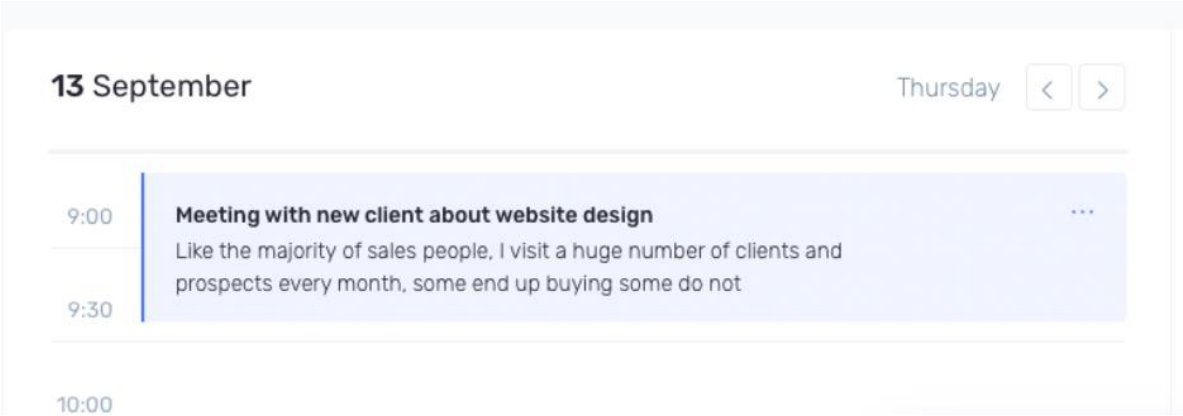



Рис. 3.36. Огляд опису запланованої зустрічі

### 3.4.10. Contacts (Контакти)

Контроль та сортування гарячих контактів (Рис.3.37):

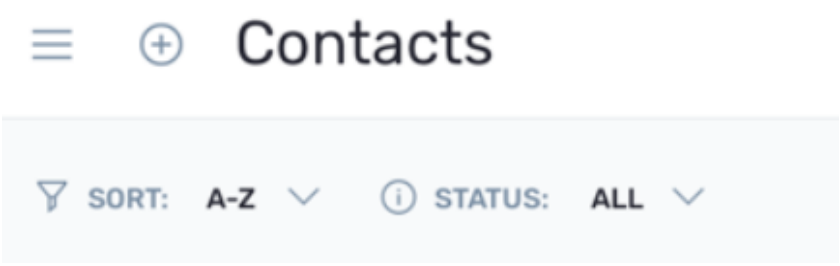



Рис. 3.37. Сортування списку контактів (розробників та клієнтів)

### 3.4.11. Chat & Inbox (Чат&Вхідні)

Можливість додавання каналів різних проектів (Рис.3.38):

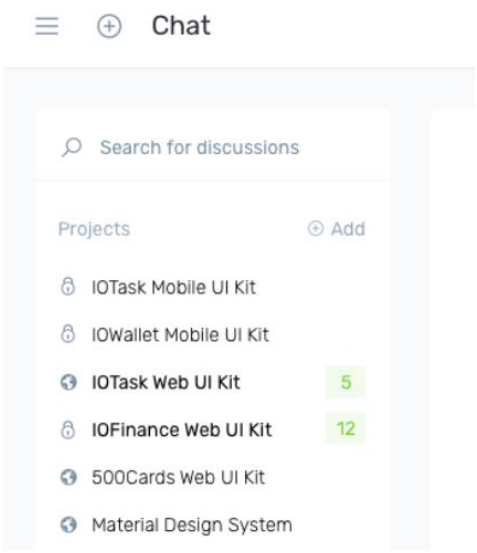



Рис. 3.38. Канали проектів

Створення діалогів в каналі проекту та можливість запрошення нових людей (Рис.3.39):


Рис. 3.39. Створення нового діалогу з контактом мережі

Створення та обробка повідомлень між членами команди (Рис.3.40):

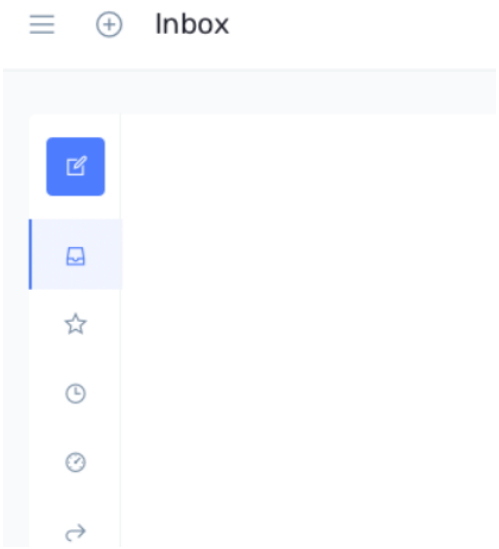



Рис. 3.40. Відображення можливостей створення повідомлень

## Висновки до розділу 3

Запропонована програмна система task-менеджер для керування IT проектами може виконувати наступні функції:

