Міністерство освіти і науки України

Одеський національний політехнічний університет

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Рудас Дмитро Борисович,

студент групи АС-151

**КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА**

Інтерактивний навчальний конструктор беклогів спринтів

Спеціальність:

121 – Інженерія програмного забезпечення

Спеціалізація:

Інженерія програмного забезпечення

Керівник:

Любченко Віра Вікторівна,

докт. техн. наук, професор

Одеса – 2020

**ЗМІСТ**

[Завдання на кваліфікаційну роботу 4](#_Toc58514478)

[Завдання на розробку розділу «Охорона праці та безпека в надзвичайних ситуаціях» 6](#_Toc58514479)

[Анотація 7](#_Toc58514480)

[Вступ 8](#_Toc58514481)

[1 Проблеми існуючих рішень для конструювання беклогів спринтів 10](#_Toc58514482)

[1.1 Аналіз потреб користувачів навчального конструктору беклогів спринтів 10](#_Toc58514483)

[1.2 Огляд існуючих рішень ведення беклогів 10](#_Toc58514484)

[1.3 Порівняння аналізованих рішень 12](#_Toc58514485)

[2 Специфікація вимог для інтерактивного навчального конструктору спринтів 14](#_Toc58514486)

[2.1 Функціональні вимоги 14](#_Toc58514487)

[2.2 Нефункціональні вимоги 20](#_Toc58514488)

[3 Проектування інтерактивного навчального конструктору спринтів 23](#_Toc58514489)

[3.1 Архітектура системи 23](#_Toc58514490)

[3.1.1 Компоненти системи 23](#_Toc58514491)

[3.1.2 Вибір реалізації сховища даних. 26](#_Toc58514492)

[3.2 Проектування структури даних для збереження та відтворення беклогу 30](#_Toc58514493)

[3.3 Проектування структури даних для імпорту юзер-сторіс до беклогу 33](#_Toc58514494)

[3.4 Проектування інтерфейсу користувача 35](#_Toc58514495)

[3.4.1 Вибір формату компонування функціональних елементів 35](#_Toc58514496)

[3.4.2 Проектування головної сторінки 36](#_Toc58514497)

[3.4.3 Проектування інтерфейсу з інформацією про юзер-сторі 39](#_Toc58514498)

[3.4.4 Проектування інтерфейсу з інформацією про спринт 41](#_Toc58514499)

[3.4.5 Проектування меню.. 44](#_Toc58514500)

[3.5 Проектування дизайну вмісту PDF файлу експорту 46](#_Toc58514501)

[4 Розробка інтерактивного навчального конструктору спринтів 47](#_Toc58514502)

[4.1 Вибір стеку технологій та інструментів для розробки 47](#_Toc58514503)

[4.2 Програмна реалізація 48](#_Toc58514504)

[5 Тестування інтерактивного навчального конструктору спринтів 52](#_Toc58514505)

[5.1 Системне тестування 52](#_Toc58514506)

[5.2 Інструкція з встановлення 58](#_Toc58514507)

[5.3 Опис використання системи 60](#_Toc58514508)

[6 Охорона праці 70](#_Toc58514509)

[Висновки 72](#_Toc58514510)

[Список використаної літератури 73](#_Toc58514511)

[Додаток А. Лістинг 76](#_Toc58514512)

[Додаток Б. Питання з охорони праці та безпеки в надзвичайних ситуаціях 77](#_Toc58514513)

Міністерство освіти і науки України

Одеський національний політехнічний університет

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Рівень вищої освіти: другий (магістерський)

Спеціальність: 121 – Інженерія програмного забезпечення

Спеціалізація: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_ Крісілов В. А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2020 р.

## ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Рудас Дмитра Борисовича, група АС-151

1. Тема роботи: Інтерактивний навчальний конструктор беклогів спринтів

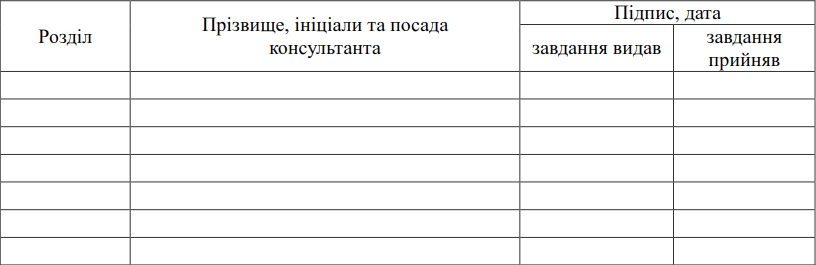
Керівник роботи: Любченко Віра Вікторівна, докт. техн. наук, професор

затверджені наказом ректора від 29 жовтня 2020 р. № 412-в

2. Зміст роботи: проблеми існуючих рішень для конструювання беклогів спринтів, специфікація вимог для інтерактивного навчального конструктору спринтів, проектування інтерактивного навчального конструктору спринтів, розробка інтерактивного навчального конструктору спринтів, тестування інтерактивного навчального конструктору спринтів, охорона праці

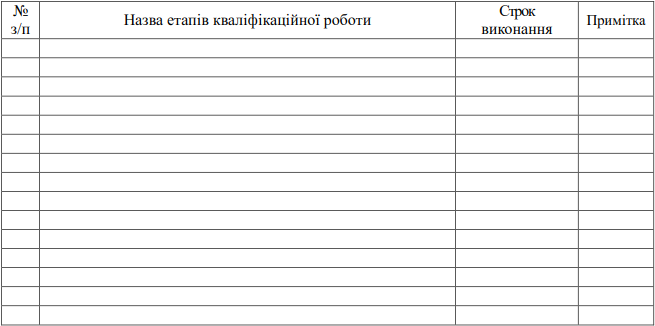
3. Перелік ілюстративного матеріалу: відповідно до слайдів презентації

4. Консультанти розділів роботи



5. Дата видачі завдання: «02» листопада 2020 р.

**КАЛЕНДАРНИЙ ПЛАН**



**Здобувач вищої освіти** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. Б. Рудас

**Керівник роботи**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. В. Любченко

## ЗАВДАННЯ на розробку розділу «Охорона праці та безпека в надзвичайних ситуаціях»

Рудас Дмитра Борисовича, група АС-151

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Тема роботи: Інтерактивний навчальний конструктор беклогів спринтів

Зміст розділу:

1. Аналіз умов праці і вибір заходів і засобів захисту від небезпечних і шкідливих виробничих факторів.

2. Аналіз техногенних небезпек і вибір заходів і засобів забезпечення безпеки у надзвичайних ситуаціях.

3. Індивідуальне завдання

Керівник роботи Консультант з охорони праці та

БНС

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. В. Любченко \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Ю. Москалюк

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2020 р. «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2020 р.

## АНОТАЦІЯ

Метою роботи є модернізація навчального процесу за рахунок впровадження інтерактивного конструктору беклогів спринтів. Методи розробки базуються на технологіях JavaScript, React, Redux, HTML, CSS.

Як результат роботи отримано програмну систему, яка дозволяє відпрацьовувати навички складання створювати беклогів спринтів у здобувачів вищої освіти. Для цього були проаналізовані недоліки системи, що вирішують аналогічні завдання, сформовані вимоги до системи, спроектовано архітектуру, виконано програмну реалізацію системи та протестовано її на предмет досягнення мети роботи.

Ключові слова: планування, беклог, Agile, веб-застосунок, конструктор, React, JavaScript.

**ABSTRACT**

The purpose of the work is to modernize the educational process by introducing the interactive backlog sprint constructor. Technologies used in development are JavaScript, React, Redux, HTML, and CSS.

The result of the work done is a software system which allows creating sprint backlogs while creating an accent on the educational process. To accomplish the work, the analogs of the system were analyzed in terms of lacking educational features, the requirements were gathered, the architecture of the system was created, the software was implemented, and was tested accordingly to satisfy the goal of the work.

Keywords: planning, backlog, Agile, web-application, constructor, React, JavaScript.

## ВСТУП

Методологія розробки Agile є сучасною та активно використовується в розробці [1], тому є бажаною для вивчення в навчальних закладах. Наприклад, у 2017 році методологію Agile використовували 71% опитаних компаній за даними Project Management Institute [2].

Сьогодні існує багато інструментів, що дозволяють виконувати планування за методологією Agile, проте їх важко застосувати як інструмент для навчання. Існуючим інструментам не вистачає навчальних властивостей, таких як підказки або заборони для того, щоб допомогти або скорегувати дії студентів.

Тому актуальною задачею є розробка програмного продукту, який забезпечує вимоги навчального процесу шляхом створення обмежень, що вимагають від студентів дотримуватися кращих практик (best practice) при формування беклогів спринтів.

Метою роботи є модернізація навчального процесу за рахунок впровадження інтерактивного конструктору беклогів спринтів.

Для досягнення мети необхідно вирішити такі завдання:

а) проаналізувати існуючі рішення для конструювання беклогів спринтів та узагальнити результати аналізу з точки зору придатності для застосування в навчальному процесі, виявити інформаційні потреби для проєктування програмного продукту;

б) розробити, проаналізувати та систематизувати вимоги до програмного продукту, призначеного для досягнення мети;

в) розробити архітектуру програмного забезпечення для реалізації вимог;

г) розробити програмний продукт з застосуванням відповідних засобів розробки;

д) спланувати і здійснити тестування і валідацію програмного продукту.

Робота має таку структуру. В розділі 1 проаналізовано проблему, що вирішує робота, розкривається актуальність і доцільність роботи шляхом аналізу та порівняння з відомими рішеннями. В розділі 2 наведені специфікації вимог як функціональних, так і нефункціональних. В розділі 3 спроектовано архітектуру системи, обрані архітектурні рішення, наведені структури даних та спроектовано інтерфейс користувача. В розділі 4 описані стек технологій, інструменти розробки та програмна реалізація системи. В розділі 5 узагальнюються результати тестування отриманого програмного продукту по відношенню до вимог та описується типове використання системи. Розділ 6 містить виконане завдання з охорони праці та безпеки в надзвичайних ситуаціях. В останньому розділі підведено підсумки виконаних робіт з розробки інтерактивного конструктору беклогів спринтів. Додатки містять лістинг коду розробленого програмного забезпечення, а також розбір питань з охорони праці та безпеки у надзвичайних ситуаціях.

