МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
«КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАДИМА ГЕТЬМАНА»
навчально-науковий інститут
«Інститут інформаційних технологій в економіці»

Кафедра інформаційних систем в економіці

галузь знань 12 «Інформаційні технології»

спеціальність 122 « Комп'ютерні науки»

Форма навчання: заочна

КВАЛІФІКАЦІЙНИЙ БАКАЛАВРСЬКИЙ ПРОЕКТ

на тему

Комп'ютерні технології візуалізації алгоритмів сортування та пошуку

здобувача Кондратюка Артема Ігоровича
_____

Науковий керівник: к.е.н., доцент
_____ Краснюк М. Т.

Бакалаврський дипломний проект допущений до захисту в кзаменаційній комісії з атестації здобувачів вищої освіти
Завідувач кафедри: к.е.н., доцент.
_____ Тішков Б. О.

Київ 2024

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ ВЕЛИЧИН І ТЕРМІНІВ

ДП — Дипломний поект

ПП — Програмний продукт

ПЗ — Програмне забезпечення

ТЗ — технічне завдання — документ який встановлює основне призначення, показники якості, техніко-економічні та спеціальні вимоги до виробу, обсягу, стадії розроблення та складу конструкторської документації.

Документ – інформація, що зафіксована в електронному або паперовому вигляді з певними реквізитами.

Фреймворк — це абстракція, в якій код що забезпечує загальну функціональність може бути вибірково змінений або доповнений додатковим кодом, написаним користувачем, таким чином дозволяючи створювати специфічне ПЗ на існуючій базі. Він забезпечує стандартний спосіб створення та розгортання додатків і є універсальним програмним середовищем для багаторазового використання, яке надає певні функціональні можливості як частину більшої програмної платформи для полегшення розробки програмних додатків, продуктів і рішень. Програмні рамки можуть включати допоміжні програми, компілятори, бібліотеки коду, набори інструментів та інтерфейси прикладного програмування (API), які об’єднують усі різні компоненти для забезпечення можливості розробки проекту або системи.

DOC, DOCX, DOT, DOTX, RTF, ODT – формат електронних документів, які створюються, редагуються та відкриваються за допомогою програми Microsoft Word.

JPG, JPEG, PNG, SVG – формати зображень.

PDF – формат електронних документів, створений компанією Adobe Systems, який можна відкрити під керівництвом будь-якої операційної системи.

API — інтерфейс прикладного програмування, інтерфейс програмування застосунків — це зв’язок між комп’ютерами або між комп’ютерними програмами. Це тип програмного інтерфейсу, який пропонує послуги іншим частинам програмного забезпечення. Документ або стандарт, який описує, як побудувати або використовувати таке з’єднання чи інтерфейс, називається специфікацією API. Вважається, що комп’ютерна система, яка відповідає цьому стандарту, реалізує або відкриває API. Термін API може посилатися або на специфікацію, або на реалізацію.

Python — є високорівневою, інтерпретованою мовою програмування загального призначення. Його філософія дизайну підкреслює читабельність коду з використанням значних відступів.

JavaScript — це мова програмування, яка є однією з основних технологій всесвітньої мережі, поряд з HTML і CSS. Це високорівнева, «точно вчасно» (just-in-time) скомпільована мова. Ця мова має динамічне типування, є об'єктно-орієнтованою базованою на прототипах, пріоритетно функціональною. JavaScript багатопарадигмальний, підтримує керований подіями, функціональний та імперативний стилі програмування. Він має інтерфейси прикладного програмування (API) для роботи з текстом, датами, регулярними виразами, стандартними структурами даних і об’єктною моделлю документа (DOM).

HTML — Мова розмітки гіпертексту, є стандартною мовою розмітки для документів, призначених для відображення у веб-браузері. Часто супроводжується такими технології, як каскадні таблиці стилів (CSS) і мовами сценаріїв, такими як JavaScript.

Visual Studio Code (vscode) — Visual Studio Code, також відомий як VS Code, — це редактор вихідного коду, створений Microsoft для Windows, Linux

і macOS. Функції включають підтримку налагодження, виділення синтаксису, інтелектуальне завершення коду, фрагменти, рефакторинг коду та вбудований Git. Користувачі можуть змінювати тему, комбінації клавіш, параметри та встановлювати розширення, які додають додаткові функції.

Linux — це сімейство операційних систем з відкритим вихідним кодом на основі ядра Linux, ядра операційної системи, вперше випущеного на 17 вересня 1991, Лайнус Торвальдс. Linux зазвичай упаковується в дистрибутив Linux.

SSH — Протокол Secure Shell (SSH) — це криптографічний мережевий протокол для безпечної роботи мережевих служб у незахищеній мережі. Його призначення — це віддалений вхід до систем і виконання командного рядка в умовах віддаленого терміналу (командного інтерфейсу). Програми SSH імплементовані на архітектурі клієнт-сервер, з’єднуючи екземпляр клієнта SSH з сервером SSH. SSH працює як багаторівневий набір протоколів, що складається з трьох основних ієрархічних компонентів: транспортний рівень забезпечує аутентифікацію сервера, конфіденційність і цілісність; протокол аутентифікації користувача перевіряє користувача на сервері; і протокол з'єднання мультиплексує зашифрований тунель у кілька логічних каналів зв'язку.

ВСТУП

Мета дипломного проекту (ДП) — вивчення та застосування методик творчого розв’язання практичних завдань з розробки програмних продуктів (ПП) виходячи з отриманих знань, професійних навичок та умінь відповідно до міжнародних стандартів та світового досвіду в розробці ПП, цифрових технологій та інновацій з використанням інструментів складових галузей програмування та розробки інформаційних технологій.

Завданням ДП є систематизація та закріплення знань отриманих під час навчання за освітньо-професійною програмою підготовки бакалаврів та проходження виробничої практики в вищому навчальному закладі ДВНЗ “Київський національний економічний університет імені Вадима Гетьмана”, виконання практичних завдань під час самостійного навчання та використання отриманих навичок на практиці.

Об’єктом дослідження ДП є розробка ПП який надає можливість візуалізувати та дослідити ефективність різних алгоритмів сортування та пошуку.

Предметом дослідження є розробка ПП який надає можливість досліджувати алгоритми сортування та пошуку з можливістю їх порівняння та додавання нових в середині ПП. Уклін є на можливість додавання та редагування власних алгоритмів користувачем. Повинна бути можливість додавати функціонал до ПП шляхом додавання користувацьких розширень.

Метою розробки даного ПП є підвищення ефективності вивчення та аналізу популярних алгоритмів сортування та пошуку і написання та модифікування нових. Зменшення ймовірності помилок в алгоритмах при аналізі та висновках.

Завданням створення ПП є створення безкоштовного продукту з відкритим кодом який дозволить аналізувати різноманітні алгоритми та порівнювати їх.

Практичним застосуванням розробленого ПП має бути аналіз та дослідження існуючих алгоритмів сортування та пошуку, можливість розробляти та редагувати нові алгоритми, проводити їх покрокове відлагодження.

РОЗДІЛ 1. ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1. Характеристика предметної галузі та об’єкта дослідження

Дослідження та розробки ефективності алгоритмів проводяться для покращення продуктивності процесів, у яких вони використовуються. Оптимізація будь-якого виробничого процесу пов’язана з упорядкованим виконанням підзадач.

Крім того, розроблено методи збору даних для спостереження за станом і траєкторією окремих компонентів у цілому потоці. Це важливий крок для оптимізації послідовностей, щоб уникнути вузьких місць і затримок для інших процесів. Це досягається шляхом створення оптимальних шляхів для кожного процесу, які усувають вузькі місця і, отже, пришвидшують шлях для досягнення найвищої ефективності. Йдеться про найкраще використання доступних ресурсів для прискорення діяльності та досягнення найефективнішого поєднання функцій. Йдеться про встановлення нового рекорду продуктивності в порівнянні з іншими.

Як постачальник обладнання та консалтинг для покращення процесу. Вивчення різноманітних алгоритмів взаємодії дозволяє досягти цих цілей.

Найкорисніше таке дослідження корисне для використання в галузях, де є потреба обробляти, перенаправляти та організовувати велику кількість різноманітних даних, які можна організувати за певним шаблоном.

Наприклад, подивіться військовий сектор, який використовував класифікацію шаблонів для організації величезних обсягів даних для стратегічного планування; або організація університету, яка використовує шаблони. Це такі, як транспортування, виробництво та роздрібна торгівля, які вимагають високого рівня точності та безперебійного потоку товарів. Ще один напрямок, який також набуває популярності зі зростанням попиту на швидкі та точні системи сортування пошти, системи складського обліку, організації

складання на підприємствах для виготовлення складних виробів, що включають велику кількість різноманітних деталей, і баз даних, з яких найбільше потрібна надійність і ефективність.

Постійний перехід до нових форм персональних і комерційних електронних пристроїв, а також поява Інтернету речей, який робить ці пристрої взаємопов’язаними, ще більше збільшив частку в

Крім того, передача великих обсягів різноманітної інформації та аналізу через електронні системи забезпечує надійну платформу для обробки, зберігання та розповсюдження таких даних. Ці фактори, а також інші призвели до сильного впровадження класифікації шаблонів і штучного інтелекту серед великих компаній. У 2017 році понад 46 мільярдів доларів США було витрачено на великі дані та інструменти розпізнавання образів для різних програм у фінансах, автомобільній промисловості, охороні здоров’я, виробництві та роздрібній торгівлі. Ключовими сегментами ринку класифікації шаблонів і розпізнавання шаблонів, які були оцінені в цьому звіті, є хмарні обчислення та програмне забезпечення як послуга.

У таких галузях і з такими завданнями необхідно приймати рішення щодо організації виробничих процесів, для чого проводиться аналіз різних способів організації даних. Для цього досліджено велику кількість способів виконання ланцюга завдань до кінцевого результату. Для цих цілей може знадобитися імітація виконання елементарних частин процесу для досягнення найкращого результату. Також проводиться моделювання реального виробничого процесу, де, наприклад, може знадобитися припустити, що певний економічний закон виконується в мережі цін на товар. Результати моделювання можуть слугувати корисними знаннями при розробці методу загального призначення для виконання процесів інформації та прийняття рішень.

