## **ДОДАТОК Б**

Специфікація ПЗ

**CS330 Software Engineering**

**Software Requirements Specification (SRS) Template**

Items that are intended to stay in as part of your document are in **bold**; explanatory comments are in *italic* text. Plain text is used where you might insert wording about your project.

The document in this file is an annotated outline for specifying software requirements, adapted from the IEEE Guide to Software Requirements Specifications (Std 830-1993).

Tailor this to your needs, removing explanatory comments as you go along. Where you decide to omit a section, keep the header, but insert a comment saying why you omit the data.

Software System for Styling and Smart Use of Clothes

SmartStyle Software Engineer

Artem Filatov

**Software Requirements Specification**

**Document**

**Version: 1.0** **Date: 20.06.2025**

**Зміст**

**1. ВСТУП**

1.1 Огляд продукту

1.2 Мета

1.3 Межі

1.4 Посилання

1.5 Означення та абревіатури

**2. ЗАГАЛЬНИЙ ОПИС**

2.1 Перспективи продукту

2.2 Функції продукту

2.3 Характеристики користувачів

2.4 Загальні обмеження

2.5 Припущення й залежності

**3. КОНКРЕТНІ ВИМОГИ**

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

3.1.2 Апаратний інтерфейс

3.1.3 Програмний інтерфейс

3.1.4 Комунікаційний протокол

3.1.5 Обмеження пам’яті

3.1.6 Операції

3.1.7 Функції продукту

3.1.8 Припущення й залежності

3.2 Властивості програмного продукту

3.3 Атрибути програмного продукту

3.3.1 Надійність

3.3.2 Доступність

3.3.3 Безпека

3.3.4 Супроводжуваність

3.3.5 Переносимість

3.3.6 Продуктивність

3.4 Вимоги бази даних

3.5 Інші вимоги

1 ВСТУП

1.1 Огляд продукту

"SmartStyle" – це інноваційний веб-додаток, розроблений для допомоги користувачам в управлінні їхнім гардеробом, отриманні персоналізованих рекомендацій щодо одягу та створенні образів на основі погодних умов та особистих уподобань. Продукт поєднує функції віртуальної шафи, погодного дашборду та рекомендацій на базі штучного інтелекту, щоб спростити щоденний вибір одягу та оптимізувати використання наявного гардеробу.

1.2 Мета

Основною метою проєкту "SmartStyle" є надання користувачам інтуїтивно зрозумілого та функціонального інструменту для:

* ефективного каталогізування власного одягу;
* отримання розумних рекомендацій щодо образів, адаптованих до поточної та прогнозованої погоди;
* збереження та керування улюбленими наборами одягу (пресетами);
* планування гардеробу для конкретних подій або періодів часу;

1.3 Межі

Проєкт "SmartStyle" зосереджений на розробці веб-додатку з наступними ключовими функціями:

* реєстрація та автентифікація користувачів;
* керування персональним профілем користувача;
* додавання, перегляд та видалення предметів одягу у віртуальному гардеробі;
* система тегування одягу (категорії, випадки, локації, активності), прив'язана до користувача;
* відображення поточної погоди та погодинного прогнозу для вибраних міст;
* генерація рекомендацій щодо образів за допомогою AI, враховуючи погоду, місце призначення, тип активності та стиль;
* можливість збереження створених образів як пресетів;
* багатомовна підтримка інтерфейсу (українська, англійська);
* поза межами проєкту залишаються;
* інтеграція з електронною комерцією або онлайн-магазинами;
* функції соціальних мереж (наприклад, обмін образами з іншими користувачами);
* розширені можливості редагування зображень одягу всередині програми;
* розробка нативних мобільних додатків (фокус на адаптивному веб-дизайні);

1.4 Посилання

* кодова база проєкту "SmartStyle" (локальний репозиторій);
* документація до зовнішніх програмних інтерфейсів: OpenWeatherMap API, Google Gemini API;
* стандарти розробки веб-додатків W3C.