## 1 ПРОБЛЕМИ ІСНУЮЧИХ РІШЕНЬ ДЛЯ КОНСТРУЮВАННЯ БЕКЛОГІВ спринтів

Аналіз існуючих рішень є важливим етапом розробки інформаційної системи. Він дозволяє зрозуміти ситуацію на ринку у заданій предметній області, а також наявний прогрес в дослідженнях. Розглянемо існуючі рішення для конструювання беклогів спринтів (sprint backlog). Кінцевими користувачами системи є студенти другого рівня освіти за спеціальністю 122 – Комп’ютерні науки, тому аспекти порівняння будуть пов’язані саме з точки зору їх потреб у навчанні.

### 1.1 Аналіз потреб користувачів навчального конструктору беклогів спринтів

Навчальний конструктор розробляється з метою модернізувати процес навчання, шляхом автоматизації перевірки правильності застосування технологій Agile, а саме створення беклогів спринтів.

Для процесу навчання важливі такі аспекти роботи системи, як можливість додавання та редагування юзер-сторіс (user stories), можливість збереження та відновлення прогресу складання беклогів, можливість імпорту набору юзер-сторіс зі стороннього ресурсу; врахування сторі поінтів (story points), пріоритетів, ризиків, залежностей між сторісами, велосіті (velocity).

Саме за цими аспектами будуть порівнюватися існуючі рішення.

### 1.2 Огляд існуючих рішень ведення беклогів

Існує багато готових рішень, що дозволяють створювати беклоги, планувати спринти та забезпечують потужні інструменти для професіоналів.

Переважна частина існуючих рішень забезпечують у своїй ціновій політиці безкоштовні тарифи для індивідуальних осіб, що має задовольняти вимоги до інструментів для навчального процесу, оскільки витрата бюджетних коштів має бути оправдана. Проте існує недолік, який притаманний багатьом з них, – вони майже неадаптовані для навчального процесу.

Через те що підхід Agile до розробки програмного забезпечення об’єднує велику кількість різноманітних технологій [3], обмежувати користувача під час планування беклогів для дотримання певної технології було б шкідливим для популярності професійного конструктору беклогів спринтів.

Проаналізуємо існуючі рішення на прикладі чотирьох популярних засобів.

**Atlassian Jira** – комерційна система з веб-інтерфейсом для відслідковування помилок, призначена для організації взаємодії з користувачами, хоча в деяких випадках використовується і для управління проектами. Розроблена компанією Atlassian, є одним з двох її основних продуктів (поряд з вікі-системою Confluence).

**Atlassian Trello** – хмарна програма для управління проектами невеликих груп, розроблена Fog Creek Software. Наразі викуплена компанією Atlassian. Trello використовує парадигму управління проектами Канбан.

**Nulab Backlog** – це онлайн-рішення для спрощення роботи з щоденними завданнями, гнучкого управління проектами та організації своєї команди для досягнення більшої кількості цілей**.**

**Taiga Agile Taiga** – це сервіс для управління проектами та спринтами за гнучкими методологіями для розробників і дизайнерів.

### 1.3 Порівняння аналізованих рішень

Важливо порівняти переваги та недоліки існуючих рішень для того, щоб створена система поєднувала у собі найкраще, що існує на ринку для досягнення мети роботи, та була позбавлена недоліків, що притаманні деяким програмним продуктам.

Серед розглянутих характеристик є такі, що жодна з систем не реалізує. Ці характеристики включені до порівняльної таблиці 1.1 через те, що вони мають позитивний вплив на досягнення поставленої мети роботи. Імпорт набору юзер-сторіс дозволяє скоротити час викладачу на підготовку завдання студентам. Зазначення ризиків, пріоритетів та залежностей юзер-сторі допомагає студентам краще зрозуміти, як правильно виконувати складання спринтів.

Для порівняння скористаємося табл. 1.1. В таблиці наведені можливості рішення та позначення чи містить у собі рішення цей аспект: «+» якщо містить та «–» якщо не містить.

##### Таблиця 1.1 – Порівняння існуючих рішень як інструментів для модернізації навчального процесу.

| **Характеристика** | **Jira** | **Trello** | **Nulab** | **Taiga** |
| --- | --- | --- | --- | --- |
| Можливість додавання та редагування юзер-сторіс | + | + | + | + |
| Можливість збереження та відновлення прогресу складання беклогів | + | + | + | + |
| Можливість імпорту набору юзер-сторіс зі стороннього ресурсу | – | – | – | – |
| Можливість задавати сторі-поінти при створенні юзер-сторі | – | – | – | + |
| Можливість вказувати пріоритети юзер-сторі | – | – | – | – |

Продовження таблиці 1.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Характеристика** | **Jira** | **Trello** | **Nulab** | **Taiga** |
| Можливість вказувати ризики для юзер-сторі | – | – | – | – |
| Можливість створювати залежності між сторісами | – | – | – | – |
| Врахування велосіті при створенні беклогу | – | – | – | + |

При порівнянні існуючих рішень стає видно, що вони майже не надають засобів для навчального процесу. Необхідно щоб кінцевий продукт направляв студента та повідомляв про те, що студент може робити неправильно.

Існуючі рішення занадто гнучкі та сприймають користувачів за кваліфікованих експертів, що не є доречим для навчальних цілей.

Таким чином, розроблювана система має відповідати усім характеристикам, що були вказані у табл. 1.1, щоб комплексно вирішувати задачу модернізації навчального процесу.

## СПЕЦИФІКАЦІЯ ВИМОГ ДЛЯ ІНТЕРАКТИВНОГО НАВЧАЛЬНОГО КОНСТРУКТОРУ СПРИНТІВ

Специфікація вимог є важливим етапом розробки інформаційної системи. Вимоги описують поведінку системи, її бажані характеристики що формуються в список функціональних та нефункціональних вимог. Це надає можливість спланувати розробку проекту, оцінити етапи її розробки та передбачити терміни її створення.

Специфікація вимог програмного забезпечення згідно до ISO/IEC/IEEE 29148:2011 – Requirements engineering є описом поведінки програми, яку потрібно розробити [4]. Вона включає ряд сценаріїв користувача, які описують всі варіанти взаємодії між користувачами і програмним забезпеченням.

Функціональні вимоги описуються за допомогою сценаріїв користувача. Сценарії описують можливі варіанти використання системи. Окрім варіантів використання, специфікації описують також нефункціональні вимоги. Вони накладають обмеження на візуальний склад системи, або її реалізацію (продуктивність, характеристики якості).

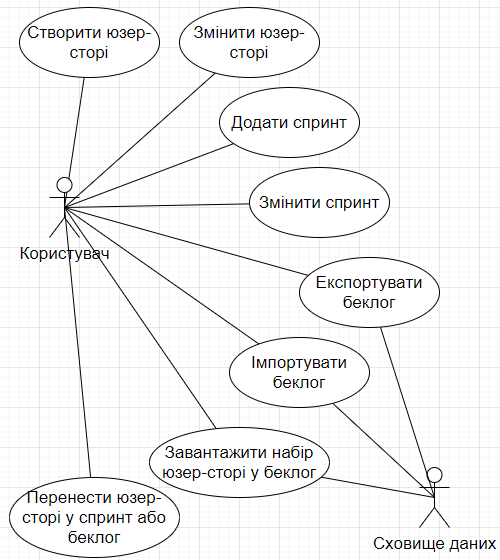
### 2.1 Функціональні вимоги

Для того, щоб описати загальні вимоги до функціональної поведінки інтерактивного навчального конструктору спринтів, застосуємо діаграму варіантів використання (рис. 2.1). Діаграма варіантів використання входить в набір UML-моделей та використовується для відображення того, що саме робить система з точки зору її використання [5].

На діаграмі представлено два актора:

1. користувач – користувач системи, що має доступ до функціоналу системи;
2. сховище даних – місце, де дані зберігаються.

Також є третій актор, що пов’язаний з усіма варіантами використання – Система, проте він опущений на діаграмі для ясності.



###### Рисунок 2.1 – Діаграма варіантів використання

Актори асоціюються з варіантами використання (8 варіантів), сценарії яких будуть описані далі.

**Варіант використання «Створити юзер-сторі»**

*Діючі особи:* Користувач – К, Система – С.

*Передумова:* немає.

*Мета:* створити юзер-сторі для подальшого планування беклогу.

*Успішний сценарій:*

1. К запрошує у С створення нової юзер-сторі. С показує К форму створення нової юзер-сторі.
2. К заповнює форму створення юзер-сторі та запрошує у С збереження юзер-сторі. С перевіряє правильність заповнення форми, зберігає нову юзер-сторі та приміщає її у беклог продукту.

*Результат:* нова юзер-сторі створена та приміщена у беклог продукту.

*Розширення:*

2а. К запитує у С скасування створення юзер-сторі.

2а.1. С запитує у К підтвердження його дії.

2а.2. К підтверджує дію. С ховає форму створення юзер-сторі та видаляє незбережені дані

*Результат:* нова юзер-сторі не створена.

**Варіант використання «Змінити юзер-сторі»**

*Діючі особи:* Користувач – К, Система – С.

*Передумова:* юзер-сторі створена.

*Мета:* змінити дані юзер-сторі для її актуалізування.

*Успішний сценарій:*

1. К запитує у С зміну юзер-сторі. С показує К форму зміни юзер-сторі, заповнену даними змінюваної юзер-сторі.
2. К змінює дані на потрібні та запитує у С збереження змін. С зберігає зміни.

*Результат:* дані юзер-сторі змінені.

*Розширення:*

1а. Юзер-сторі знаходиться не в беклозі продукту.

1а.1. С повідомляє К, що змінити дані юзер-сторі неможливо, оскільки вона знаходиться не в беклозі продукту.

*Результат:* дані юзер-сторі не змінені.

2а. Заповнені дані некоректні.

2а.1. С повідомляє К, які дані некоректні. Перехід до П.2.

**Варіант використання «Додати спринт»**

*Діючі особи:* Користувач – К, Система – С.

*Передумова:* немає.

*Мета:* створити спринт для подальшого його заповнення.

*Успішний сценарій:*

1. К запитує у С створення спринту. С показує К форму створення спринту.
2. К заповнює форму створення спринту та запитує у С збереження спринту. С перевіряє правильність заповнення форми та зберігає спринт.

*Результат:* новий спринт створено та збережено.

*Розширення:*

2а. Заповнені дані некоректні.