Впровадження методу загального призначення для здійснення процесів інформації та прийняття рішень

Для здійснення процесів отримання інформації та прийняття рішень на початковій стадії промислового розвитку складних систем необхідні рішення та знання про матеріали, процеси та цінності в глобальному ланцюжку вартості, а також їх кількісні та якісні характеристики.

На вибір певного рішення поставленої задачі впливає велика кількість різноманітних факторів. Це складність виконуваних підпроцесів, їх кількість, можливість і вартість автоматизації та комп'ютеризації завдань, кількість персоналу та можливість залучення додаткових людських ресурсів, вартість обслуговування інфраструктури, необхідність створення резервної інфраструктури. і ресурсів у разі збою основного технологічного ланцюга. У випадку транспортних і логістичних систем високий ступінь автоматичного контролю різних підсистем у виробничому ланцюгу може гарантувати високий ступінь пунктуальної та своєчасної доставки. Варто відзначити, що такі рішення вимагають високого ступеня інтеграції між підсистемами. У випадку виробничого ланцюга на основі обладнання, крім вимог системної інтеграції, складність автоматизованих систем ще більше збільшується, оскільки системи керування дуже складні.

IT-системи розробляються в середовищі, яке характеризується економічною ефективністю з точки зору початкових інвестиційних витрат, витрат на обслуговування, а також довгострокових операційних витрат. Отже, витрати на систему необхідно порівнювати з можливостями, які вона може запропонувати. На даний момент автоматизація не націлена на системи зі значним ступенем міжмашинного зв'язку (M2M). Таким чином, реалізація високорівневої автоматизованої системи, хоча технічно можлива, не пропонує багато переваг для великомасштабних систем, які не пов'язані між собою кабелем або датчиками.

1.2. Аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі

На сьогодні існує багато способів створити мережевий застосунок. Будь-яка сучасна високорівнева мова програмування має фреймворк або механізм на базі якого можна наростити необхідну бізнес-логіку та додати інтерактивний користувацький інтерфейс (REST API, GraphQL, зібрана зі сторони серверу статична сторінка, веб-сторінка з прогресивним завантаженням). Найчастіше даний механізм або фреймворк реалізує веб-сервер який базується на патерні програмування MVC (модель-вид-контролер) який є супроводжуваний допоміжними інструментами специфічними для кожного окремого фреймворку. Такі бібліотеки є на сьогоднішній день найзручнішим способом створення інтерактивних веб-застосунків. Більшість сучасних веб-ресурсів користуються подібними рішеннями. Без цих рішень складно уявити собі сучасний інтернет. Тим не менш, навіть з використанням таких інструментів, важко повністю перейти на архітектуру програмного забезпечення або структуру, розроблену для інтерактивних веб-сервісів. Інша причина цього полягає в тому, що, якщо ваша власна бізнес-логіка або представлення даних повинні залишатися поза цією структурою, ви не зможете безпечно використовувати цю структуру для створення всієї програми, тобто всіх залежностей вашого проекту. Більшість фреймворків знають про вимоги та обмеження конкретних структур даних і дозволяють налаштовувати або замінювати ці об’єкти. Якщо ваша програмна логіка та структура бази даних базуються на об’єктно-орієнтованих структурах даних, ви можете вибрати відповідну структуру для своєї бізнес-логіки та даних.

Для взаємодії з елементами сторінки найпоширенішим способом є використання мови JavaScript на веб сторінках разом з сотнями написаних базуючись на цій мові програмування бібліотек для роботи з мережевим інтерфейсом. Існують десятки доступних бібліотек, які забезпечують функціональність, яка додає абсолютно новий рівень інтерактивності. Деякі з

них добре відомі, але багато хто тільки починає знайомство зі світом програмування на Javascript. Простий пошук у блозі бібліотек Javascript для веб-сторінок надасть багато нових проектів для розробників і програмістів Javascript. З огляду на велику різноманітність цього програмування та широкий спектр доступних мов програмування та інструментів розробки, важливо вибрати правильний інструмент для правильного завдання.

На сьогоднішній день також почали з’являтись інші інструменти здатні на подібну взаємодію такі як наприклад WebAssembler (WASM) на базі якого інші сучасні високорівневі мови програмування розробляють свої компілятори та імплементатори даючи можливість їх використання в веб застосунках що відкриває широкі можливості для портування застосунків, інструментів, ігор розроблених для настільних комп’ютерів як веб застосунків і розробки інших потужних веб-інструментів для професіоналів. Але в цього рішення є один значний недолік — WASM базується на мові JavaScript, його команди транслюються на цю мову з використанням великої кількості оптимізацій, але не дивлячись на це продуктивність виконуваної програми страждає. Також, бінарні файли для виконання базованих на WASM потужних професійних застосунків або просто складних застосунків можуть сягати значних розмірів і при поганому поєднанні з інтернетом можуть завантажуватися впродовж значного проміжку часу. Також в середовищі веб-сторінки мережеві переглядачі доволі сильно обмежують доступні для використання ресурси комп’ютера що також є лімітуючим фактором. В платформи WASM безперечно є свої проблеми, як і в будь-якої нової технології на початку її шляху, тим не менш розробку технологій WASM підтримують та спонсорують великі корпорації. Інтеграція вже є присутньою в більшості сучасних веб переглядачів. Можна впевнено сказати що це є дуже цікава та перспективна технологія яка одного дня дозволить реалізуватись великій кількості складних та корисних проектів які не можливо було б реалізувати з використанням класичних на сьогодні веб технологій.

РОЗДІЛ 2. РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ/ПІДСИСТЕМИ

В даному розділі буде розглядатись основні вимоги до ПП розробленого в процесі виконання ДП. Розробка вимог до ПП складається з декількох етапів:

- Знаходження вимог (визначення та задоволення потреб цільового користувача ПП)
- Аналіз вимог (перевірка повноти та цілісності вимоги)
- Специфікація (документування вимог до ПП)
- Тестування вимог

Вимоги до програмного продукту за характером поділяють на функціональні і нефункціональні.

До вимог функціонального характеру відносять:

- Бізнес вимоги
- Вимоги користувача
- Функціональні вимоги

До нефункціональних за характером вимог відносять:

- Бізнес правила (обмеження предметної області)
- Системні вимоги (до програмних інтерфейсів, надійності та обладнання)
- Атрибути якості
- Зовнішні системи та інтерфейси
- Обмеження

Так як для виконання ДП використовується методологія **водоспаду** етап аналізу та розробки вимог повинен бути повністю завершений до початку проектування та розробки ПП, а власне проектування та розробка не може завершитися до завершення аналізу вимог.

Методологія розробки водоспаду — це техніка управління проектом, яка забезпечує своєчасне завершення кінцевого проекту. Він базується на ітераційній концепції, згідно з якою програмне забезпечення слід часто оновлювати, щоб відповідати мінливим потребам користувачів. По суті, часте оновлення програмного забезпечення скорочує витрати на розробку, оскільки оновлення можна легко створювати та розповсюджувати з невеликими витратами. У порівнянні зі швидкою ітерацією розробка каскаду є менш ризикованою, оскільки вона базується на попередніх ітераціях та оновленнях. Це добре відома методологія розробки програмного забезпечення, яка допомагає компаніям ефективно створювати нові програмні продукти.

Методологія водоспадної розробки висвітлює етапи створення нових програмних продуктів. Нижче наведено чотири етапи проекту розвитку водоспаду:

1. Аналіз. На етапі аналізу підприємства аналізують свої поточні системи та потреби в розробці плану продукту. Це передбачає збір даних, аналіз систем, оцінку потреб і визначення пріоритетів завдань. По суті, аналіз даних і систем готує команди до створення нових програмних продуктів. Збір даних включає проведення опитувань і спілкування з користувачами, щоб зрозуміти, що потрібно створити і як це має виглядати. Після аналізу даних і вимог користувачів команди створюють ідеальний дизайн продукту перед тим, як реалізувати його в коді.

2. Дизайн. На етапі проектування команди зосереджуються на створенні компонентних інтерфейсів для свого нового продукту. Вони також розробляють екрани для введення користувачами та структури бази даних для цілей зберігання. По суті, дизайнери створюють візуально привабливі дизайни перед тим, як реалізувати їх у коді або шаблонах коду. Порівняно з моделюванням у 3D-просторі, проектування можна порівняти зі створенням моделей перед установкою на місце. Тут моделі замінюють візуальні елементи, такі як будівлі чи персонажі з унікальною зовнішністю.

3. Впровадження. Під час впровадження розробники пишуть вихідний код для компонентів програми та інтерфейсів компонентів, використовуючи такі мови програмування, як C++ або Python (залежно від проекту). Після написання вихідного коду розробники тестують свої програми за допомогою

модульних тестів, щоб переконатися, що кожен компонент працює належним чином. Після завершення тестування розробники впроваджують свої програми в програми за допомогою часу виконання, наприклад JavaScript або PHP (залежно від проекту). Після завершення впровадження бізнес-користувачі можуть запускати тестування програми, надаючи відгук про функціональні можливості для вдосконалення майбутніх дизайнів програм.

4. Виробництво. На етапі виробництва менеджери щодня контролюють продуктивність програми після завершення впровадження. Вони також навчають співробітників, як використовувати нову програму, коли вона стане доступною для загального використання. Після того як співробітники ознайомляться з тим, як користуватися новою програмою, менеджери сприяють продуктивності співробітників, встановлюючи цілі щодо того, як часто співробітники повинні використовувати програму, і навчаючи співробітників тому, як ефективно використовувати її самостійно.

Вимоги до ПП складовою частиною технічного завдання (ТЗ).

Вимоги до ПП розробленого в ході виконання ДП розроблені методом мозкового штурму, аналізу нормативної документації та аналізу бізнес-процесів.