1.5 Означення та абревіатури

* ui (User Interface): Користувацький інтерфейс;
* ux (User Experience): Досвід користувача;
* api (Application Programming Interface): Програмний інтерфейс додатків;
* jwt (JSON Web Token): Стандарт для створення токенів доступу, що використовуються для автентифікації;
* ai (Artificial Intelligence): Штучний інтелект;
* frontend: Клієнтська частина веб-додатку, що працює в браузері користувача;
* backend: Серверна частина веб-додатку, що обробляє логіку та взаємодіє з базою даних;
* db (Database): База даних;
* crud: Акронім для Create, Read, Update, Delete (створення, читання, оновлення, видалення) – основні операції з даними;
* mui (Material-UI): Бібліотека компонентів React, що реалізує Material Design.

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

"SmartStyle" – це незалежний веб-додаток, який функціонує за моделлю клієнт-сервер. Фронтенд-частина, розроблена на React, взаємодіє з бекенд-частиною (Node.js з фреймворком Express.js) через RESTful API. Бекенд відповідає за бізнес-логіку, керування базою даних PostgreSQL [3] та інтеграцію із зовнішніми API для отримання даних про погоду та генерації рекомендацій. Додаток розроблено як самодостатнє рішення для персонального стилю, не інтегроване в більші системи.

2.2 Функції продукту

* автентифікація користувачів: Реєстрація, вхід та вихід із системи;
* керування профілем: Перегляд та редагування імені користувача;
* керування гардеробом: Додавання, перегляд та видалення предметів одягу;
* створення образів: Візуальний підбір одягу на інтерактивному манекені;
* збереження пресетів: Можливість зберігати комбінації одягу;
* погодні дані: Відображення поточної погоди та прогнозу;
* ai-рекомендації: Генерація пропозицій щодо образу на основі погоди та критеріїв користувача;
* тегування одягу: Створення та керування користувацькими тегами;
* локалізація: Підтримка англійської та української мов.

2.3 Характеристики користувачів

* зареєстровані користувачі: Основна цільова аудиторія. Користувачі, які пройшли процес реєстрації та мають доступ до повного функціоналу програми. Передбачається, що вони мають базові навички роботи з веб-додатками;
* незареєстровані користувачі: Відвідувачі сайту, які можуть ознайомитися з продуктом через загальні сторінки (наприклад, "Про нас"), але не мають доступу до персоналізованих функцій.

2.4 Загальні обмеження

* залежність від зовнішніх API: Функціонал рекомендацій та погодних даних повністю залежить від доступності та коректної роботи API OpenWeatherMap та Google Gemini;
* локальне розгортання: На етапі розробки бекенд розгортається локально, що обмежує доступність для зовнішніх користувачів;
* веб-середовище: Додаток призначений для роботи у веб-браузерах і не оптимізований як нативний мобільний додаток;
* зберігання зображень: Наразі керування зображеннями відбувається через URL-адреси. Для масштабованого використання потрібне більш надійне рішення для зберігання файлів (наприклад, хмарне сховище).

2.5 Припущення й залежності

* користувачі повинні мати стабільне інтернет-з'єднання;
* правильно налаштовані API-ключі для зовнішніх сервісів повинні бути доступні в оточенні бекенду;
* база даних PostgreSQL повинна бути встановлена, налаштована та доступна для підключення;
* бекенд-сервер та фронтенд-додаток повинні бути розгорнуті в сумісних середовищах.

3. КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

* дизайн: Сучасний, чистий та адаптивний дизайн, що забезпечує коректне відображення на різних пристроях. Використання темної палітри кольорів з фіолетовими акцентами;
* навігація: Інтуїтивно зрозуміла навігація через верхню навігаційну панель та футер;
* елементи управління: Використання компонентів Material-UI для стандартних елементів для забезпечення консистентності;
* завантаження зображень: Зручний інтерфейс для завантаження зображень одягу (drag-and-drop або вибір файлу).