2а.1. С повідомляє К, які дані некоректні. Перехід до П.2.

**Варіант використання «Змінити спринт»**

*Діючі особи:* Користувач – К, Система – С.

*Передумова:* спринт створено.

*Мета:* змінити дані існуючого спринту.

*Успішний сценарій:*

1. К запитує у С зміну даних існуючого спринту. С показує К форму для зміни існуючого спринту, заповнену даними існуючого спринту.
2. К змінює дані існуючого спринту. С перевіряє коректність введених даних та зберігає спринт.

*Результат:* дані існуючого спринту змінено.

*Розширення:*

2а. Заповнені дані некоректні.

2а.1. С повідомляє К, які дані некоректні. Перехід до П.2.

**Варіант використання «Експортувати беклог»**

*Діючі особи:* Користувач – К, Система – С, Сховище Даних – СД .

*Передумова:* немає.

*Мета:* експортувати беклог до файлу для збереження даних поза програмою, або майбутнього його імпорту

*Успішний сценарій:*

1. К запитує у С експорт беклогу. С показує К форму експорту беклогу.
2. К вибирає місце збереження файлу з беклогом у СД та назву збережуваного файлу. С експортує беклог до СД.

*Результат:* беклог експортовано до СД.

*Розширення:*

2а. У СД немає вільного місця для збереження файлу експорту.

2а.1. С показує, що вільного місця у СД немає та пропонує К вибрати інше місце. Перехід до П.2.

**Варіант використання «Імпортувати беклог»**

*Діючі особи:* Користувач – К, Система – С, Сховище Даних – СД.

*Передумова:* в СД існує файл беклогу.

*Мета:* імпортувати беклог з файлу до конструктору беклогів.

*Успішний сценарій:*

1. К запитує у С імпортувати беклог. С показує К форму імпорту беклогу та попереджає К про те, що імпортовані дані перезапишуть поточний стан беклогу.
2. К обирає файл беклогу. С імпортує беклог до конструктору.

*Результат:* беклог імпортовано, всі незбережені дані перезаписані.

*Розширення:*

2а. С не може прочитати вказаний К файл беклогу.

2а.1. С повідомляє К що не може прочитати вказаний файл.

*Результат:* беклог не було імпортовано. Поточний стан беклогу не змінено.

2б. Дані, що зберігаються в беклозі файлу, некоректні.

2б.1. С повідомляє К, що не може імпортувати вказаний файл через некоректні дані.

*Результат:* беклог не було імпортовано. Поточний стан беклогу не змінено.

**Варіант використання «Завантажити набір юзер-сторі у беклог»**

*Діючі особи:* Користувач – К, Система – С, Сховище Даних – СД .

*Передумова:* існує файл з набором юзер-сторі.

*Мета:* завантажити набір раніше записаних юзер-сторі у беклог, не втрачаючи поточних даних.

*Успішний сценарій:*

1. К запитує у С завантаження набору юзер-сторі з файлу. С показує К форму вибору файлу з набором юзер-сторі.
2. К обирає файл з набором юзер-сторі в необхідному форматі. С завантажує набір юзер-сторі з файлу та додає його до беклогу продукту.

*Результат:* набір юзер-сторі додано до беклогу продукту, поточні дані не втрачено.

*Розширення:*

2а. С не може прочитати вказаний К файл з набором юзер-сторі.

2а.1. С повідомляє К, що не може прочитати вказаний файл.

*Результат:* набір юзер-сторі не було імпортовано. Поточний стан беклогу не змінено.

2б. Дані, що зберігаються в файлі, некоректні.

2б.1. С повідомляє К, що не може імпортувати вказаний файл через некоректні дані.

*Результат:* набір юзер-сторі не було імпортовано. Поточний стан беклогу не змінено.

**Варіант використання «Перенести юзер-сторі у спринт або беклог»**

*Діючі особи:* Користувач – К, Система – С.

*Передумова:* попередньо створені спринт та юзер-сторі.

*Мета:* перенести юзер-сторі у спринт або беклог для планування беклогу спринту.

*Успішний сценарій:*

1. К намагається перенести юзер-сторі до певного спринту або беклогу. С перевіряє, чи це можливо, та у разі можливості переносить юзер-сторі та сповіщає про це К.
2. К підтверджує перенос. С зберігає зміни

*Результат:* юзер-сторі перенесено до спринту або беклогу.

*Розширення:*

1а. Перенос юзер-сторі неможливий.

1а.1. С повідомляє К, що перенос юзер-сторі неможливий.

*Результат:* юзер-сторі не перенесено, поточний стан беклогу не змінено.

2а. К не підтвердив перенос.

2а.1. С скасовує зміни.

*Результат*: юзер-сторі не перенесено, поточний стан беклогу не змінено.

### 2.2 Нефункціональні вимоги

Нефункціональні вимоги, або як їх ще називають вимоги якості [6], визначають якісні характеристики, яким повинна відповідати система. Опишемо вимоги до нашої системи.

Функціональність системи передбачає, що вона повинна надавати можливості:

* візуалізувати поточний стан беклогу;
* взаємодіяти з локальним сховищем, бо це умова для збереження, або завантаження даних;
* надавати можливість взаємодіяти з беклогом, змінюючи його стан;
* зберігати та змінювати ризики, пріоритет та кількість сторі поінтів в юзер-сторі для подальшого використання цих даних в допомозі користувачеві дотримуватися правил планування беклогу;
* задавати залежності одних юзер-сторі від інших та дотримуватися цих залежностей при формуванні беклогу.

Надійність системи потребує, щоб через візуальний інтерфейс взаємодії з користувачем система повідомляла користувачу про некоректні дії, або помилки, що можуть виникнути через некоректні файли у сховищі даних.

Щоб працювати без помітних затримок, система повинна реагувати на переміщення юзер-сторі не довше ніж за 10 мс на комп’ютері з тактовою частотою процесора не нижче за 1.6 ГГц і 1 ядром та завантаженістю не більше ніж на 70%. За тих самих характеристик комп’ютера, зчитування даних з локального сховища, розмір якого не перевищує 5 Кб, повинно займати не більше ніж 10 мс. Збереження даних, об’ємом до 5 Кб, до пустого локального сховища повинно займати не більше 10 мс за тих самих характеристик комп’ютера.

Для комфортного використання системи студентами, вона має вміти зберігати та зчитувати дані з зовнішнього середовища, як то файлова система. Щоб студенти могли демонструвати свої результати, зроблені вдома, під час навчального процесу, система має вміти зберігати результати в файл, який потім ця ж система, або її копія має вміти зчитувати та відтворювати збережені дані. Збережений файл має містити лише дані, необхідні для відтворення прогресу для мінімізації його розміру.

Аби випадково не втратити весь прогрес, система повинна перепитати користувача чи впевнений він що хоче закрити сторінку з конструктором якщо дані були змінені та не експортовані.

Для комфортного використання системи викладачами, система має надавати можливість імпортувати набір юзер-сторі у вигляді файлу, що містить необхідні для цього дані. Файл має бути один та редагуватись будь-яким текстовим редактором для полегшення взаємодії з системою.

Для подальшого друку, або перегляду даних не використовуючи систему, необхідна можливість експортувати дані у вигляді відформатованого файлу формату PDF.

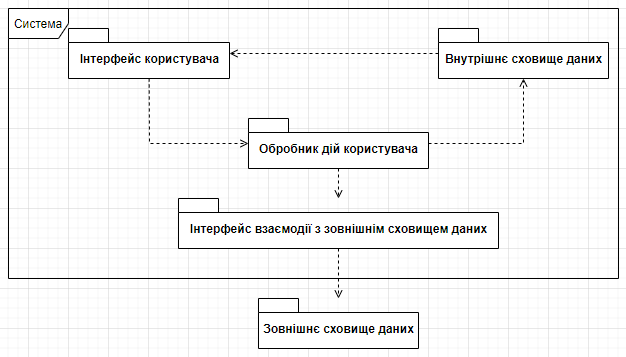
Система має працювати в віртуальному контейнері Docker [7] для розгортання її на комп’ютері з будь-якою операційною системою, що підтримує Docker.

## 3 ПРОЕКТУВАННЯ ІНТЕРАКТИВНОГО НАВЧАЛЬНОГО КОНСТРУКТОРУ СПРИНТІВ

### 3.1 Архітектура системи

Для будування будь-якої програмної системи необхідно спроектувати її архітектуру. Це допоможе забезпечити більшу прогнозованість подальшої розробки програмного продукту.

### 3.1.1 Компоненти системи. Система складається з трьох основних компонент: інтерфейс користувача, компонент сховища даних, компонент модифікування беклогу. Компонент модифікування беклогу має внутрішній стан та реалізує основну логіку роботи конструктора. На рис. 3.1 за допомогою діаграми пакетів UML [8] показані відношення між компонентами системи.



###### Рисунок 3.1 – Діаграма пакетів системи інтерактивного навчального конструктору спринтів

За відношенням пакетів та їх смисловим значенням, можна побачити, що архітектура системи схожа на класичний архітектурний патерн MVC [9].

Пакет «Інтерфейс користувача» відповідає за візуальну репрезентацію внутрішнього стану системи. Він реагує на зміни, що відбуваються після дій користувача.

Пакет «Обробник дій користувача» відповідає за внутрішню логіку системи. Саме в ньому перевіряються дані та дії користувача на валідність, визначається, коли та які застереження будуть виводитися користувачу, віддаються команди на модифікацію внутрішнього стану системи після дій користувача.