2.1. Аналіз і специфікація вимог до інформаційної системи/підсистеми

В даному підрозділі виконується аналіз і специфікація вимог до ПП.

Специфікація і аналіз вимог являють собою повний опис поведінки системи що розробляється. В ході цієї процедури досліджуються всі взаємодії які користувачі мають з ПП. Цей процес встановлює основу для угоди між замовниками та підрядниками чи постачальниками про те, як повинен функціонувати програмний продукт. Специфікація вимог до програмного забезпечення — це сувора оцінка вимог перед більш конкретними етапами проектування системи, і її мета — зменшити подальше перепроектування. Він також має забезпечити реалістичну основу для оцінки вартості продукту, ризиків і графіків. При належному використанні специфікації вимог до програмного забезпечення можуть допомогти запобігти збою проекту програмного забезпечення.

У документі зі специфікацією вимог до програмного забезпечення перераховано достатні та необхідні вимоги для розробки проекту. Щоб отримати вимоги, розробник повинен мати чітке і повне розуміння продуктів, що розробляються. Це досягається шляхом детального та постійного спілкування з командою проекту та замовником протягом усього процесу розробки програмного забезпечення.

У процесі специфікації вимог до програмного забезпечення однією з головних проблем для розробників є висунення вимог, які надають інформацію, необхідну для максимально ефективного виконання проекту. Вимоги визначають, як досягти бажаної мети проекту з точки зору функціональності, масштабованості, безпеки та доступності. У свою чергу, специфікація системних вимог може служити важливою методологією для забезпечення того, щоб вимоги були адекватно деталізованими, досяжними та доступними для відображення вимог.

Документ специфікації вимог до програмного забезпечення описує бажаний програмний продукт. Ці вимоги включають залежність продукту від інших компонентів системи. Вимоги включатимуть списки різних елементів системи, а також вимоги до системного процесу та інформацію про інтерфейси між цими елементами та системним середовищем.

Функціональні вимоги:

- ПП повинен мати підсистему аутентифікації
- ПП повинен мати підсистему реєстрації нових користувачів
- ПП повинен бути веб-застосунком
- Доступ до основного функціоналу повинні мати лише аутентифіковані користувачі
- Дані користувача повинні бути в безпеці та не повинні передаватись стороннім особам
- Повинна бути можливість відображати виконання алгоритмів покроково
- Повинна бути можливість створювати випадкові датасети для виконання алгоритмів
- Повинна бути можливість вводити власні датасети
- Може бути можливість завантажувати датасети з локальних файлів
- Повинна бути можливість зберігати створені датасети
- Повинна бути можливість вибирати датасети зі списку збережених
- Може бути можливість редагувати алгоритми
- Може бути можливість створювати власні алгоритми

Нефункціональні вимоги:

- Затримка між кроками виконання алгоритмів повинна бути незмінною і дорівнювати 0.2 секунди.

- Взаємодія з конкретним алгоритмом не повинна виконуватися з використанням більше однієї веб сторінки
- Застосунок повинен бути розрахований на подальше розширення функціональності (відповідно має бути організована структура коду)
- Може бути передбачена система резервного копіювання даних користувачів веб застосунку адміністратором та відновлення їх в разі проблем з системою

2.2. Постановка задачі

Задача ПП є надання навчального інструменту людям які починають вивчати комп’ютерні науки або викладачам яким потрібен інструмент візуалізації принципу роботи алгоритмів сортування та пошуку. За допомогою комп'ютерної системи подібні задачі можна автоматизувати, що і є ціллю даного веб застосунку.

2.3. Моделювання інформаційної системи/підсистеми

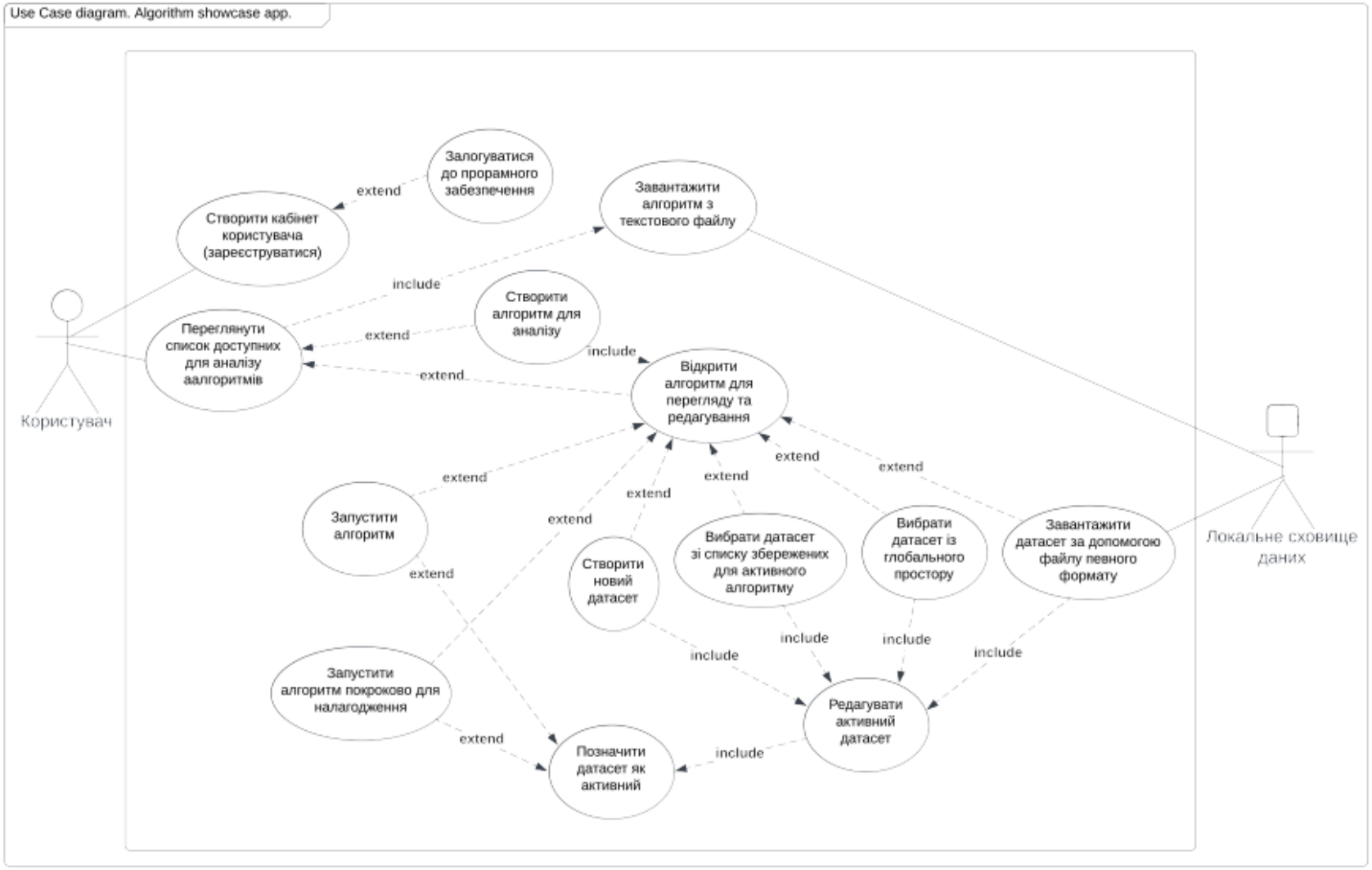


Рис. 2.3.1 діаграма прецедентів застосунку.

ПП реалізоване в формі веб застосунку. Застосунок завантажується динамічно.

Для використання застосунку необхідно бути зареєстрованим та аутентифікованим. Для реєстрації система вимагає безпечний пароль та унікальну, не використану в системі поштову адресу. Для завершення реєстрації необхідно підтвердити валідність електронної адреси перейшовши за посиланням висланим на вказану при реєстрації електронну пошту. Після завершення реєстрації користувач може аутентифікуватися і використовувати весь функціонал веб застосунку.

Користувач має доступ до наступного функціоналу після аутентифікації:

- переглядати список стандартних алгоритмів
- створювати нові алгоритми
- редагувати алгоритми сортування
- порівнювати виконання алгоритмів зі списку
- виконувати алгоритми на випадкових наборах даних
- виконувати алгоритми на власних наборах даних

Рис. 2.3.2 Діаграма послідовності “Реєстрація”