3.1.2 Апаратний інтерфейс

Вимоги до апаратного забезпечення системи є стандартними для сучасних веб-додатків та не передбачають використання спеціалізованого обладнання. З боку клієнта, для повноцінної роботи з додатком "SmartStyle" користувачеві потрібен будь-який пристрій (персональний комп'ютер, ноутбук, планшет або смартфон) з встановленим сучасним веб-браузером та стабільним доступом до мережі Інтернет. Це забезпечує максимальну доступність продукту для широкого кола користувачів.

З боку сервера, для розгортання та функціонування бекенд-частини та бази даних потрібен сервер, фізичний або віртуальний. Конкретні вимоги до обчислювальних ресурсів, таких як потужність центрального процесора (CPU) та обсяг оперативної пам'яті (RAM), не є фіксованими та залежать від очікуваного навантаження, зокрема кількості одночасних користувачів. Для етапу розробки та тестування достатньо ресурсів стандартної локальної машини розробника.

3.1.3 Програмний інтерфейс

Програмна архітектура "SmartStyle" побудована на взаємодії кількох ключових програмних компонентів та технологій.

Клієнтська частина (Frontend) реалізована на мові JavaScript (стандарт ES6+) з використанням бібліотеки React.js, яка є основою для побудови користувацького інтерфейсу. Навігація між сторінками додатку забезпечується бібліотекою React Router DOM. Взаємодія з серверним API відбувається за допомогою бібліотеки Axios, що дозволяє надсилати асинхронні HTTP-запити. Для інтернаціоналізації інтерфейсу використовується бібліотека i18next. Візуальне оформлення та компоненти інтерфейсу реалізовані за допомогою Material-UI (MUI).

Серверна частина (Backend) побудована на платформі Node.js з використанням фреймворку Express.js для організації RESTful API. Для взаємодії з базою даних PostgreSQL використовується клієнт pg. Безпека паролів забезпечується за допомогою бібліотеки bcrypt для їх хешування, а автентифікація на основі токенів реалізована за допомогою jsonwebtoken (JWT). Для керування змінними середовища використовується пакет dotenv.

В якості системи управління базами даних виступає PostgreSQL [7]. Також система інтегрується із зовнішніми програмними інтерфейсами: OpenWeatherMap API для отримання метеорологічних даних та Google Gemini API для генерації інтелектуальних рекомендацій.

3.1.4 Комунікаційний протокол

Взаємодія між клієнтською та серверною частинами системи "SmartStyle" здійснюється за допомогою протоколу HTTP/HTTPS. В основі комунікації лежить архітектурний стиль REST (Representational State Transfer), що забезпечує безстатусний (stateless) обмін даними.

Для передачі даних між клієнтом та сервером використовується уніфікований формат JSON (JavaScript Object Notation), який є легким, читабельним для людини та легко обробляється на стороні JavaScript. Автентифікація користувача для доступу до захищених ресурсів API реалізована за допомогою Bearer токенів на основі стандарту JWT. Після успішного входу користувач отримує токен, який він повинен додавати до заголовка Authorization кожного наступного запиту до захищених кінцевих точок.

3.1.5 Обмеження пам’яті

Хоча система не має жорстких лімітів на використання пам'яті, її архітектура спроєктована з урахуванням принципів ефективності. На клієнтській стороні додаток оптимізовано для раціонального використання пам'яті браузера, щоб уникнути "витоків" та забезпечити плавну роботу інтерфейсу навіть на менш потужних пристроях. Це досягається за рахунок ефективного керування станом та життєвим циклом компонентів React.