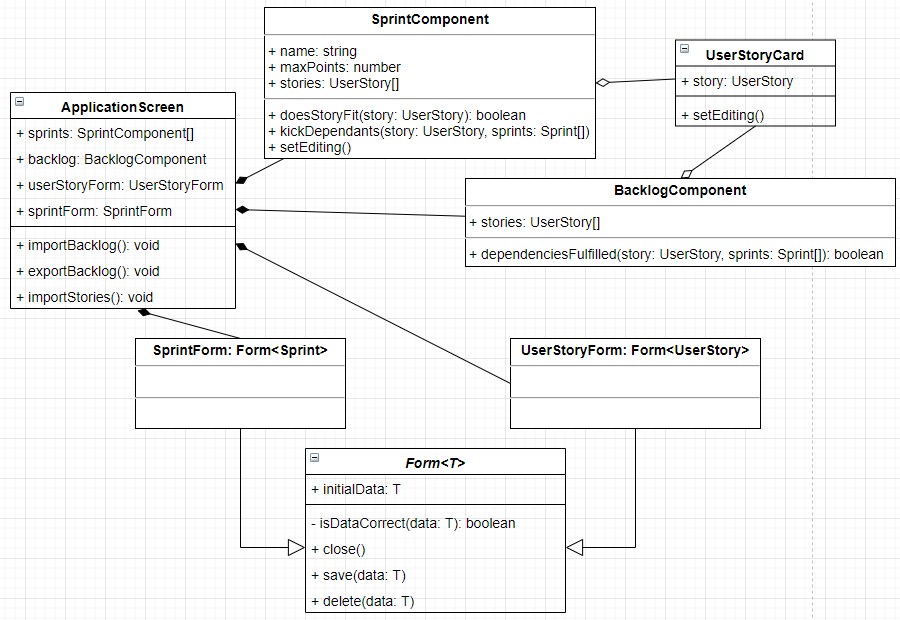
Пакет «Внутрішнє сховище даних» відповідає за внутрішній стан системи. Він реагує на команди, що надаються обробником дій користувача та модифікує внутрішній стан певним чином.

Пакет «Інтерфейс взаємодії з зовнішнім сховищем даних» відповідає за взаємодію з файловою системою користувача. Саме він надає інструменти зчитування та збереження даних до зовнішньої файлової системи.

Тепер, коли були розібрані шари логіки системи, варто привести діаграму класів UML [10], що опише які саме візуальні компоненти системи існують (рис. 3.2).

Хоча логіка обробника дій користувача і візуальні компоненти відокремлені один від одного, для простоти сприйняття, у компонентів наведені методи, що насправді містяться у відповідних контролерах. Тобто, якщо у класу SprintComponent на діаграмі є метод doesStoryFit, це означає що насправді він міститься у SprintComponentController, а сам візуальний компонент лише має посилання на контроллер та визиває відповідний метод коли це потрібно.

Клас ApplicationScreen є кореневим компонентом, що містить у собі інші компоненти і відображає їх.



###### Рисунок 3.2 – Діаграма класів візуальних компонентів системи

Клас SprintForm та UserStoryForm представляють собою модальні вікна, що з’являються коли потрібно редагувати або створити відповідні сутності даних.

Клас SprintComponent представляє собою компонент спринту та містить у собі юзер-сторі.

Клас BacklogComponent є компонентом, що містить список спринтів, що не були розташовані ні в одному з спринтів.

Клас UserStoryCard це компонент, що представляє юзер-сторі у системі. Він може міститися в SprintComponent та BacklogComponent

### 3.1.2 Вибір реалізації сховища даних. Через те, що система повинна зберігати та зчитувати дані з зовнішнього середовища, необхідно обрати певну реалізацію сховища даних.

Існують різні підходи до збереження даних, розберемо найпопулярніші з них, а саме реляційні (SQL) бази даних, документо-орієнтовані (NoSQL) бази даних, збереження даних в текстові файли певних форматів, наприклад, JSON або CSV.

**Реляційна база даних** – база даних, заснована на реляційній моделі даних (модель даних що описує відношення між сутностями). Для роботи з реляційними БД застосовують реляційні СКБД. Інакше кажучи, реляційна база даних – це база даних, яка сприймається користувачем як набір нормалізованих відношень різного ступеня [11].

У реляційних базах даних всі дані представлені у вигляді простих таблиць, розбитих на рядки і стовпці, на перетині яких розташовані дані. Запити до таких таблиць повертають таблиці, які самі можуть ставати предметом подальших запитів. Кожна база даних може включати кілька таблиць.

Переваги реляційної моделі:

* простота і доступність для розуміння користувачем. Єдиною використовуваною інформаційною конструкцією є таблиця;
* суворі правила проектування, які базуються на математичному апараті;
* ACID (Atomicity, Consistency, Isolation, Durability) – це набір властивостей, що гарантують надійну роботу транзакцій бази даних: атомарність, узгодженість, ізольованість, довговічність.

Недоліки реляційної моделі:

* не завжди предметна область може бути представлена у вигляді таблиць, або це представлення може бути незручним для подальшого використання;
* в результаті логічного проектування з'являється множина таблиць. Це призводить до труднощів розуміння структури даних;
* проектування таблиць та зв’язків між ними часто трудомісткий процес, що потребує певного часу та знань;
* для організації запитів і написання прикладного ПЗ, необхідно користуватись інструментами взаємодії з БД, часто треба створювати механізм переводу внутрішньої моделі стану ПЗ до таблиць БД та навпаки.

**Нереляційні бази даних** – бази даних що зберігають дані без використання таблиць [12]. Замість таблиць, вони використовують такі структури даних як документи, формат ключ-значення, графи та інші.

Часто вони використовуються для зберігання величезної кількості складних та різних за структурою даних. Наприклад, для зберігання даних про покупця, в яких зберігається вся його інформація, як то ім’я, адреса, список замовлень та інформація кредитної картки.

Переваги нереляційних баз даних:

* можливість зберігання великих обсягів неструктурованої інформації. У NoSQL немає обмежень на типи даних, що зберігаються, а при необхідності можна додавати нові типи даних;
* NoSQL-бази краще піддаються масштабуванню. Хоча масштабування підтримується і в SQL-базах, це вимагає набагато більших витрат людських і апаратних ресурсів;
* ключові переваги NoSQL баз в розподілених системах полягають в процедурах шаринга і реплікації. Реплікація – це копіювання оновлених даних на інші сервера. Це дозволяє домогтися більшої відмовостійкості і масштабованості системи. Шаринг – це поділ інформації за різними вузлів мережі;

Недоліки нереляційних баз даних:

* застосування сильно прив'язується до конкретної СУБД. Мова SQL універсальна для всіх реляційних сховищ, тому в разі зміни СУБД не доведеться переписувати весь код;
* обмежена ємність вбудованого мови запитів. SQL має тривалу історію і багато стандартів. Це дуже потужний і складний інструмент для операцій з даними і складання звітів. Практично всі мови запитів і методи API сховищ NoSQL були створені на основі тих чи інших функцій SQL, як наслідок, вони мають значно меншу функціональність;
* відсутність ACID властивостей через шаринг та реплікації. З’являється таке поняття як можлива узгодженість (eventual consistency) – коли дані, щоб узгодитись між усіма серверами, потребують деякий час.

**Збереження даних в текстовому файлі формату JSON або CSV** не має функціональних переваг перед розглянутими раніше рішеннями. Перш за все, це текстовий файл. Проте, факт того що немає вбудованих функціональних переваг не означає що переваг взагалі немає.

Переваги збереження даних в файлі:

* немає необхідності створювати окремий сервер, або сервіс для бази даних. Так як дані зберігаються в текстовому файлі, для його редагування та підтримки не треба спеціалізованих інструментів;
* легкість транспортування. Файл можна переносити будь-яким зручним для користувача способом. Дана перевага особливо важлива в навчальному процесі, бо результати роботи в конструкторі необхідно чином демонструвати викладачеві;

Недоліками можна назвати відсутність функціональних переваг баз даних: механізмів транзакцій, обмеження, механізмів масштабованості.

Для того аби обрати найбільш зручне рішення сховища даних, необхідно нагадати для яких функціональних вимог воно потребується. Їх всього дві, розглянемо кожну з них.

1. Необхідно щоб система могла зчитувати зі сховища юзер-сторі та завантажувати їх у беклог.

Сховище даних має легко переноситися, щоб користувач міг завантажити його, наприклад, на флеш-накопичувач та вивантажити на іншому комп’ютері, або мобільному пристрої.

1. Необхідно, щоб система могла експортувати стан беклогу до файлу для подальшого його імпортування.

Як і у попередньому випадку, потрібна мобільність.

Серед розглянутих рішень для сховища даних, для реалізації функціоналу навчального інтерактивного конструктору, підходить текстовий файл. Реляційні та нереляційні бази даних мають переваги, що не потрібні в даному випадку, а їх недоліки у вигляді низької мобільності при переносі та потреби в зайвому програмному шарі для взаємодії з ними, взагалі роблять їх поганим рішенням.

Формат текстового файлу теж варто обрати. Розглянемо згадані вище варіанти CSV та JSON.

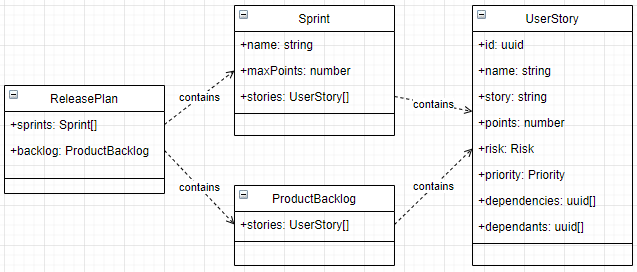
Для імпорту набору юзер-сторі з файлу більше пасує CSV формат, через той факт, що він має фіксовану структуру даних [13], якою і є юзер-сторі. Структура даних описується в вигляді заголовку один раз на початку файлу. Використання JSON, натомість, було б зайвим, оскільки його незручно створювати та редагувати вручну – треба кожен раз описувати властивості об’єкту, а також знати синтаксис формату.

Файл формату CSV можна створювати як у текстовому редакторі так і у редакторі таблиць Microsoft Excel та Google Sheets. Для того аби це зробити засобами Microsoft та Google, треба експортувати таблицю в форматі CSV.

Для імпорту або експорту стану беклогу до файлу найкращим варіантом було б використання формату JSON. Він дозволяє описати структуру об’єкту будь-якої складності в форматі документу [14], чим нагадує NoSQL рішення. А так як стан системи буде зберігатися у вигляді одного об’єкту, зберегти його у JSON-файл та відтворити стан з нього буде досить легко. Також, згенерований файл JSON створений для зручного читання людиною, тому прочитати та змінити його можна буде навіть поза використання конструктору.

### 3.2 Проектування структури даних для збереження та відтворення беклогу

Беклог представляє собою набір юзер-сторі, розподілених по спринтам та беклогу продукту. Його стан зберігається в системі у вигляді об’єктів та відношень між ними (рис 3.3). Саме цей стан і буде зберігатися в файлі JSON для імпорту та експорту.



###### Рисунок 3.3 – Діаграма класів моделей стану беклогу

На діаграмі класів (рис. 3.3) показані 3 класи: Sprint, ReleasePlan, ProductBacklog та UserStory. Далі буде детальний опис представлених класів.