- До системи можна додавати всі необхідні файли, посилання на матеріали, зауваження, інструкції, коментарі до поставлених завдань.
- Реалізовано нагадування про закінчення терміну проекту чи окремих його етапів розсилається автоматичною системою кожному з учасників процесу та зокрема відповідальним особам персонально.
- Зберігається історія роботи над поставленими завданнями: кожен із учасників процесу може публікувати коментарі та коригування в ході виконання робочих завдань, а також простежити історію внесення змін та осіб, які їх вносили.
- Є можливість відстежити трудовитрати кожного з працівників та оцінити ефективність роботи всього персоналу.
- Учасники трудового процесу можуть обмінюватися миттєвими повідомленнями під час роботи над проектом.
- Система передбачає можливість розсилання інформації про закінчення чергового етапу роботи електронною поштою.
- Програма надає можливість зберігання інформації та історії щодо кожного з проектів, включаючи співробітників, задіяних у роботі, їх трудовитрати, час на реалізацію, загальна кількість проектів у яких брав участь кожен окремий співробітник, динаміка виконання проекту, відсоток завершеності, порушення термінів, проблеми та складності у роботі, комплекс заходів вжитих їх усунення, звітність за всіма проектами, повна історія взаємодії з клієнтами.
- У разі потреби система дозволяє встановити обмежений доступ до інформації для окремих користувачів, якщо ці дані не мають безпосереднього відношення до їхньої роботи. Співробітнику буде відкритий доступ лише до конкретних завдань та даних, що знаходяться в межах його компетенції та під його відповідальністю, що забезпечує додаткову безпеку та конфіденційність за проектом усередині компанії.

Компанія, що використовує у роботі Систему керування проектами, значно економить свій час і налаштовується на максимально ефективний результат роботи за допомогою оптимізації взаємодії працівників. Керівник отримує у своє розпорядження інструмент, який дозволяє керувати процесом та здійснювати контроль за виконанням завдань на кожному етапі проходження замовлення, включаючи кожного працівника, який бере участь у робочому процесі, миттєво реагувати та вносити коригування у хід виконання роботи.

Все це можна робити незалежно від місцезнаходження співробітників. Система керування завданнями дозволяє здійснювати спільну роботу над проектом, навіть якщо люди працюють у різних підрозділах, перебувають у бізнес-поїздці чи працюють віддаленим методом. Це забезпечує додаткову зручність, оскільки не завжди є можливість або необхідність швидко зібрати весь персонал разом.

Реалізована система працює справно та має гарні показники у використанні.

# ВИСНОВКИ

Для забезпечення будь-якого проекту системою керування задачами потрібно зробити базову модель з відкритим для вдосконалень кодом, з чого випиває, що це практично апробована модель.

Кожна організація в умовах жорсткої конкуренції прагне оптимізації робочих процесів, значного скорочення витрат у процесі виконання замовлення, максимально ефективно застосовувати трудові ресурси, ліквідувати втрату та спотворення актуальної інформації, яка застосовується в роботі. Однак не кожна компанія здатна втілити в життя всі перелічені зміни якнайкраще. Ідеальним помічником у цій ситуації може стати Система керування завданнями (Task management software).

Система керування завданнями (Task management software) допомагає модернізувати роботу над поставленими завданнями, зробивши її максимально ефективною, незалежно від обсягу штату компанії та місця перебування працівників. Кожному з учасників цього проекту надається перелік доручень та завдань, які ставляться безпосередньо перед ним, а також ступінь виконання та пріоритетність завдання в окремий проміжок часу порівняно з іншими завданнями.

Запропонована програмна система task-менеджер для керування IT проектами може виконувати наступні функції:

- До системи можна додавати всі необхідні файли, посилання на матеріали, зауваження, інструкції, коментарі до поставлених завдань.
- Реалізовано нагадування про закінчення терміну проекту чи окремих його етапів розсилається автоматичною системою кожному з учасників процесу та зокрема відповідальним особам персонально.
- Зберігається історія роботи над поставленими завданнями: кожен із учасників процесу може публікувати коментарі та коригування в ході виконання робочих завдань, а також простежити історію внесення змін та осіб, які їх вносили.
- Є можливість відстежити трудовитрати кожного з працівників та оцінити ефективність роботи всього персоналу.