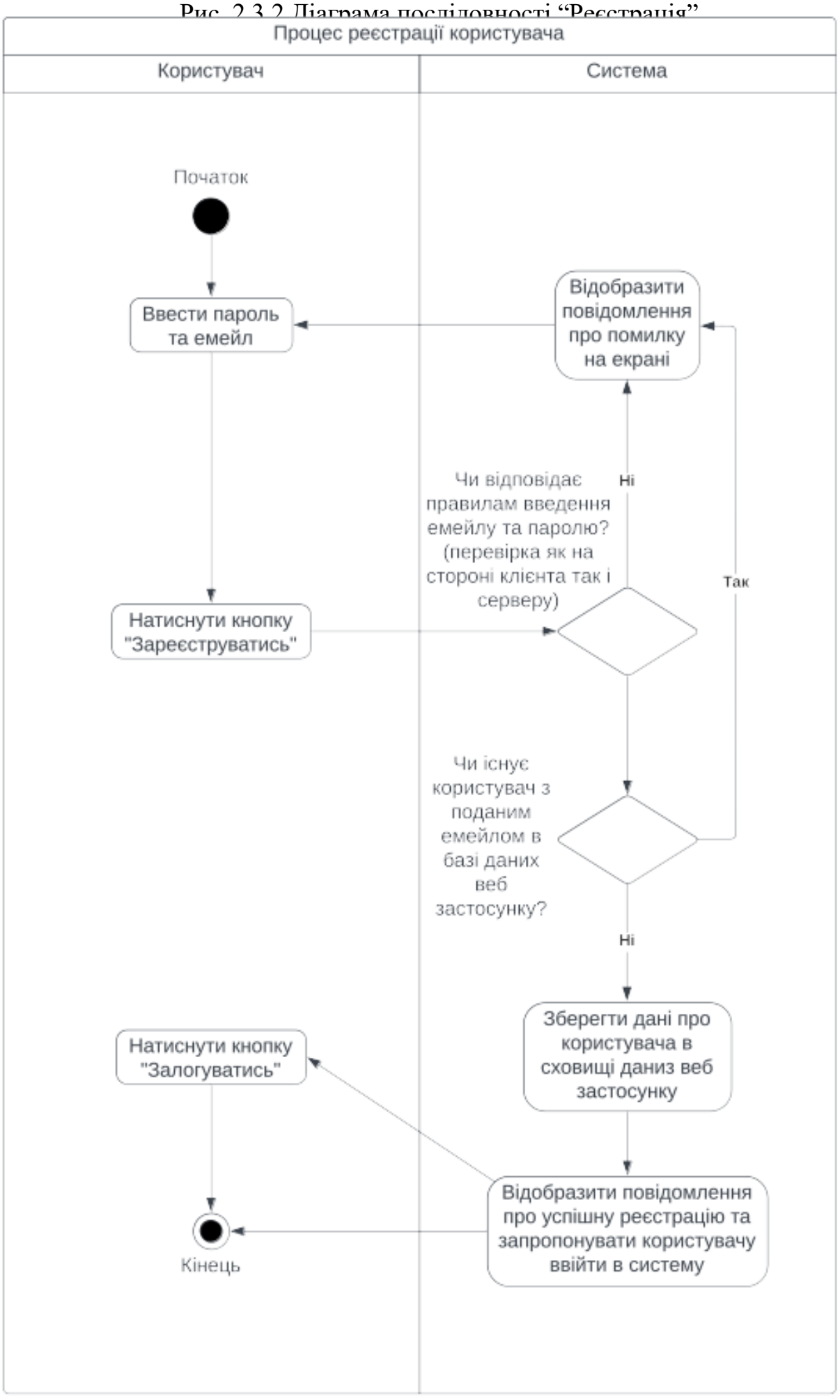
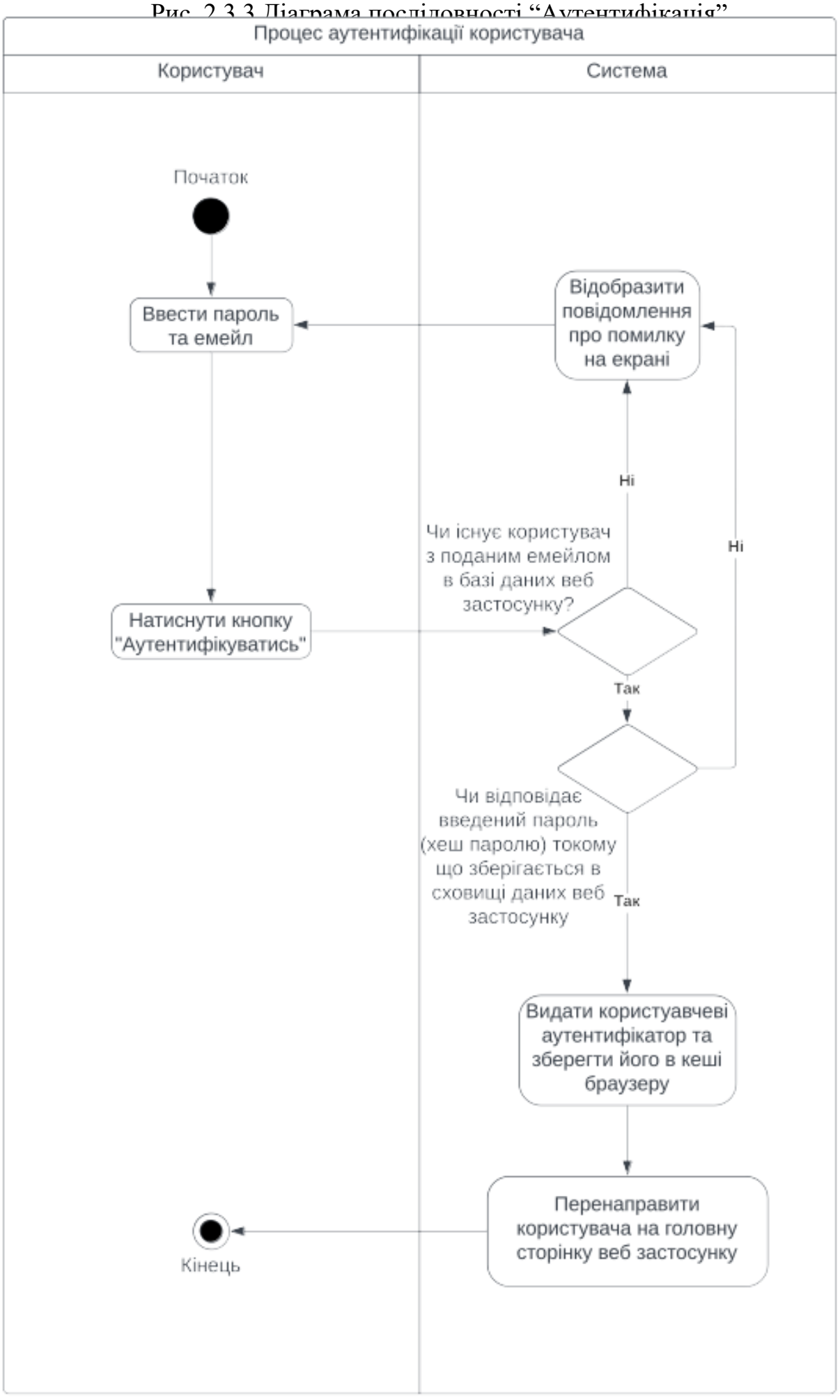


Рис. 2.3.3 Діаграма послідовності “Аутентифікація”



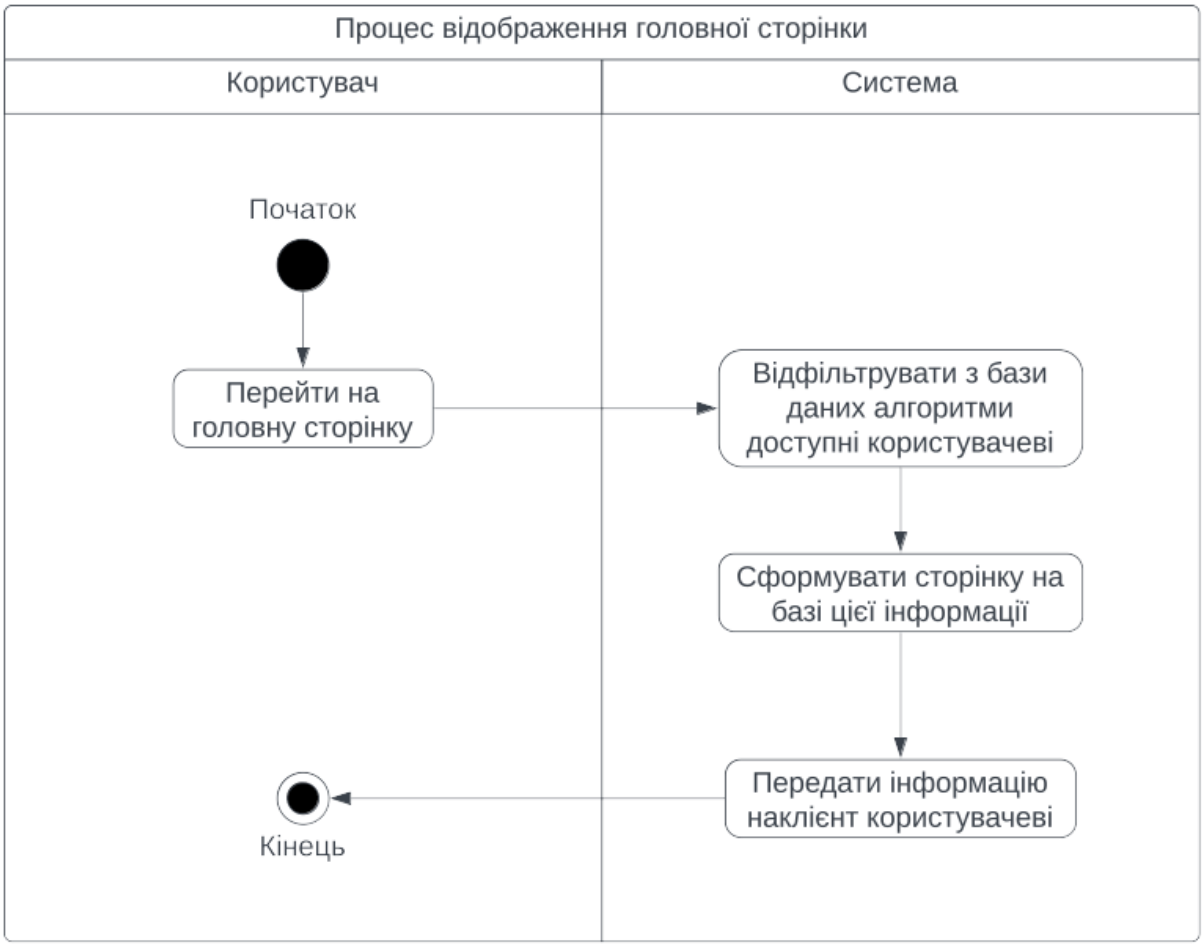


Рис 2.3.4 Діаграма послідовності “Відображення головної сторінки”