**UserStory** – це клас, що описує юзер-сторі, тобто базову одиницю беклогу. Саме з юзер-сторі та їх розташування планується спринт та беклог в цілому. Клас UserStory має такі поля як:

* іd – ідентифікатор в форматі UUID;
* name – назва юзер-сторі в форматі string;
* story – поле в форматі string, що містить текстове представлення юзер-сторі, зазвичай починається з “As a user I…”;
* points – кількість сторі-поінтів юзер-сторі типу даних number;
* risk – ризик, зберігається в вигляді перелічуваного типу даних, що має значення LOW, MEDIUM, HIGH – низький, середній та високий відповідно;
* priority – пріоритет юзер-сторі, також є перелічуваним типом даних та має значення LOW, MEDIUM, HIGH – низький, середній та високий відповідно;
* dependencies – залежності даної юзер-сторі. Зберігається в форматі UUID та містить ідентифікатори інших юзер-сторі;
* dependants – юзер-сторі, що залежать від даної юзер-сторі. Зберігається в форматі UUID та містить ідентифікатори інших юзер-сторі.

**Sprint** – це клас, що описує спринт беклогу. Він може містити у собі юзер-сторі та має обмеження за кількістю збережених сторі-поінтів. Клас Sprint має наступні поля:

* name – назва спринту. Зберігається в форматі string;
* maxPoints – максимальна кількість збережених сторі-поінтів. Формат даних number;
* stories – юзер-сторі, що містяться у спринті. Поле має формат масив UserStory.

**ProductBacklog** – клас, що містить у собі нерозподілені юзер-сторі. Він має лише одне поле stories, формату масив UserStory.

**ReleasePlan** – клас, в якому зберігаються спринти беклогу та беклог продукту. В нього є поле sprints, формату масив Sprint.

Зважаючи на описані вище моделі системи, наведемо приклад структури файлу експорту та імпорту беклогу у форматі JSON на рис. 3.4.

|  |
| --- |
| {  “sprints”: [  {  “name”: “Sprint 1”,  “maxPoints”: 20,  “stories”: [  {  “id”: “238d6abd-cda6-45d1-89c1-d8066b6de087”,  “name”: “Sample user story”,  “story”: “AS a user I want something”,  “points”: 3,  “risk”: Risk.MEDIUM,  “priority”: Priority.HIGH,  “dependencies”: [],  “dependants”: [“897940ff-1fb7-4b21-989e-3e337b2f481c”]  }  ]  }  ],  “backlog”: {  “stories”: [  {  “id”: “897940ff-1fb7-4b21-989e-3e337b2f481c”,  “name”: “Backlog Story”,  “story”: “As a user I don’t want anything”,  “points”: 13,  “risk”: Risk.HIGH,  “priority”: Priority.LOW,  “dependencies”: [“238d6abd-cda6-45d1-89c1-d8066b6de087”],  “dependants”: []  }  ]  }  } |

###### Рисунок 3.4 – Приклад структури файлу експорту та імпорту беклогу у форматі JSON

Наведена на рис. 3.4 структура даних передбачає виконання усіх функціональних та нефункціональних вимог до системи: юзер-сторі, їх ризики, пріоритет, сторі-поінти та їх обмеження, залежності юзер-сторі, спринти та беклог з їх відповідними подробицями.

### 3.3 Проектування структури даних для імпорту юзер-сторіс до беклогу

Серед функціональних вимог до системи існує необхідність, щоб система зчитувала дані, що містять опис юзер-сторі, та додавала їх до беклогу продукту. Для реалізації цієї вимоги необхідно спершу описати структуру даних, яка необхідна для імпорту в систему.

Форматом текстового сховища даних було обрано CSV, тому приклади структури файлу будуть описані саме у цьому форматі.

В підрозділі 3.2 було описано структуру стану системи. Оскільки юзер-сторі плануються до імпорту в існуючий стан системи, структура файлу імпорту має бути максимально схожа до моделі юзер-сторі в системі.

Структура файлу має містити усі поля, що містяться в моделі юзер-сторі в системі, однак дещо повинно відрізнятися. А саме поля id, бо вони мають відрізнятися від існуючих полів id в системі, тому мають генеруватися системою на момент імпорту даних. Поле id застосовується в моделі юзер-сторі декілька разів: саме поле id, поле dependencies, що є переліком полів id інших об’єктів, та поле dependants, що також є переліком полів id інших об’єктів.

Оскільки генерувати uuid ідентифікатори вручну незручно, а передбачається, що дані для імпорту мають створюватися вручну, формат ідентифікатору для даних для імпорту варто змінити на числовий.

Окрім поля id, у структурі файлу імпорту буде відсутнє поле dependants. Воно присутнє в структурі стану системи для підвищення швидкодії та не потрібне при формуванні даних до імпорту людиною.

Запис масиву залежностей буде представлено в форматі чисел id, записаних через кому.

Беручи до уваги зазначені вище особливості, опишемо структуру файлу для імпорту юзер-сторіс у систему в табл. 3.1.

##### Таблиця 3.1 – Структура файлу CSV для імпорту юзер-сторіс в беклог

|  |  |  |
| --- | --- | --- |
| **Назва поля** | **Тип даних** | **Призначення** |
| іd | integer | Ідентифікатор юзер-сторі, унікальний для файлу імпорту |
| name | String | Назва юзер-сторі |
| story | String | Зміст юзер-сторі |
| points | Number | Кількість поінтів юзер-сторі |
| risk | Risk | Ризик, який несе в собі юзер-сторі |
| priority | Priority | Пріоритет виконання юзер-сторі |
| dependencies | integer[] | Перелік ідентифікаторів юзер-сторі від яких залежить дана юзер-сторі |

За приведеною вище в табл. 3.1 структурою файлу з даними імпорту юзер-сторі до беклогу, наведемо приклад вмісту файлу (рис. 3.5), який можливо імпортувати до системи.

|  |
| --- |
| id,name,story,points,risk,priority,dependencies  1,Log in,As a user I want to be able to log in,3,LOW,HIGH,  2,Registration,As a user I want to be able to register,3,LOW,HIGH,"1, 3"  3,Database instantiation,As a user I want to store data about myself,4,MEDIUM,HIGH, |

###### Рисунок 3.5 – Приклад вмісту файлу імпорту юзер-сторі у беклог

В файлі імпорту значення мають розділятися комами, бо це формат за замовчуванням у Google Sheets та Microsoft Excel. Якщо в якихось значеннях полів зустрічається кома, усе значення потрібно виділити в лапки, а самі лапки, якщо потрібно помістити у значення поля, виокремлюються за допомогою тих самих лапок, тому якщо значенням поля буде «”1,1”», потрібно записувати його в вигляді «"""1,1"""».

### 3.4 Проектування інтерфейсу користувача

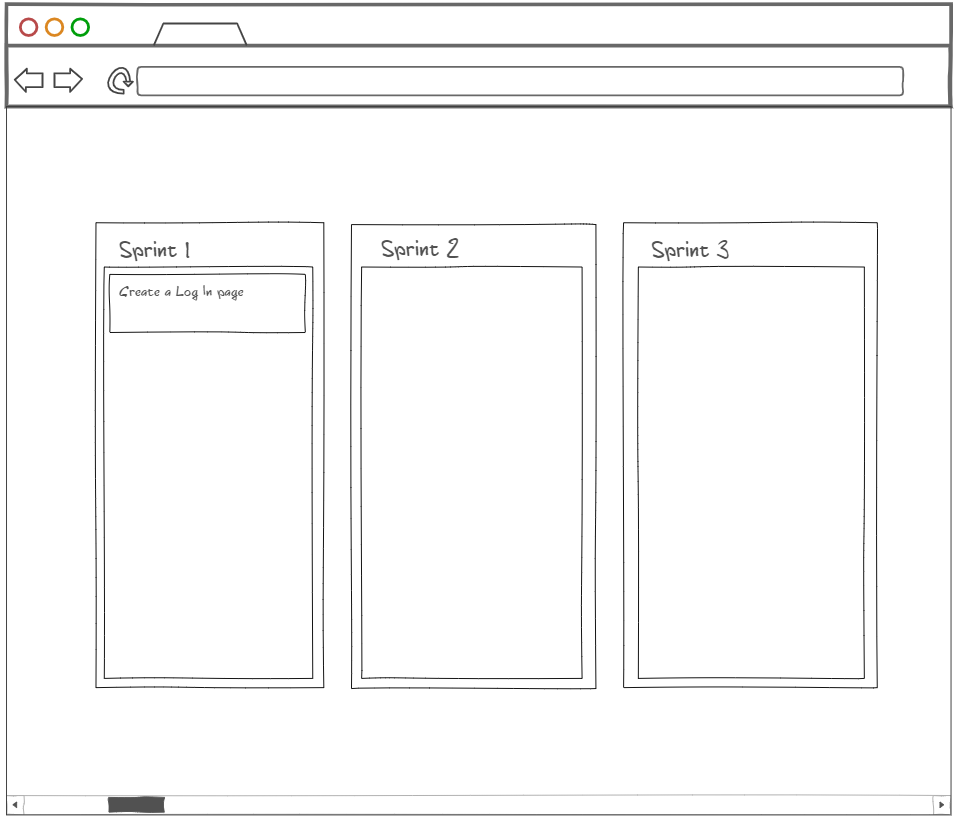
Інтерфейс користувача – важлива частина програмного продукту від якої залежить те, наскільки зручно та ефективно користувач зможе взаємодіяти з системою.

### 3.4.1 Вибір формату компонування функціональних елементів. Інтерфейс користувача інтерактивного навчального конструктору виконано в форматі Single Page Application – односторінкового застосунку [15]. Такий формат має переваги перед багатосторінковим за деякими аспектами:

* він переважно швидше завантажується через те, що усі візуальні компоненти знаходяться на одній сторінці;
* UX (досвід користувача) дозволяє втрачати фокус з контенту сторінки на другорядні елементи такі як налаштування, бо контент залишається на тій самій сторінці, що позитивно впливає на процес планування беклогу;
* односторінкові веб-застосунки є сучаснішими за багатосторінкові як за дизайном так і за технологіями, що може вплинути на естетичні вподобання студента, так як люди часто віддають перевагу застосунку з більш сучасним дизайном.