На серверній стороні було прийнято важливе архітектурне рішення для мінімізації навантаження на базу даних та використання пам'яті. Замість зберігання бінарних даних зображень одягу безпосередньо в базі даних (у вигляді BLOB-об'єктів), система зберігає лише URL-адреси, що вказують на місцезнаходження файлів. Самі файли зображень передбачається зберігати у файловій системі сервера або в спеціалізованому хмарному сховищі, що є значно більш продуктивним та масштабованим підходом.

3.1.6 Операції

Програмний інтерфейс додатку (API) надає набір кінцевих точок (endpoints) для виконання всіх необхідних операцій. Ці операції логічно згруповані за функціональними модулями:

* автентифікація (/api/auth): Ця група операцій відповідає за керування життєвим циклом користувача, включаючи реєстрацію нового акаунту (POST /register), вхід у систему (POST /login), оновлення даних профілю (PUT /users/:userId) та отримання інформації про поточного автентифікованого користувача (GET /me);
* керування гардеробом (/api/clothing, /api/tags): Набір CRUD-операцій для взаємодії з особистим гардеробом. Включає отримання списку одягу, додавання нових предметів, їх видалення, а також створення та отримання користувацьких тегів;
* інтелектуальні функції (/api/ai, /api/weather): Операції, що надають доступ до "розумних" можливостей додатку. Це, зокрема, отримання AI-рекомендацій для часової шкали (POST /ai/timeline-suggestion) та запит актуальних погодних даних (GET /weather).

3.1.7 Функції продукту

Ключовий функціонал продукту визначається набором можливостей, доступних кінцевому користувачеві. Система забезпечує повний цикл автентифікації та авторизації, дозволяючи користувачам безпечно створювати облікові записи та отримувати доступ до персонального простору. В рамках керування профілем користувач може змінювати своє відображуване ім'я.

Центральною функцією є керування гардеробом. Користувачі можуть завантажувати зображення предметів одягу, класифікувати їх за категоріями та призначати їм набір персоналізованих тегів. На основі цього цифрового гардеробу реалізована функція підбору образів, де користувач може візуально комбінувати речі на інтерактивному манекені та зберігати вдалі комбінації як "пресети" для майбутнього використання. Найбільш інноваційною є функція AI-рекомендацій, яка аналізує погоду, критерії користувача та його гардероб для генерації пропозицій щодо образу.

3.1.8 Припущення й залежності

Коректна робота системи "SmartStyle" базується на кількох ключових припущеннях та залежностях. Передбачається, що для успішної роботи криптографічних функцій автентифікації на сервері має бути налаштована змінна середовища JWT\_SECRET. Якість та точність даних про погоду та AI-рекомендацій безпосередньо залежать від стабільності та доступності зовнішніх API (OpenWeatherMap та Google Gemini), а також від якості даних, що надаються цими сервісами.

Також робиться припущення, що користувачі завантажуватимуть коректні файли зображень у підтримуваних форматах (PNG, JPG, JPEG, SVG) і що для виконання операцій, які вимагають ідентифікації користувача (наприклад, редагування профілю), його user.id буде коректно отриманий з контексту автентифікації на стороні клієнта.

3.2 Властивості програмного продукту

Система повинна надавати повний набір функцій для реалізації свого основного призначення. Користувач повинен мати можливість пройти повний цикл від реєстрації та входу до активного використання всіх можливостей додатку. Система має забезпечувати сесійну стійкість після входу, дозволяючи користувачеві залишатися в системі при перезавантаженні сторінки.

Ключовою функціональною вимогою є надання повного набору CRUD-операцій (створення, читання, оновлення, видалення) для управління особистими предметами одягу. Система повинна реалізовувати механізм генерації рекомендацій щодо образів, який враховує погодні умови та критерії, задані користувачем. Також система зобов'язана відображати актуальні погодні дані та надавати інструменти для створення та збереження персоналізованих наборів одягу (пресетів). Нарешті, інтерфейс має підтримувати повну багатомовність з можливістю перемикання між українською та англійською мовами.