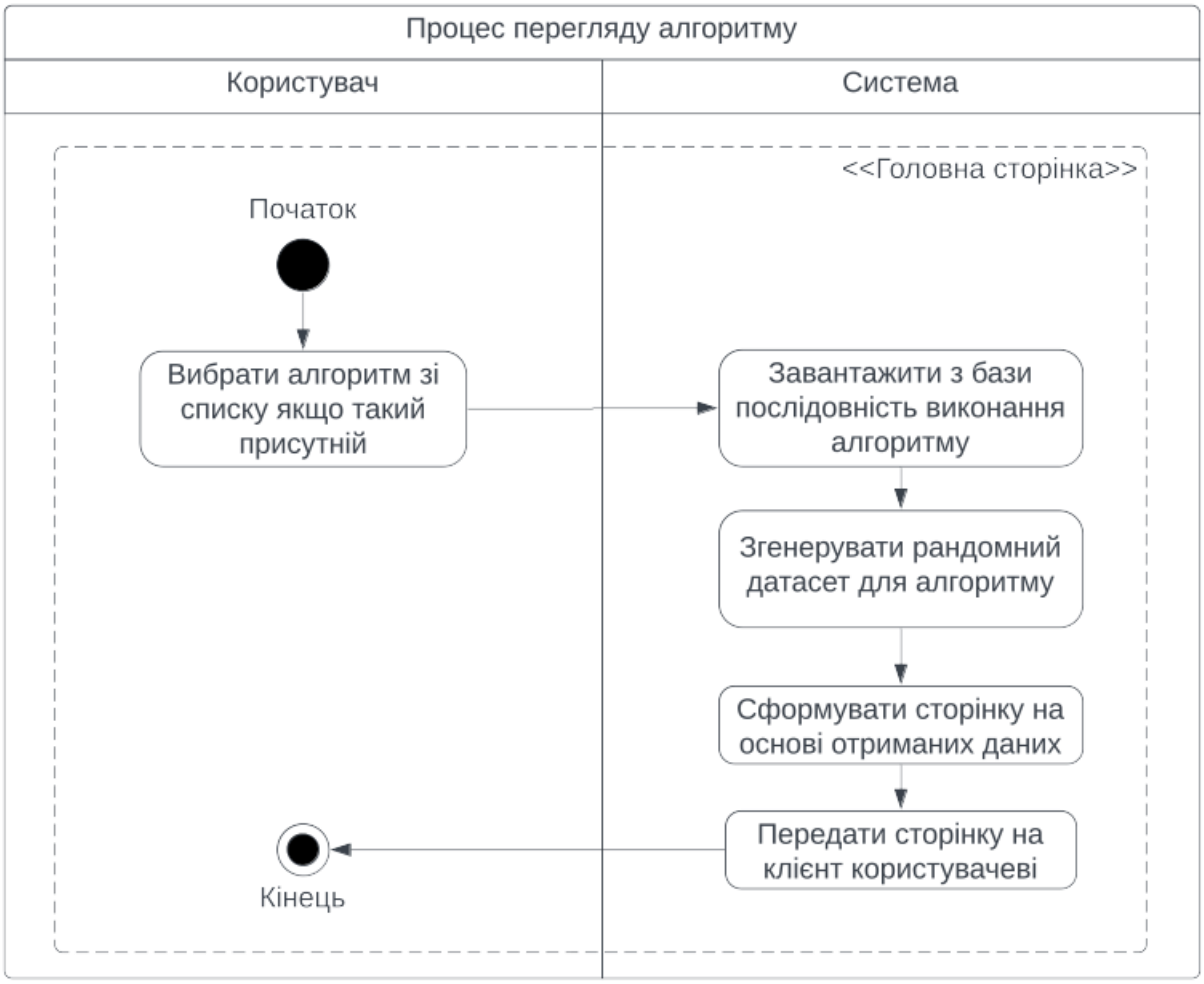


Рис 2.3.5 Діаграма послідовності “Перегляд алгоритму”

РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ/ПІДСИСТЕМИ

3.1. Інформаційне забезпечення

Django — це високорівневий веб-фреймворк Python, який заохочує швидку розробку та чистий, прагматичний дизайн. Однією з визначальних особливостей Django є акцент на модульності, що відображається на структурі програм Django.

Проект Django — це набір однієї або кількох програм Django, які разом створюють веб-програму. Програма Django — це пакет Python, який містить увесь код для певної частини функціональності в проекті. Наприклад, у вас може бути програма Django для обробки автентифікації користувачів, інша для керування блогом і інша для обробки замовлень в онлайн-магазині.

На найвищому рівні проект Django організований у кілька каталогів і файлів, як показано нижче:

```
project_name/
├── manage.py
├── project_name/
│   ├── __init__.py
│   ├── asgi.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── app1/
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── migrations/
│   ├── models.py
│   ├── tests.py
│   └── views.py
├── app2/
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── migrations/
│   ├── models.py
│   ├── tests.py
│   └── views.py
└── ...
```

Давайте детальніше розглянемо кожен із цих каталогів і файлів:

manage.py

manage.py — це утиліта командного рядка, яка дозволяє вам взаємодіяти з вашим проектом Django. Ви можете використовувати його для запуску команд керування Django, таких як запуск сервера розробки, створення нових програм і застосування міграції бази даних.

project_name/

Це кореневий каталог вашого проекту Django, і він містить основні файли конфігурації проекту.

__init__.py — це порожній файл Python, який повідомляє Python, що цей каталог слід розглядати як пакет Python.

asgi.py — це точка входу для ASGI-сумісних веб-серверів для обслуговування вашого проекту. ASGI (Asynchronous Server Gateway Interface)

— це специфікація для створення асинхронних веб-додатків і є альтернативою WSGI (Web Server Gateway Interface).

settings.py — це основний файл конфігурації вашого проекту Django. Він містить налаштування для конфігурації бази даних, встановлених програм, проміжного ПЗ тощо.

urls.py — це модуль Python, який визначає шаблони URL для вашого проекту. Ці шаблони відображають URL-адреси у представленнях, які є функціями Python, які обробляють запити та повертають відповіді.

wsgi.py — це точка входу для WSGI-сумісних веб-серверів для обслуговування вашого проекту. WSGI — це специфікація для створення веб-додатків, і це найпоширеніший спосіб розгортання додатків Django у виробництві.

app1/, app2/, ...

Це каталоги для ваших програм Django. Кожна програма має власний каталог, і назва каталогу є назвою програми.

__init__.py — це порожній файл Python, який повідомляє Python, що цей каталог слід розглядати як пакет Python.

admin.py — це модуль Python, який визначає настроюваний інтерфейс адміністрування для вашої програми. Django має вбудований інтерфейс адміністратора, який дозволяє вам керувати своїми даними через веб-інтерфейс, і ви можете налаштувати поведінку інтерфейсу адміністратора, визначивши власні моделі, поля та дії в цьому файлі.

apps.py — це модуль Python, який визначає конфігурацію програми Django. Цей файл використовується для визначення назви та мітки програми, а також будь-яких настроюваних дій або налаштувань, які ви хочете визначити для програми.

migrations/ — це каталог, який містить міграції бази даних Django. Міграції — це спосіб Django перенести зміни, які ви вносите у свої моделі (додавання поля, видалення моделі тощо), у вашу схему бази даних. Вони створені здебільшого автоматично, але вам потрібно знати про них, коли ви робите такі дії, як створення моделей і додавання полів.

models.py — це модуль Python, який визначає моделі Django. Модель — це клас, який представляє таблицю в базі даних і визначає поля та поведінку даних, які потрібно зберегти.

tests.py — це модуль Python, який містить тестові функції для вашої програми. Django містить вбудовану програму тестування, яка дозволяє перевірити ваш код і переконатися, що він працює правильно.

views.py — це модуль Python, який визначає функції перегляду для вашої програми. Представлення — це функція Python, яка приймає запит і повертає відповідь, і відповідає за обробку логіки вашої програми.

Окрім цих основних каталогів і файлів, ви також можете мати додаткові файли та каталоги залежно від потреб вашого проекту. Наприклад, у вас може бути каталог static/ для зберігання статичних ресурсів, таких як файли CSS і JavaScript, каталог templates/ для зберігання шаблонів HTML і каталог media/ для зберігання завантажених користувачами медіафайлів.

Окремим компонентом конкретної імплементації є вкладений файл формату sqlite який слугує до зберігання бази даних на етапі розробки та відлагодження ПП. Під час кінцевого розгорнення рекомендується мати окремий сервер бази даних на базі повноцінної реляційної бази даних. Sqlite має на ціль зберігати дані як умога компактніше і не розрахована на подальше розширення, цілісність даних в різних умовах не є основною її ціллю.

Підключити базу даних до програмного продукту написаного на Django можна прописавши конфігурацію підключення в файлі settings.py наступним чином:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'mydatabase',
        'USER': 'myuser',
        'PASSWORD': 'mypassword',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

В випадку з рушієм sqlite це виглядає наступним чином:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

Опис масивів даних

Найменування масиву — Авторизований користувач
Ідентифікатор масиву — auth_user
Найменування носія інформації — datebase
Ключі упорядкування — id
Ідентифікатор індексного масиву — auth_user

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний/вторинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
ID	id	id	integer	PK	AUTOINCREMENT	Yes	Yes	
Password	password	password	varchar(128)			Yes		
Last login	last_login	last_login	datetime					
Is superuser	is_superuser	is_superuser	bool			Yes		
Username	username	username	varchar(150)		UNIQUE	Yes		
Last name	last_name	last_name	varchar(150)			Yes		
Email	email	email	varchar(254)			Yes		
Is stuff	is_stuff	is_stuff	bool			Yes		
Is active	is_active	is_active	bool			Yes		
Date joined	date_joined	date_joined	datetime			Yes		
First name	first_name	first_name	varchar(150)			Yes		

Найменування масиву — Користувацька група
Ідентифікатор масиву — auth_group
Найменування носія інформації — datebase
Ключі упорядкування — id
Ідентифікатор індексного масиву — auth_group

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний/вторинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
ID	id	id	integer	PK	AUTOINCREMENT	Yes	Yes	

Name	name	name	varchar(150)		UNIQUE			

Найменування масиву — Дозволи-групи(поєднувальна таблиця)
Ідентифікатор масиву — auth_group_permission
Найменування носія інформації — datebase
Ключі упорядкування — id
Ідентифікатор індексного масиву — auth_group_permission