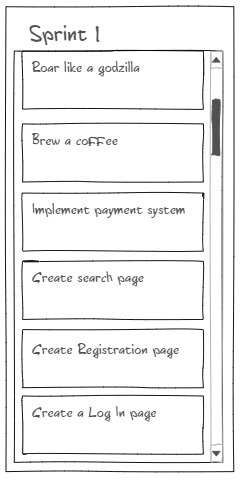
Проте, навіть незважаючи на переваги описані вище, доцільніше використовувати односторінковий формат через малу кількість функціоналу, що можна зручно вмістити на одну сторінку.

### 3.4.2 Проектування головної сторінки. Після того як ми обрали формат веб-застосунку, настала черга створення інтерфейсу головної сторінки. Проаналізувавши існуючі аналоги системи, ями дійшли висновку, що краще за все для планування спринтів підходить формат компонування інтерфейсу, який показано на рис. 3.6.



###### Рисунок 3.6 – Розташування елементів інтерфейсу для планування беклогів спринтів

Такий формат компонування елементів інтерфейсу дозволяє використати як вертикальну область екрану, так і горизонтальну. При створенні більшої кількості спринтів повзунок знизу дозволить відобразити до трьох спринтів одночасно в будь-якому горизонтальному положенні сторінки. Якщо ж необхідно буде вмістити велику кількість юзер-сторі на один спринт, всередині нього з’явиться вертикальний повзунок, як на рис. 3.7.



###### Рисунок 3.7 – Приклад вертикального повзунка у спринту, в якому велика кількість юзер-сторі

Сформований приклад інтерфейсу дозволяє відобразити частину функціональних вимог, що сформовані до конструктору, проте треба розмістити ще список юзер-сторі беклогу та функціонал, що дозволяє маніпулювати даними через їх імпорт та експорт та їх створенням. На рис. 3.8 представлений остаточний вигляд мокапу інтерфейсу з виконанням майже усіх функціональних вимог до продукту. Не враховані лише можливості для зміни даних спринтів та юзер-сторі через те, що цей функціонал буде реалізовано модальними вікнами, що будуть показані далі.



###### Рисунок 3.8 – Мокап інтерфейсу програми зі списком беклогу продукту та випадаючими меню «Створити новий…» та «Імпорт/Експорт»

На рис. 3.8 інтерфейс умовно поділений на дві функціональні частини: ліва частина, на якій розташований список спринтів та юзер-сторі (майбутній беклог спринтів) та права частина, на якій можна побачити колонку беклогу продукту, в якій містяться ще не розподілені юзер-сторі та функціональні випадаючі меню – «Створити новий…» та «Імпорт/Експорт».

Для переміщення юзер-сторі до беклогу продукту або спринтів, користувачу потрібно буде навести курсор на юзер-сторі, затиснути ліву кнопку миші та перенести картку куди йому потрібно. При цьому, система зеленим кольором фону буде показувати куди користувач може перенести картку, а червоним – куди не може через технологічні обмеження (досягнутий максимум сторі-поінтів).

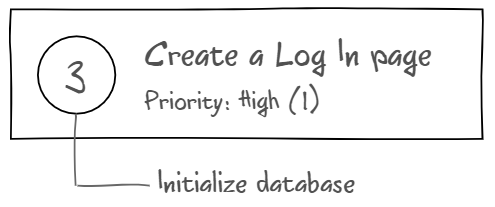
### 3.4.3 Проектування інтерфейсу з інформацією про юзер-сторі. Юзер-сторі представлені у вигляді карточок, наповнених короткою інформацією про них. На рисунку 3.9 можна детально розглянути яка інформація на них міститься.



###### Рисунок 3.9 – Детальний вигляд картки юзер-сторі

Інформацію, що міститься на картці юзер-сторі, можна поділити на заголовок («Create a Log In page»), кількість сторі поінтів (цифра «3» зліва від тексту), пріоритет («Priority: High (1)»). Існує ще одна важлива частина інформації, що важко показати чорно-білим, а саме ризик. Він відображається кольором фону картки: низький ризик – зеленим, помірний – жовтим, високий ризик – червоним.

Якщо у юзер-сторі є залежності – вони відображаються в беклозі у вигляді шляху (рис. 3.10), що йде до назви юзер-сторі, від якої вона залежить.



###### Рисунок 3.10 – Відображення юзер-сторі, що має залежності

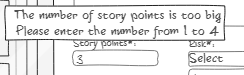
Вміст юзер-сторі на карточці опущено. Його можна прочитати натиснувши на картку. Тоді з’явиться модальне вікно, приклад якого можна побачити на рис. 3.11.



###### Рисунок 3.11 – Модальне вікно з детальною інформацією про юзер-сторі

Окрім відображення детальної інформації про юзер-сторі, модальне вікно несе в собі фунціонал редагування інформації про юзер-сторі. Так, якщо змінити текст або іншу інформацію та натиснути кнопку «Save», змінені дані юзер-сторі оновляться всередині конструктору.

Якщо введена інформація для збереження буде некоректною (наприклад, кількість сторі поінтів юзер-сторі занадто велика) – система не дасть зберегти інформацію та покаже користувачеві (рис. 3.12) в тому ж вікні на формі, які дані потрібно виправити для збереження.

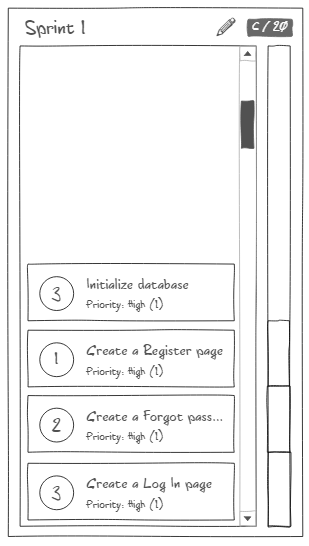


###### Рисунок 3.12 – Система відображає користувачеві які дані потрібно виправити для збереження юзер-сторі

Для того, аби закрити модальне вікно, можна натиснути будь-де за межами вікна, або на хрестик, що розташований справа зверху. Аби видалити юзер-сторі, треба натиснути на кнопку з позначкою сміттєвої корзини зліва від кнопки «Save». Після цього з’явиться модальне вікно що запросить підтвердження видалення.

### 3.4.4 Проектування інтерфейсу з інформацією про спринт. На сторінці конструктору спринт представлений у вигляді стовпцю, який містить у собі картки юзер-сторі. Окрім юзер-сторі він також містить назву та максимальну кількість сторі поінтів, що він може містити.

Поміж інформації, що зберігається в поточному сховищі даних застосунку, компонент спринту відображає також і агреговану інформацію як: кількість поточних сторі поінтів, що є сумою сторі поінтів юзер-сторі, який він містить, а також шкалу ризиків – стовпець, що фарбується в кольори ризиків, що містяться в юзер-сторі, прямо пропорційно до їх кількості. Детальне представлення компоненту спринту можна побачити на рис. 3.13.



###### Рисунок 3.13 – Детальне представлення компоненту спринту

На рис. 3.13 зверху розташовані назва спринту, кнопка редагування, індикатор заповненості (6 із 20 можливих сторі поінтів заповнено). Зліва розташований список юзер-сторі, що містяться у спринту, а справа шкала ризиків. Шкала містить у собі максимальну кількість сторі поінтів спринту, а три можливі стовпчики всередині неї відображають ризикованість спринту. Так, на рис. 3.13 знизу вверх йдуть зелений стовпчик (3 сторі поінти) – низький ризик, жовтий стовпчик (3 сторі поінти) – середній ризик і червоний стовпчик (3 сторі поінти) – великий ризик. Таке відображення дозволить користувачеві легше розуміти наскільки ризикованим може бути запланований спринт.

Аналогічно з юзер-сторі, інформацію про спринт також можна редагувати. Для цього потрібно натиснути на олівець біля індикатору заповненості сторі-поінтами. Після натискання, з’явиться модальне вікно як на рис. 3.14.



###### Рисунок 3.14 – Модальне вікно редагування інформації про спринт

Коли користувач редагує спринт, він може вказати потрібну йому інформацію, зберегти зміни, або видалити його. У випадку видалення, користувачу потрібно буде підтвердити своє бажання. Як і в випадку з редагуванням юзер-сторі, щоб закрити модальне вікно, користувачу потрібно буде натиснути або на хрестик, або за межами вікна.

### 3.4.5 Проектування меню. Над списком юзер-сторі в беклозі продукту містяться випадаючі меню «Додати новий…» та «Імпорт/Експорт». Ці пункти меню дозволяють скористуватися рештою функціоналу конструктору. Розглянемо їх докладніше.

Якщо натиснути на випадаюче меню «Додати новий…», користувач побачить варіанти сутностей (рис. 3.15), які можна додати до беклогу.



###### Рисунок 3.15 – Варіанти сутностей, що можна додати до беклогу

Якщо він натисне будь-який пункт меню, з’явиться відповідне модальне вікно, які були описані на рис. 3.11 та 3.14. Дані у формі заповнені не будуть, а коли користувач їх заповнить та натисне кнопку «Зберегти», відповідна сутність буде додана до беклогу: якщо спринт – пустий спринт справа від створених, а якщо юзер-сторі – картка буде поміщена в колонку беклогу продукту.

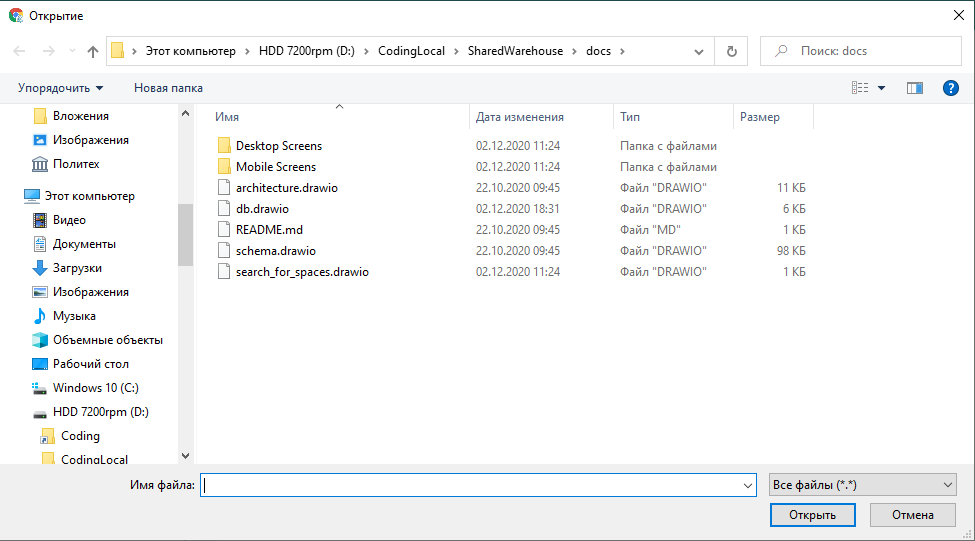
У випадку коли користувач натисне на меню «Імпорт/Експорт», йому буде запропоновано обрати одну дію з трьох, що показані на рис. 3.16.