3.3 Атрибути програмного продукту

3.3.1 Надійність

Система спроєктована з фокусом на високу надійність. Це досягається за рахунок ретельної обробки помилок як на клієнтській, так і на серверній стороні, що запобігає несподіваним збоям та надає користувачеві зрозумілі повідомлення про проблеми. Цілісність даних у базі даних PostgreSQL забезпечується використанням обмежень (constraints) та механізмів транзакцій, що унеможливлює появу неузгоджених даних.

3.3.2 Доступність

Додаток розробляється з метою забезпечення цілодобової доступності (24/7), за винятком коротких періодів планового технічного обслуговування. Зручність використання є пріоритетом: інтерфейс є інтуїтивно зрозумілим, що дозволяє користувачам з різним рівнем технічної підготовки легко освоїти функціонал системи.

3.3.3 Безпека

Безпека є одним з найважливіших атрибутів системи. Захист даних користувачів реалізовано шляхом зберігання паролів у вигляді криптографічних хешів за допомогою алгоритму bcrypt. Автентифікація всіх запитів до захищених ресурсів API відбувається за допомогою JWT токенів з обмеженим терміном дії. Авторизація гарантує, що користувачі мають доступ лише до власних персональних даних. Також система має базові механізми захисту від поширених веб-уразливостей, таких як SQL-ін'єкції та XSS.

3.3.4 Супроводжуваність

Для забезпечення легкості підтримки та подальшого розвитку, структура коду є модульною та добре організованою. Код відповідає загальноприйнятим стандартам кодування, а критичні та складні його секції належним чином прокоментовані. Усі зовнішні залежності чітко визначені у файлах package.json, що спрощує розгортання та налаштування середовища розробки.

3.3.5 Переносимість

Система спроєктована з урахуванням переносимості. Фронтенд-додаток забезпечує крос-браузерну сумісність і коректно функціонує в останніх версіях основних веб-браузерів (Chrome, Firefox, Safari, Edge). Крос-платформенність серверної частини гарантується використанням Node.js та PostgreSQL, які можуть бути розгорнуті на різних операційних системах (Windows, macOS, Linux).

3.3.6 Продуктивність

Продуктивність системи є важливим нефункціональним атрибутом. Час відповіді API для більшості стандартних операцій не повинен перевищувати 500 мс. Час завантаження основних сторінок додатку оптимізовано для забезпечення швидкого доступу. Розуміючи, що продуктивність AI-операцій залежить від зовнішнього сервісу, в інтерфейсі передбачено візуальні індикатори завантаження, щоб інформувати користувача про процес очікування.

3.4 Вимоги бази даних

Для зберігання даних використовується система управління базами даних PostgreSQL. Схема бази даних спроєктована за реляційною моделлю і включає набір таблиць для зберігання ключових сутностей: users (користувачі), clothing\_items (предмети одягу), tags (теги), outfit\_sets (збережені образи). Для реалізації зв'язків "багато-до-багатьох" (наприклад, між одягом та тегами) використовуються спеціальні асоціативні таблиці. Для оптимізації продуктивності запитів передбачено використання індексів на полях, які часто використовуються для пошуку та зв'язування, зокрема на зовнішніх ключах.

3.5 Інші вимоги

Окрім вищезазначених, до системи висуваються й інші вимоги. Зручність використання має бути максимальною, а процес взаємодії – логічним та простим. Код повинен супроводжуватися належною внутрішньою документацією у вигляді коментарів. Система обробки помилок повинна надавати користувачеві не лише технічну інформацію, а й зрозумілі підказки щодо вирішення проблеми, що покращує загальний досвід користування продуктом.

## **ДОДАТОК В**

Слайди презентації

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.1

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.2

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.3

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.4

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.5

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.6

Изображение выглядит как текст, снимок экрана, диаграмма, Прямоугольник

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.7

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.8

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.9

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.10

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.11

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.12

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.13