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний/вторинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
ID	id	id	integer	PK	AUTOINCREMENT	Yes	Yes	
Password	group_id	group_id	integer	FK		Yes		auth_group (id)
Last login	permission_id	permission_id	integer	FK		Yes		auth_permission (id)

Найменування масиву — Дозволи
Ідентифікатор масиву — auth_permission
Найменування носія інформації — datebase
Ключі упорядкування — id
Ідентифікатор індексного масиву — auth_permission

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний/вторинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
ID	id	id	integer	PK	AUTOINCREMENT	Yes	Yes	
Content Type ID	content_type_id	content_type_id	integer	FK		Yes		content_type (id)
Code Name	codename	codename	varchar (100)			Yes		
Name	name	name	varchar (255)			Yes		

Найменування масиву — Користувачі-групи(поєднувальна таблиця)
Ідентифікатор масиву — auth_user_groups
Найменування носія інформації — datebase
Ключі упорядкування — id
Ідентифікатор індексного масиву — auth_user_groups

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний/вторинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
ID	id	id	integer	PK	AUTOINCREMENT	Yes	Yes	
User ID	user_id	user_id	integer	FK		Yes		auth_user (id)
Group ID	group_id	group_id	integer	FK		Yes		auth_group (id)

Найменування масиву — Користувачі-дозволи(поєднувальна таблиця)
Ідентифікатор масиву — auth_user_user_permissions
Найменування носія інформації — datebase
Ключі упорядкування — id
Ідентифікатор індексного масиву — auth_user_user_permissions

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний/вторинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
ID	id	id	integer	PK	AUTOINCREMENT	Yes	Yes	
User ID	user_id	user_id	integer	FK		Yes		auth_user (id)
Permission ID	permission_id	permission_id	integer	FK		Yes		auth_permission (id)

Найменування масиву — Тип наповнення
Ідентифікатор масиву — content_type
Найменування носія інформації — datebase
Ключі упорядкування — id
Ідентифікатор індексного масиву — content_type

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний/вторинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
ID	id	id	integer	PK	AUTOINCREMENT	Yes	Yes	
App Label	app_label	app_label	varchar (100)			Yes		
Model	model	model	varchar (100)			Yes		

Найменування масиву — Дозволи
Ідентифікатор масиву — content_type
Найменування носія інформації — datebase
Ключі упорядкування — id
Ідентифікатор індексного масиву — auth_permission

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний/вторинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
ID	id	id	integer	PK	AUTOINCREMENT	Yes	Yes	
App Label	app_label	app_label	varchar (100)			Yes		
Model	model	model	varchar (100)			Yes		

3.2. Технічне забезпечення

Для розробки ПП було використано ноутбук марки ASUS.

При розгортанні програми рекомендується використовувати окрему машину або деплоймент як ізольований контейнер чи віртуальна машина.

Для хостингу програмного забезпечення в рамках тестування та демонстрації може використовуватись будь-який комп’ютер зі стабільним підключенням до інтернету операційна система якого має підтримку інтерпретатора Python. Мінімальні рекомендовані характеристики комп’ютеру який виконує функцію серверу-хосту є такими: 4 ядра процесору, 4 гігабайти оперативної пам’яті, 120 гігабайт внутрішнього сховища даних, підключення до мережі в 100 мегабіт. Таких характеристик має бути достатньо для невеликої користувацької бази (до 1000 користувачів)

Також однім із варіантів розгортання сервер ПП запущеним на сервісі віддаленого хостингу з подібними характеристики.

Для зберігання користувацьких даних повинна використовуватися окрема система в випадку повномасштабного розгортання веб застосунку. Ця система повинна бути дуплікована всередині локальної мережі і має бути створена резервна копія в хмарі яка має бути регулярно оновлена і обслуговувана.

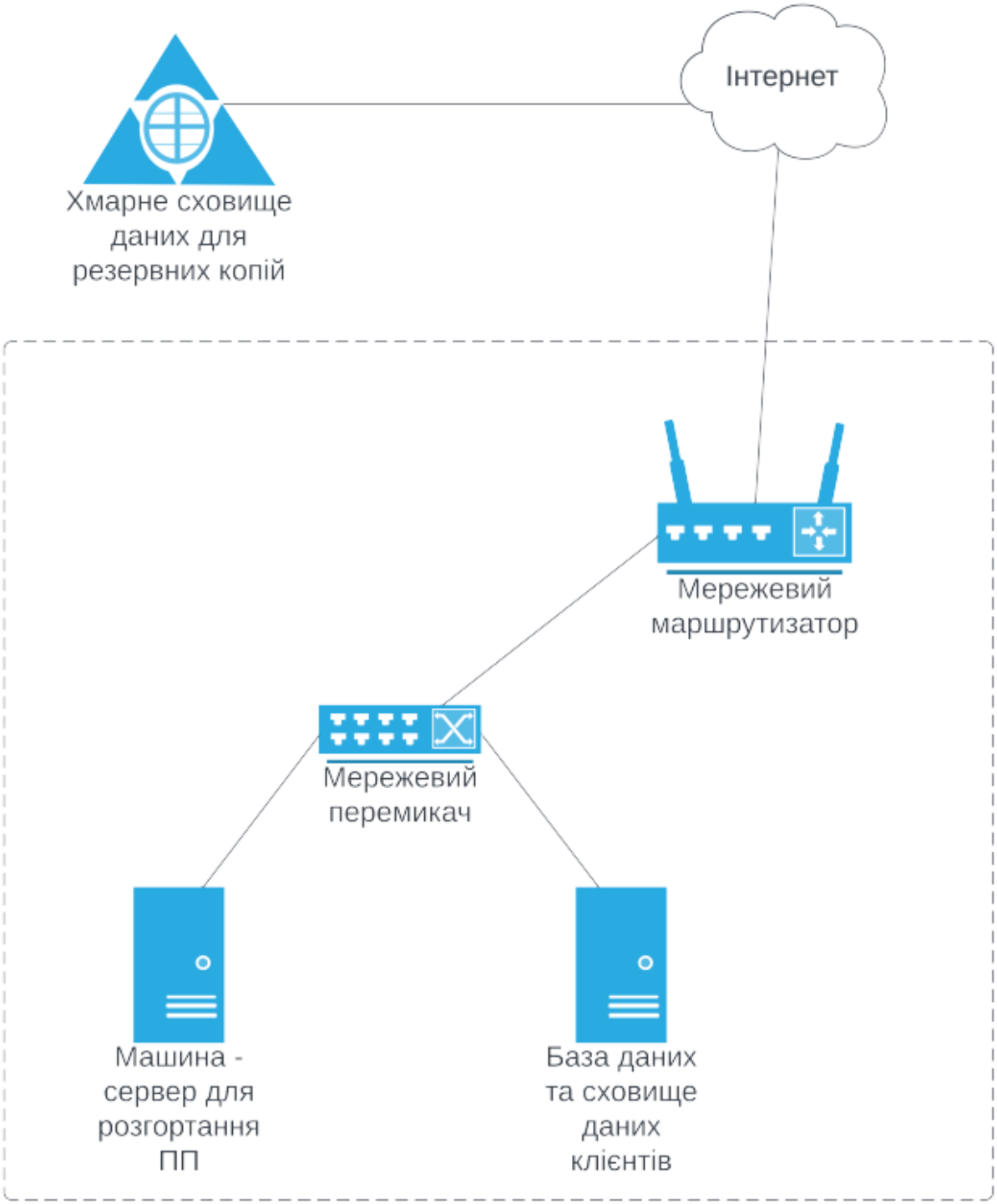


Рисунок 3.2.1 — Діаграма мережевої інфраструктури для розгорнення веб застосунку

На діаграмі вище можна побачити мережеву діаграму системи рекомендованої для локального хостингу веб застосунку розробленого в ході виконання ДП.

3.3. Програмне забезпечення

Важливо правильно підібрати правильний набір інструментів для дизайну, архітектури, розробки та документації ПП. Від цього залежить на скільки зручно та ефективно буде прогресувати в розробці ПП. Також від підбору інструментів залежить подальше вдосконалення, обслуговування та адміністрування ПП.

Структура програмного забезпечення

Структура моєї веб-програми Django включає кілька компонентів, які працюють разом, щоб забезпечити бажану функціональність.

По-перше, моделі, визначені у файлі models.py, представляють структуру бази даних і використовуються для взаємодії з базою даних через ORM Django. Ці моделі є класами Python, які створюють підклас django.db.models.Model і визначають поля, які представляють стовпці в таблицях бази даних.

Далі файл views.py містить функції, які обробляють HTTP-запити та шаблони візуалізації. Ці функції перегляду відповідають за виконання будь-якої необхідної логіки та рендеринг відповідного шаблону у відповідь на запит.

Файл urls.py визначає шаблони URL-адрес, які відображають URL-адреси функціям перегляду. Ці шаблони дозволяють користувачам отримувати доступ до різних частин програми, переходячи за певними URL-адресами у своїх веб-переглядачах.

Шаблони в каталозі шаблонів визначають структуру та макет веб-сторінок і можуть включати динамічний вміст та анімацію за допомогою мови шаблонів Django та JavaScript або CSS.

Нарешті, сервер розробки та веб-сервер (наприклад, Apache або Nginx) використовуються для тестування та розгортання програми відповідно.

Підсумовуючи, структура моєї веб-програми Django складається з моделей бази даних, функцій перегляду для обробки запитів HTTP, шаблонів URL-адрес для відображення URL-адрес у переглядах, шаблонів для визначення структури та макета веб-сторінок і серверів для тестування та розгортання додаток.

Системне програмне забезпечення

Як операційна система де рекомендується розгортати веб-застосунок може використовуватися будь-який Linux дистрибутив який на поточний момент отримує безпекові оновлення. Для розробки, тестування та пробного розгортання системи використано дистрибутив Fedora Linux 37.

Як RAD засіб використано онлайн сервіс Lucid Charts який дає велику варіацію інструментів для будь-якого типу діаграм які були необхідно створити в ході виконання дипломного проекту. Безкоштовна версія надає доступ до більшості функціоналу з обмеженням на кількість діаграм доступних для редагування одночасно (останні 3 створені діаграми є можливим редагувати, всі інші є доступними для перегляду/експорту).