###### Рисунок 3.16 – Випадаюче меню Імпорту/Експорту

Користувач може обрати «Зберегти як…», «Завантажити з…», «Експортувати у вигляді PDF» та «Імпорт до беклогу…».

Перша дія («Зберегти як…») запропонує користувачу обрати місце та назву файлу для збереження на своєму носієві даних (рис. 3.17). Файл буде мати розширення .json та може бути використаний для подальшого завантаження у конструктор для відновлення прогресу.



###### Рисунок 3.17 – Система пропонує користувачу обрати шлях до файлу

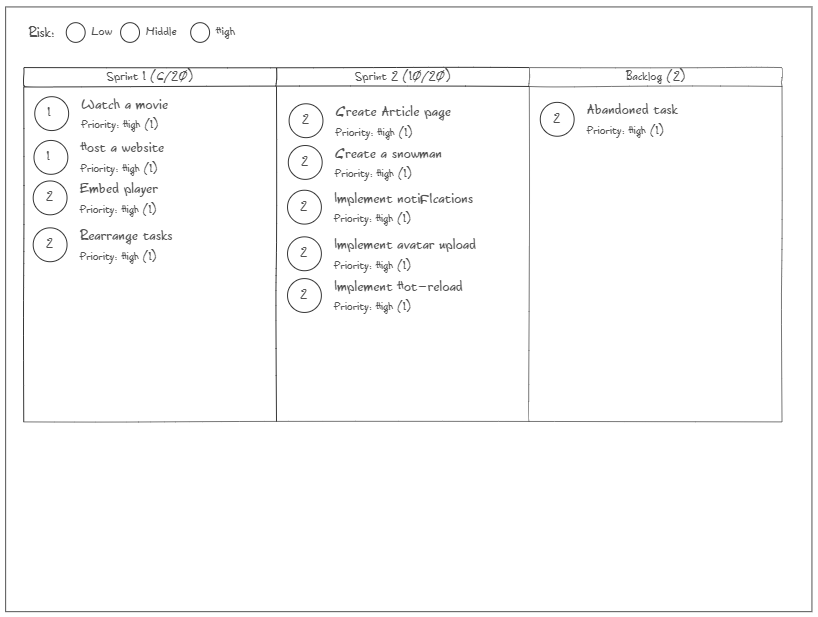
Якщо користувач обирає пункт «Завантажити з…», йому буде запропоновано обрати шлях до файлу (рис. 3.17), який він, або інший користувач, мав згенерувати дією «Зберегти як…» для завантаження прогресу, що збережений у файлі до конструктору.

У випадку, коли користувач натискає «Експортувати у вигляді PDF», система запропонує йому обрати шлях (рис. 3.17), де він захоче зберегти файл, що буде містити у собі звіт про сформовані спринти та беклог.

Коли буде натиснута кнопка «Імпортувати до беклогу…», користувачу буде запропоновано обрати (рис. 3.17) csv файл, що містить у собі список юзер-сторі, які будуть додані до беклогу проекту.

### 3.5 Проектування дизайну вмісту PDF файлу експорту

Коли користувач експортує до PDF поточну інформацію з конструктору, генерується файл, який містить у собі усю інформацію, що зберігається в конструкторі на момент експорту. Компонування елементів на згенерованій сторінці показані на рис. 3.18.



###### Рисунок 3.18 – Компонування інформації з конструктору у файлі експорту PDF

На сторінці файлу зверху розташовані умовні позначення ризику юзер-сторі. Кольором ризик визначається аналогічно до кольорів на рис. 3.8. Всередині файлу експорту кольором виділені кола, на яких містяться сторі-поінти.

Спринти та беклог в файлі експорту відображені як таблиця з заголовком, на якому відображаються назва спринту або беклогу та кількість поточних та максимальних сторі поінтів, а також вміст, що сформований у вигляді списку юзер-сторі які належать певним спринтум або беклогу продукту.

## 4 РОЗРОБКА ІНТЕРАКТИВНОГО НАВЧАЛЬНОГО КОНСТРУКТОРУ СПРИНТІВ

Після проектування системи була виконана її програмна розробка. Даний розділ присвячений опису використовуваних технологій, а також програмній реалізації вимог до системи.

### 4.1 Вибір стеку технологій та інструментів для розробки

Для швидкої та якісної розробки SPA веб-застосунку було використано JavaScript фреймворк React [16].

В якості бібліотеки готових візуальних компонентів було застосовано Semantic UI [17].

Оскільки необхідно зберігати дані для подальшого їх експорту та імпорту, бажано, щоб було централізоване місце сховища даних. Існує бібліотека-фреймворк Redux, що бере на себе роль імутабельного сховища даних [18]. Воно зберігається як json об’єкт, що дуже зручно для функціоналу експорту та імпорту.

Для поєднання фреймворку React з Redux, скористаємося спеціалізованою обгорткою React Redux [19].

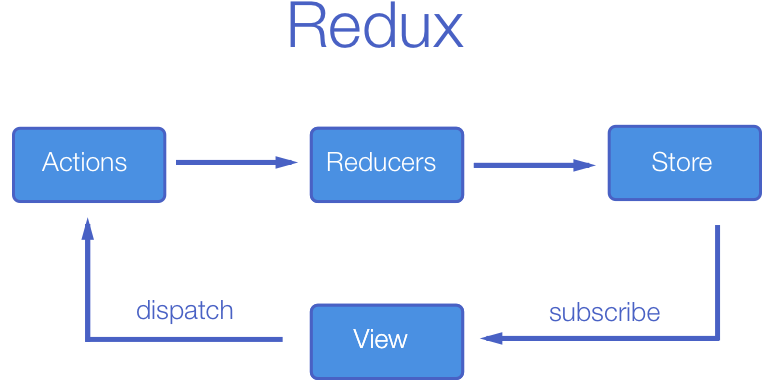
Щоб зчитувати дані з CSV файлу буде використана бібліотека csv-parse для JavaScript [20].

Для тестування скористаємося бібліотеками Jest [21] та React Testing Library [22].

В якості IDE скористаємося безкоштовним Visual Studio Code [23].

### 4.2 Програмна реалізація

При використанні фреймворку Redux, зміна поточного стану системи відбувається через так званий Redux flow [24]. Схема протікання Redux flow приведено на рис. 4.1.



###### Рисунок 4.1 – Схема Redux flow

Store – це сховище даних застосунку. Воно містить усі дані необхідні для відображення у застосунку.

View – це набір візуальних компонентів системи. Вони беруть дані зі Store шляхом підписання (subscribe) на його зміни. Ці компоненти можуть відправляти (dispatch) певні дії (Actions), які проходять через низку функцій, що на схемі називаються Reducers, які, в свою чергу, змінюють Store.

Кожен раз, коли потрібно змінити стан системи, необхідно, щоб компонент системи відправив Action з необхідними даними. Action – це об’єкт з певними параметрами. Він має тип (назву для подальшої обробки) та дані (payload).

Для того, аби втілити логіку роботи системи, були створені певна кількість actions, що описані в табл. 4.1.

##### Таблиця 4.1 – Actions для зміни стану системи

|  |  |  |
| --- | --- | --- |
| **Тип action** | **Дані (payload)** | **Опис** |
| CREATE USER STORY | story: UserStory | Створення нової юзер-сторі та додавання її до беклогу. |
| CREATE SPRINT | sprint: Sprint | Створення нового спринту. |
| UPDATE USER STORY | story: UserStory | Зміна даних існуючої юзер-сторі. |
| UPDATE SPRINT | sprint: Sprint, index: number | Зміна даних існуючого спринту за індексом у масиві. |
| DELETE USER STORY | id: uuid | Видалення юзер-сторі за айді. |
| DELETE SPRINT | index: number | Видалення спринта за індексом в масиві. |
| IMPORT BACKLOG | plan: ReleasePlan | Завантаження прогресу з файлу імпорту. |
| IMPORT STORIES | stories: UserStory | Імпорт юзер-сторі до беклогу з файлу імпорту. |
| MOVE USER STORY TO SPRINT | id: uuid, index: number | Перенесення юзер-сторі до спринту. |
| MOVE USER STORY TO BACKLOG | id: uuid | Перенесення юзер-сторі до беклогу. |
| KICK USER STORIES FROM SPRINTS | ids: uuid[] | Перенесення набору юзер-сторі до беклогу |

Отримані actions в подальшому обробляються в Reducers, що мають бути створені в системі. Reducer – це метод, що приймає на вхід стан системи (store) та action. В залежності від типу та даних action модифікують стан системи, створюючи новий об’єкт стану та повертаючи його.

Окрім зміни стану, що обробляється фреймворком Redux, існує внутрішня логіка системи, пов’язана з перевірками на коректність введених даних, коректність та можливість переносу юзер-сторі до спринту. В табл. 4.2 описані методи, що являють собою внутрішню логіку системи.