- Учасники трудового процесу можуть обмінюватися миттєвими повідомленнями під час роботи над проектом.
- Система передбачає можливість розсилання інформації про закінчення чергового етапу роботи електронною поштою.
- Програма надає можливість зберігання інформації та історії щодо кожного з проектів, включаючи співробітників, задіяних у роботі, їх трудовитрати, час на реалізацію, загальна кількість проектів у яких брав участь кожен окремий співробітник, динаміка виконання проекту, відсоток завершеності, порушення термінів, проблеми та складності у роботі, комплекс заходів вжитих їх усунення, звітність за всіма проектами, повна історія взаємодії з клієнтами.
- У разі потреби система дозволяє встановити обмежений доступ до інформації для окремих користувачів, якщо ці дані не мають безпосереднього відношення до їхньої роботи. Співробітнику буде відкритий доступ лише до конкретних завдань та даних, що знаходяться в межах його компетенції та під його відповідальністю, що забезпечує додаткову безпеку та конфіденційність за проектом усередині компанії.

Компанія, що використовує у роботі Систему керування проектами, значно економить свій час і налаштовується на максимально ефективний результат роботи за допомогою оптимізації взаємодії працівників.

Система керування завданнями дозволяє здійснювати спільну роботу над проектом, навіть якщо люди працюють у різних підрозділах, перебувають у бізнес-поїздці чи працюють віддаленим методом. Це забезпечує додаткову зручність, оскільки не завжди є можливість або необхідність швидко зібрати весь персонал разом.

Реалізована система працює справно та має гарні показники у використанні.

# СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Управління проєктами. Політична енциклопедія [Електронний ресурс] – Режим доступу: http://surl.li/ayuil (дата звернення 15.10.2021р) – Назва з екрана.

2. Project Management [Електронний ресурс] – Режим доступу: https://www.investopedia.com/terms/p/project-management.asp (дата звернення 10.09.2021р) – Назва з екрана.

3. Особливості системи управління проєктами в it-компаніях [Електронний ресурс] – Режим доступу: http://www.agrosvit.info/?op=1&z=3205&i=14 (дата звернення 15.10.2021р) – Назва з екрана.

4. Кар'єра в IT: чим займається Project Manager, плюси та мінуси професії – Режим доступу: https://dou.ua/lenta/articles/project-manager-pros-and-cons/ (дата звернення 20.10.2021р) – Назва з екрана.

5. Як правильно оцінити IT-проект [Електронний ресурс] – Режим доступу: https://itukraine.org.ua/yak-pravylno-otsinut-it-proekt.html (дата звернення 10.11.2021р) – Назва з екрана.

6. Поняття життєвого циклу проекту – Режим доступу: https://pidru4niki.com/12810419/ekonomika/zhittyeviy_tsikl_proektu (дата звернення 15.11.2021р) – Назва з екрана.

7. Teamwork Formula – Режим доступу: https://www.rathboneresults.com/resources/teamwork-formula-5-principles-for-hardwiring-employee-motivation-to-teamwork (дата звернення 29.11.2021р) – Назва з екрана.

8. Understand the principles of effective teamwork [Електронний ресурс] – Режим доступу: https://pressbooks.bccampus.ca/learningtolearnonline/chapter/understand-the-principles-of-effective-teamwork/ (дата звернення 29.11.2021р) – Назва з екрана.

9. Основи управління IT проектами [Електронний ресурс] – Режим доступу: https://ela.kpi.ua/bitstream/123456789/34480/1/2019_Osnovy_upravlinnia.pdf (дата звернення 10.11.2021р) – Назва з екрана.

10. Управління IT проектами [Електронний ресурс] – Режим доступу: http://surl.li/ayuje (дата звернення 12.10.2021р) – Назва з екрана.

11. ТОП-7 найкращих таск-менеджерів для віддаленого керування робочим процесом [Електронний ресурс] – Режим доступу: http://surl.li/ayujh (дата звернення 20.11.2021р) – Назва з екрана.

12. Переваги і недоліки мови Python [Електронний ресурс] – Режим доступу: https://blog.ithillel.ua/ua/articles/perevagi-i-nedoliki-movi-python (дата звернення 25.11.2021р) – Назва з екрана.

13. Flask vs Django in 2021: Which Framework to Choose? [Електронний ресурс] – Режим доступу: https://hackr.io/blog/flask-vs-django (дата звернення 27.11.2021р) – Назва з екрана.

14. Python interface to Tcl/Tk [Електронний ресурс] – Режим доступу: https://docs.python.org/3/library/tkinter.html (дата звернення 15.10.2021р) – Назва з екрана.

15. Task managers: what to do and how to choose your own [Електронний ресурс] – Режим доступу: https://studway.com.ua/task-menedzheri/ (дата звернення 17.11.2021р) – Назва з екрана.