Для розробки ПП в якості мов програмування використовувались Python для написання серверу програми, JavaScript та мова розмітки HTML. Фреймворк Django був використаний для зв’язування компонентів розроблених з використанням вище вказаних мов.

Прикладне програмне забезпечення

В якості редактора коду було використано засіб для створення, редагування та налагоджування програм на різних мовах програмування Visual Studio Code (vscode). Даний застосунок є дуже гнучким. Vscode може бути встановлений на будь-який сучасний персональний комп’ютер так як є написаним на мові JavaScript і по суті є локальним веб застосунком завдяки чому може бути портований на будь-яку платформу без зайвих проблем. Цей редактор має дуже велике поширення серед розробників-професіоналів і завдяки відкритості коду та зручному API інтерфейсу має тисячі розширень розроблених ентузіастами та компаніями-розробниками які дозволяють організувати розробку ПП практично на будь-якій з існуючих популярної мов

програмування. Прямо в редактор є інтегровано каталог цих розширень з можливості їх встановлення та налаштування. Є можливість створювати налаштування редакторів, інтерпретаторів, налагоджувачів різних мов програмування, як глобально на робочій станції так і індивідуально для кожного проекту. Також для окремих проектів є можливість налаштувати багато унікальних конфігурацій для автоматизованого запуску програми. Можна підключатись до віддалених робочих станцій за допомогою протоколу SSH прямо в редакторі коду і зручно адмініструвати наприклад віддалений сервер де є розгорнуто веб застосунок. Редактор vscode можна порівняти зі швейцарським ножем в світі інструментів розробки ПЗ за допомогою якого можна організувати процес розробки з будь-якою можливою комбінацією мовних та фреймворкових середовищ.

Програмна документація

Програма призначена для використання студентами та викладачами для вивчення та аналізу алгоритмів. Також розробники можуть проаналізувати свої алгоритми на працездатність та порівняти їх до існуючих рішень.

3.4. Результати реалізації інформаційної системи/підсистеми

Як розробник, перше, що я зробив, це спланував та зібрав вимоги до програми. Це важливий крок у процесі розробки, оскільки він допомагає гарантувати, що програма відповідає потребам і очікуванням користувачів та інших зацікавлених сторін. Щоб зібрати вимоги, я провів дослідження користувачів, зібрав інформацію від зацікавлених сторін і створив дорожню карту проекту, яка описує функції та функції програми.

Після збору вимог наступним кроком було налаштування середовища розробки. Це передбачає встановлення Django та будь-яких інших необхідних залежностей, таких як система керування базами даних і будь-які сторонні бібліотеки чи фреймворки. Налаштування середовища розробки також передбачає налаштування сервера розробки, який є локальним сервером, який я можу використовувати для запуску та тестування програми під час процесу розробки.

Після налаштування середовища розробки я перейшов до розробки моделі програми. Модель є частиною програми Django, яка визначає структури даних і зв’язки, які будуть використовуватися для зберігання та керування даними програми. Це передбачає створення моделей Django, які є класами Python, які визначають поля та поведінку даних, які програма зберігатиме. Наприклад, якщо програма є простим блогом, модель може містити модель публікації, яка визначає поля для заголовка, основної частини та дати кожної публікації блогу.

Після розробки моделі я перейшов до створення представлень і шаблонів для програми. Представлення — це частина програми Django, яка обробляє запити та відповіді HTTP, а шаблони — це файли HTML, які визначають структуру та макет інтерфейсу користувача. Для створення представлень і шаблонів я спочатку розробив загальний макет і структуру інтерфейсу користувача, а потім створив представлення Django для обробки різних типів

запитів HTTP, які отримуватиме програма. Наприклад, якщо програма є простим блогом, я міг би створити подання для відображення списку всіх публікацій блогу та інше подання для відображення деталей однієї публікації блогу.

Коли представлення та шаблони були готові, я перейшов до тестування та налагодження програми. Тестування та налагодження є важливим кроком у процесі розробки, оскільки це допомагає переконатися, що програма функціонує належним чином і не містить помилок. Щоб перевірити програму, я запустив її та вручну перевірив, чи всі функції та функції працюють належним чином. Якщо я стикався з будь-якими проблемами, я використовував вбудовану систему тестування Django, щоб визначити джерело проблеми та виправити її.

Нарешті, коли додаток було повністю протестовано та налагоджено, я розгорнув його на робочому сервері. Розгортання передбачає налаштування робочого сервера, встановлення необхідних залежностей і розгортання коду та будь-яких необхідних активів на сервері. Я також налаштував базу даних та іншу інфраструктуру для підтримки програми у виробництві.

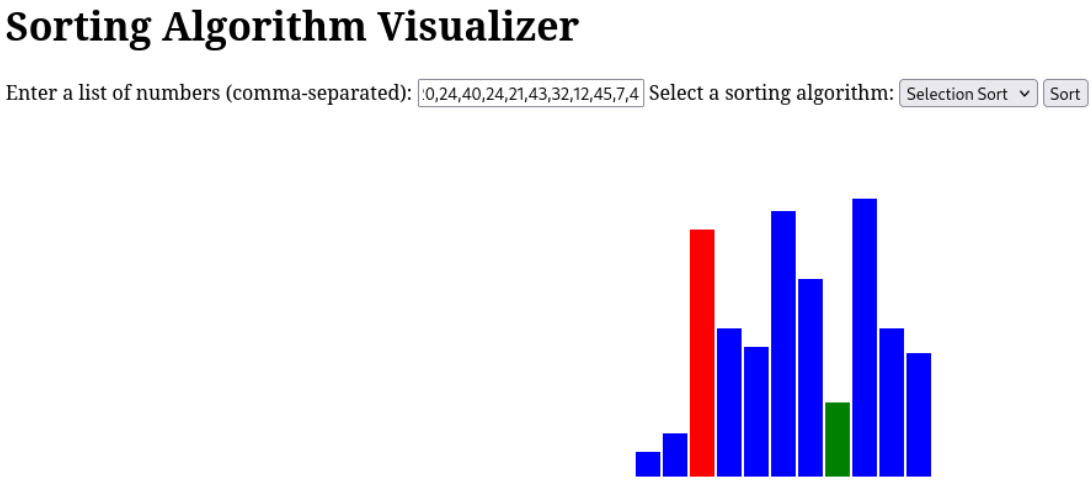


Рисунок 3.4.1 — Реалізація проєкту

Приклад логу з серверу:

```
[20/Jun/2024 18:23:33] "GET /sorting_algorithm/ HTTP/1.1" 200 1901
[20/Jun/2024 18:23:38] "POST /sorting_algorithm/sort/ HTTP/1.1" 200 1064
[20/Jun/2024 18:23:46] "GET /sorting_algorithm/ HTTP/1.1" 200 1901
[20/Jun/2024 18:23:50] "GET /sorting_algorithm/ HTTP/1.1" 200 1901
[20/Jun/2024 18:26:55] "GET /sorting_algorithm/ HTTP/1.1" 200 1962
[20/Jun/2024 18:26:55] "GET /static/sorting_algorithm/sort.js HTTP/1.1" 200 1785
Not Found: /favicon.ico
[20/Jun/2024 18:26:55] "GET /favicon.ico HTTP/1.1" 404 2233
[20/Jun/2024 18:26:58] "POST /sorting_algorithm/sort/ HTTP/1.1" 200 1064
[20/Jun/2024 18:27:05] "GET /sorting_algorithm/ HTTP/1.1" 200 1962
Not Found: /favicon.ico
[20/Jun/2024 18:27:05] "GET /favicon.ico HTTP/1.1" 404 2233
[20/Jun/2024 18:27:07] "POST /sorting_algorithm/sort/ HTTP/1.1" 200 1064
[20/Jun/2024 18:27:34] "POST /sorting_algorithm/sort/ HTTP/1.1" 200 374
[20/Jun/2024 18:27:41] "POST /sorting_algorithm/sort/ HTTP/1.1" 200 131
[20/Jun/2024 18:27:57] "POST /sorting_algorithm/sort/ HTTP/1.1" 200 200
[20/Jun/2024 18:28:02] "POST /sorting_algorithm/sort/ HTTP/1.1" 200 3401
[20/Jun/2024 18:28:43] "GET /sorting_algorithm/ HTTP/1.1" 200 1962
[20/Jun/2024 18:28:43] "GET /static/sorting_algorithm/sort.js HTTP/1.1" 200 1785
[20/Jun/2024 18:28:57] "POST /sorting_algorithm/sort/ HTTP/1.1" 200 3401
[20/Jun/2024 18:29:17] "POST /sorting_algorithm/sort/ HTTP/1.1" 200 3401
```

Висновок

Будучи студентом інформатики, я був зацікавлений у розробці інструменту, який би полегшив студентам і професіоналам розуміння та візуалізацію алгоритмів сортування в рамках мого випускного бакалаврського проекту. Я визнав, що ці алгоритми є фундаментальним аспектом інформатики та мають численні застосування в різних галузях, але зрозуміти та порівняти їх продуктивність може бути складно.

Щоб вирішити цю проблему, я вирішив створити веб-додаток, призначений для інтерактивної візуалізації різних алгоритмів сортування. Моя програма дозволяє користувачам вибирати з низки алгоритмів, вводити власні дані, бачити, як працюють алгоритми, і порівнювати їх ефективність.

Розробляючи додаток, я зосередився на зручності та простоті. Я хотів створити інструмент, який буде простим у використанні та зрозумілим навіть для тих, хто має невеликий або зовсім не має досвіду роботи з алгоритмами сортування. Щоб досягти цього, я надав чіткі та стислі пояснення кожного алгоритму, а також інтерактивні візуалізації, які показують етапи процесу сортування в реальному часі.

Додаток містить низку алгоритмів, у тому числі такі популярні, як швидке сортування, сортування злиттям і групове сортування, а також менш відомі алгоритми, такі як сортування за принципом і сортування по блоку. Для кожного алгоритму я надав детальне пояснення базових принципів і принципів роботи алгоритму, а також приклади коду різними мовами програмування.

Користувачі можуть вибрати алгоритм і ввести власні дані, які можуть бути у формі списку чисел, рядків або інших типів даних. Потім програма візуалізує процес сортування в режимі реального часу, показуючи кроки, які виконує алгоритм, і виділяючи ключові елементи процесу.

Однією з ключових особливостей програми є можливість порівнювати продуктивність різних алгоритмів. Користувачі можуть вибрати кілька алгоритмів і ввести однакові дані, а програма покаже результати пліч-о-пліч, дозволяючи користувачам побачити, як працюють алгоритми за однакових умов. Це важлива функція для розуміння компромісів між різними алгоритмами та допомагає користувачам вибрати правильний алгоритм для конкретного завдання.

Окрім візуалізацій, програма містить низку інших ресурсів та інструментів для вивчення та розуміння алгоритмів сортування. До них входять вправи та тести для перевірки знань користувачів, а також посилання на зовнішні ресурси, такі як статті та відео про певні алгоритми.

Розробка програми була складним, але вдячним процесом. Однією з головних проблем, з якою я зіткнувся, було знайти правильний баланс між простотою та вичерпністю. Я хотів надати достатньо деталей і пояснень, щоб зробити алгоритми зрозумілими, але не настільки, щоб додаток став непереборним або заплутаним. Іншою проблемою було забезпечення чіткості та точності візуалізацій, що вимагало ретельного проектування та тестування.

Незважаючи на ці проблеми, я вважаю, що програма є корисним і зручним інструментом для вивчення та розуміння алгоритмів сортування. Він надає інтерактивний та захоплюючий спосіб дізнатися про ці алгоритми та побачити, як вони працюють, а можливість порівняти продуктивність різних алгоритмів є цінним ресурсом як для студентів, так і для професіоналів.

Перелік використаних джерел

1. https://uk.wikipedia.org/wiki/Вимоги_до_програмного_забезпечення
2. https://en.wikipedia.org/wiki/Secure_Shell
3. https://en.wikipedia.org/wiki/Linux
4. https://en.wikipedia.org/wiki/Visual_Studio_Code
5. https://en.wikipedia.org/wiki/HTML
6. https://en.wikipedia.org/wiki/JavaScript
7. https://en.wikipedia.org/wiki/Python_(programming_language)
8. https://uk.wikipedia.org/wiki/Прикладний_програмний_інтерфейс
9. https://en.wikipedia.org/wiki/API
10. https://en.wikipedia.org/wiki/Software_framework
11. https://uk.wikipedia.org/wiki/Програмний_каркас
12. https://ru.wikipedia.org/wiki/Фреймворк
13. https://www.djangoproject.com/
14. https://www.w3schools.com/django/
15. https://www.tutorialspoint.com/django/index.htm
16. https://lucid.app

Структура проекту:

```
sorting_app
├── manage.py
├── sorting_algorithm
│   ├── admin.py
│   ├── apps.py
│   ├── forms.py
│   ├── models.py
│   ├── static
│   │   └── sorting_algorithm
│   │       ├── sort.js
│   │       └── styles.css
│   ├── templates
│   │   └── sorting_algorithm
│   │       ├── base.html
│   │       ├── index.html
│   │       └── sort.html
│   ├── tests.py
│   ├── urls.py
│   └── views.py
└── sorting_app
    ├── asgi.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

sorting_app/manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'sorting_app.settings')
    try:
    from django.core.management import execute_from_command_line
    except ImportError as exc:
    raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
    ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

sorting_app/sorting_algorithm/sorting_algorithm/sort.js

```
console.log('sort.js loaded');
document.addEventListener('DOMContentLoaded', function () {
  const form = document.getElementById('sorting-form');  // Ensure the form has this ID

  if (form) {
     form.addEventListener('submit', function (event) {
     event.preventDefault();  // This prevents the form from submitting normally

     const formData = new FormData(form);
     fetch(form.action, {
     method: 'POST',
     body: formData,
     headers: {
               'X-Requested-With': 'XMLHttpRequest',
               'X-CSRFToken': form.querySelector('[name=csrfmiddlewaretoken]').value
     }
     })
     .then(response => {
     if (!response.ok) {
               throw new Error('Network response was not ok ' + response.statusText);
     }
     return response.json();
     })
     .then(data => {
     console.log(data);
     visualizeSorting(data.steps);
     })
     .catch(error => console.error('Error:', error));
     });
  }
});

function visualizeSorting(steps) {
  const chartContainer = document.getElementById('chart-container');
  chartContainer.innerHTML = '';  // Clear previous bars

  steps.forEach((step, index) => {
     const [numbers, num1, num2] = step;
     setTimeout(() => {
     chartContainer.innerHTML = '';  // Clear existing bars
     numbers.forEach((number, idx) => {
     const bar = document.createElement('div');
     bar.style.width = '20px';
     bar.style.height = `${number * 5}px`;  // Adjust scaling as necessary
     bar.style.margin = '1px';
     bar.style.backgroundColor = (idx === num1 || idx === num2) ? (idx === num1 ? 'red' : 'green') : 'blue';
     bar.style.display = 'inline-block';
     chartContainer.appendChild(bar);
     });
     }, index * 200);  // Delay each step
  });
}
```

sorting_algorithm/urls.py

```
from django.urls import path
from . import views

app_name = 'sorting_algorithm'

urlpatterns = [
    path('', views.index, name='index'),
    path('sort/', views.sort_numbers, name='sort'),
]
```

sorting_algorithm/views.py

```
from django.shortcuts import render
from django.http import JsonResponse
from .forms import SortingForm

def index(request):
    form = SortingForm()
    return render(request, 'sorting_algorithm/index.html', {'form': form})

def sort_numbers(request):
    if request.method == 'POST':
    form = SortingForm(request.POST)
    if form.is_valid():
            numbers = form.cleaned_data['numbers'].split(',')
            algorithm = form.cleaned_data['algorithm']
            numbers = [int(num.strip()) for num in numbers]
            steps = []

            if algorithm == 'selection':
            selection_sort(numbers, steps)
            elif algorithm == 'insertion':
            insertion_sort(numbers, steps)
            elif algorithm == 'bubble':
            bubble_sort(numbers, steps)
            elif algorithm == 'merge':
            merge_sort(numbers, steps)
            elif algorithm == 'quick':
            quick_sort(numbers, steps)

            return JsonResponse({'steps': steps})

    form = SortingForm()
    return render(request, 'sorting_algorithm/index.html', {'form': form})


def selection_sort(numbers, steps):
    n = len(numbers)
    for i in range(n - 1):
    min_index = i
    for j in range(i + 1, n):
            if numbers[j] < numbers[min_index]:
            min_index = j
            steps.append((numbers.copy(), i, min_index))
    numbers[i], numbers[min_index] = numbers[min_index], numbers[i]
    steps.append((numbers.copy(), i, min_index))
    return numbers


def insertion_sort(numbers, steps):
    n = len(numbers)
    for i in range(1, n):
    key = numbers[i]
    j = i - 1
    while j >= 0 and numbers[j] > key:
            numbers[j + 1] = numbers[j]
            j -= 1
            steps.append((numbers.copy(), j + 1, j + 2))
```

```
        numbers[j + 1] = key
        steps.append((numbers.copy(), j + 1, i))
        return numbers


    def bubble_sort(numbers, steps):
        n = len(numbers)
        for i in range(n - 1):
        for j in range(n - i - 1):
                    if numbers[j] > numbers[j + 1]:
                    numbers[j], numbers[j + 1] = numbers[j + 1], numbers[j]
                    steps.append((numbers.copy(), j, j + 1))
        return numbers

    def merge_sort(numbers, steps):
        if len(numbers) <= 1:
        return numbers

        mid = len(numbers) // 2
        left_half = merge_sort(numbers[:mid], steps)
        right_half = merge_sort(numbers[mid:], steps)
        merged = merge(left_half, right_half, steps)
        steps.append((merged, 0, len(merged) - 1))
        return merged

    def merge(left_half, right_half, steps):
        merged = []
        left_index = 0
        right_index = 0

        while left_index < len(left_half) and right_index < len(right_half):
        if left_half[left_index] < right_half[right_index]:
                    merged.append(left_half[left_index])
                    left_index += 1
        else:
                    merged.append(right_half[right_index])
                    right_index += 1
        steps.append((merged.copy(), left_index, right_index))

        merged.extend(left_half[left_index:])
        merged.extend(right_half[right_index:])
        return merged

    def quick_sort(numbers, steps):
        if len(numbers) <= 1:
        return numbers

        pivot = numbers[0]
        less_than_pivot = [num for num in numbers[1:] if num <= pivot]
        greater_than_pivot = [num for num in numbers[1:] if num > pivot]

        sorted_less = quick_sort(less_than_pivot, steps)
        sorted_greater = quick_sort(greater_than_pivot, steps)
        sorted_list = sorted_less + [pivot] + sorted_greater
        steps.append((sorted_list, 0, len(sorted_list) - 1))
        return sorted_list
```

sorting_app/urls.py

```
"""
URL configuration for sorting_app project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('sorting_algorithm/', include('sorting_algorithm.urls', namespace='sorting_algorithm')),
    ]
```

sorting_algorithm/forms.py

```
from django import forms

SORTING_ALGORITHMS = (
        ('selection', 'Selection Sort'),
        ('insertion', 'Insertion Sort'),
        ('bubble', 'Bubble Sort'),
        ('merge', 'Merge Sort'),
        ('quick', 'Quick Sort'),
)

class SortingForm(forms.Form):
        numbers = forms.CharField(label='Enter a list of numbers (comma-separated)',
                widget=forms.TextInput(attrs={'class': 'form-control'}))
        algorithm = forms.ChoiceField(label='Select a sorting algorithm', choices=SORTING_ALGORITHMS,
                widget=forms.Select(attrs={'class': 'form-control'}))
```