##### Таблиця 4.2 – Опис методів, що реалізують внутрішню логіку системи

|  |  |  |  |
| --- | --- | --- | --- |
| **Назва методу** | **Аргументи** | **Вихідні параметри** | **Опис** |
| doesStoryFit | story: UserStory | boolean | Перевіряє, чи юзер-сторі під­ходить до перенесення до спринту |
| kickDependants | story: UserStory,  sprints: Sprint[] | Void | Перевіряє, чи розташовані залежні юзер-сторі в спринтах та викликає зміну стану, що переносить залежні юзер-сторі до беклогу. |
| setEditing | - | Void | Встановлює поточний еле­мент до редагування. |
| dependenciesFulfilled | story: UserStory,  sprints: Sprint[] | boolean | Перевіряє чи всі юзер-сторі від яких залежить юзер-сторі встановлені у спринтах |
| importBacklog | - | Void | Викликає діалог заванта­ження стану беклогу. |
| exportBacklog | - | Void | Викликає діалог виванта­ження стану беклогу до файлу. |

Продовження таблиці 4.2

|  |  |  |  |
| --- | --- | --- | --- |
| **Назва методу** | **Аргументи** | **Вихідні параметри** | **Опис** |
| importStories | - | Void | Викликає діалог імпорту на­бору юзер-сторіс до беклогу |
| isDataCorrect | data: T | boolean | Перевіряє, чи є дані корект­ними до збереження. |
| close | - | Void | Закриває форму редагування. |
| save | data: T | Void | Зберігає дані з форми до си­стеми. |
| delete | data: T | Void | Видаляє дані з системи. |

Методи є занадто тривіальними для їх детального розгляду, проте варто зауважити, що метод doesStoryFit перевіряє, чи має поточний спринт достатню кількість вільних сторі-поінтів, а якщо у юзер-сторі є залежності, перевіряє чи містяться всі залежності в ньому та попередніх спринтах, бо це є умови перенесення юзер-сторі.

Метод kickDependants перевіряє лише поточний спринт та ті, що знаходяться після нього, бо doesStoryFit не дасть змогу встановити залежні спринти раніше того, від кого вони залежать.

Лістинги програм наведені в додатку А.

## 5 ТЕСТУВАННЯ ІНТЕРАКТИВНОГО НАВЧАЛЬНОГО КОНСТРУКТОРУ СПРИНТІВ

Після програмної реалізації системи необхідно провести її тестування. Оскільки основна частина архітектурної реалізації лежить на фреймворці Redux, має сенс провести системне тестування, бо проводити інші види тестування, наприклад, unit-тестування, не дуже доцільно при роботі з візуальними компонентами.

### 5.1 Системне тестування

Системне тестування – це тестування програмного забезпечення, що виконується на повнофункціональній, інтегрованій системі з метою перевірки відповідності системи функціональним та нефункціональним вимогам [25].

Тестування реалізовано в ручному режимі, всі прецеденти тестувалися на перезавантаженій системі, а кроки описані з початкової сторінки конструктору. Прецеденти описані далі.

**Створення юзер-сторі**

*Основний успішний сценарій:*

* 1. Натиснемо на випадаюче меню «Add new» та виберемо юзер-сторі зі списку. Система виводить форму додавання нової юзер-сторі.
  2. Заповнимо форму коректними даними та натиснемо кнопку «Save». Система закриє форму та в колонці беклогу справа з’явиться нова юзер-сторі. *Поведінка коректна*.

**Зміна даних юзер-сторі**

*Основний успішний сценарій:*

1. Натиснемо на юзер-сторі. Система виводить форму зміни даних юзер-сторі.
2. Змінюємо дані, натискаємо кнопку «Save». Система закриває форму зміни даних та оновлює дані в юзер-сторі. *Поведінка коректна.*

*Розширення:*

1а.1. Натискаємо на юзер-сторі що розташована у спринті. Система виводить форму зміни даних юзер-сторі.

1а.2. Змінюємо дані, проте кількість сторі поінтів вказуємо більшу за суму попередніх сторі поінтів і вільних сторі поінтів спринту. Система повідомляє що кількість сторі поінтів занадто велика та необхідно її змінити, система очікує повторного вводу. *Поведінка коректна.*

**Видалення юзер-сторі**

*Основний успішний сценарій:*

1. Натиснемо на юзер-сторі. Система виводить форму зміни даних юзер-сторі.
2. Натиснемо на кнопку з іконкою смітника. Система запитує підтвердження дії.
3. Підтверджуємо. Система видаляє юзер-сторі та дані про неї з усіх юзер-сторі, що від неї залежать та від яких вона залежить. *Поведінка коректна.*

*Розширення:*

3а.1. Не підтверджуємо. Система залишає форму редагування та очікує вводу користувача. *Поведінка коректн*а.

**Додавання спринту**

*Основний успішний сценарій:*

* 1. Натиснемо на випадаюче меню «Add new» та виберемо спринт зі списку. Система виводить форму додавання нового спринту.
  2. Заповнимо форму даними та натиснемо кнопку «Save». Система закриває форму та створює новий спринт. *Поведінка коректна*.

**Зміна даних спринту**

*Основний успішний сценарій:*

1. Натиснемо на кнопку у вигляді олівця справа від назви створеного спринту. Система виводить форму редагування спринту.

2. Змінимо дані та натиснемо кнопку «Save». Система закриває форму та оновлює дані спринту. *Поведінка коректна.*

*Розширення:*

2а.1. Змінимо дані, проте максимальну кількість сторі поінтів встановимо меншу за суму сторі поінтів що містяться у юзер-сторі, які розміщені в даному спринті. Система повідомляє користувачеві про некоректні дані та чекає на повторний ввод.

**Видалення спринту**

*Основний успішний сценарій:*

1. Натиснемо на кнопку у вигляді олівця справа від назви створеного спринту. Система виводить форму редагування спринту.
2. Натиснемо на кнопку з іконкою смітника. Система запитує підтвердження дії.
3. Підтверджуємо. Система видаляє спринт. *Поведінка коректна.*

*Розширення:*

2а. Спринт, що ми намагаємося видалити, містить юзер-сторі.

2а.1. Система повідомляє про неможливість видалити спринт, через те що в ньому містяться юзер-сторі та підказує що спершу їх треба видалити, або перенести. Система чекає на повторний ввод користувача.

**Експорт беклогу**

*Основний успішний сценарій:*

1. Натиснемо на випадаюче меню «Import/Export» та виберемо пункт «Save as…». Система виводить вікно вибору місця та ім’я файлу для експорту.
2. Оберемо необхідне місце та ім’я файлу і натиснемо кнопку «Зберегти». Система закриває вікно вибору та зберігає файл у обраному місці. *Поведінка коректна.*

**Імпорт беклогу**

*Основний успішний сценарій:*

1. Натиснемо на випадаюче меню «Import/Export» та виберемо пункт «Load from…». Система виводить вікно вибору файлу для імпорту.
2. Оберемо файл імпорту беклогу та натиснемо кнопку «Відкрити». Система закриває вікно вибору та замінює поточний стан беклогу на той, що було експортовано у файл.

*Розширення:*

2а. Файл імпорту некоректний, тому система не може його прочитати.

2а.1. Система виводить користувачеві повідомлення про те, що не може прочитати обраний файл.

2а.2. Натискаємо кнопку «ОК». Система пропонує обрати інший файл. *Поведінка коректна.*

**Завантаження набору юзер-сторі у беклог**

*Основний успішний сценарій:*

1. Натиснемо на випадаюче меню «Import/Export» та виберемо пункт «Import to backlog…». Система виводить вікно вибору файлу для імпорту набору юзер-сторі.
2. Оберемо файл та натиснемо кнопку «Відкрити». Система додає юзер-сторі, що були описані у файлі до беклогу. *Поведінка коректна.*

*Розширення:*

2а. Файл імпорту некоректний, тому система не може його прочитати.

2а.1. Система виводить користувачеві повідомлення про те, що не може прочитати обраний файл.

2а.2. Натискаємо кнопку «ОК». Система пропонує обрати інший файл. *Поведінка коректна.*

**Перенесення юзер-сторі у беклог, або спринт**

*Основний успішний сценарій:*

1. Пересуваємо юзер-сторі з беклогу до спринту. Юзер-сторі перенесена до спринту. *Поведінка коректна.*

**Перевірка чи показує система обмеження, що можуть застосовуватися до юзер-сторі при переносі**

Для демонстрації результатів тестування скористаємося табл. 5.1.

##### Таблиця 5.1 – Результати тестування обмежень при переносі юзер-сторі.

|  |  |  |
| --- | --- | --- |
| **Умова** | **Результат** | **Чи коректна поведінка?** |
| Спринт, до якого переноситься юзер-сторі, заповнений | Юзер-сторі не перенесено. Фон спринту виділяється червоним на момент переносу. | Так |
| Юзер-сторі, що переноситься, має залежності, які ще не були розташовані | Юзер-сторі не перенесено. Юзер-сторі неможливо почати переносити, а вона сама висвітлюється ніби вона вимкнена. | Так |

Продовження таблиці 5.1

|  |  |  |
| --- | --- | --- |
| **Умова** | **Результат** | **Чи коректна поведінка?** |
| Юзер-сторі переноситься до спринту, що знаходиться раніше, ніж спринти, у яких розташовані юзер-сторі, від яких залежить дана юзер-сторі | Юзер-сторі не перенесено. Фон спринту виділяється червоним на момент переносу. | Так |

У таблиці наведені тест-кейси, що проводилися із застосунком в ручному режимі. Як бачимо, всі тести завершені успішно.

**Перевірка перепитування намірів, коли користувач намагається закрити сторінку з незбереженими даними**

*Основний сценарій:*

1. Створимо новий спринт.
2. Закриємо сторінку веб-застосунку. Веб-браузер повідомить про те, що існують незбережені зміни. *Поведінка коректна.*

*Розширення:*

2а.1. Експортуємо беклог у файл.

2а.2. Закриємо сторінку веб-застосунку. Сторінка закриється. *Поведінка коректна.*

**Перевірка швидкодії**

Для демонстрації результатів тестування швидкодії скористаємося порівняльною табл. 5.2.

##### Таблиця 5.2 – Порівняльна таблиця з результатами тестування швидкодії

|  |  |  |  |
| --- | --- | --- | --- |
| **Умова** | **Реальний результат** | **Очікуваний результат** | **Успіх?** |
| Переміщуємо юзер-сторі з беклогу до спрінту на віртуальній машині, на яку виділено 1 ядро процесору з тактовою частотою 1.6ГГц | Переміщено за 1мс | Переміщено за 10мс | Так |
| Імпортуємо беклог з файлу, розмір якого становить 9.8Кб | Імпортовано за 124мс | Імпортовано за 300мс | Так |
| Експортуємо до файлу беклог, який складається з одного спринту та 10 юзер-сторі | Експортовано за 235мс | Експортовано за 300мс | Так |

Усі результати, що наведені в таблиці затратили менше часу ніж потрібно, що свідчить про успішність складання тестування.

### 5.2 Інструкція з встановлення

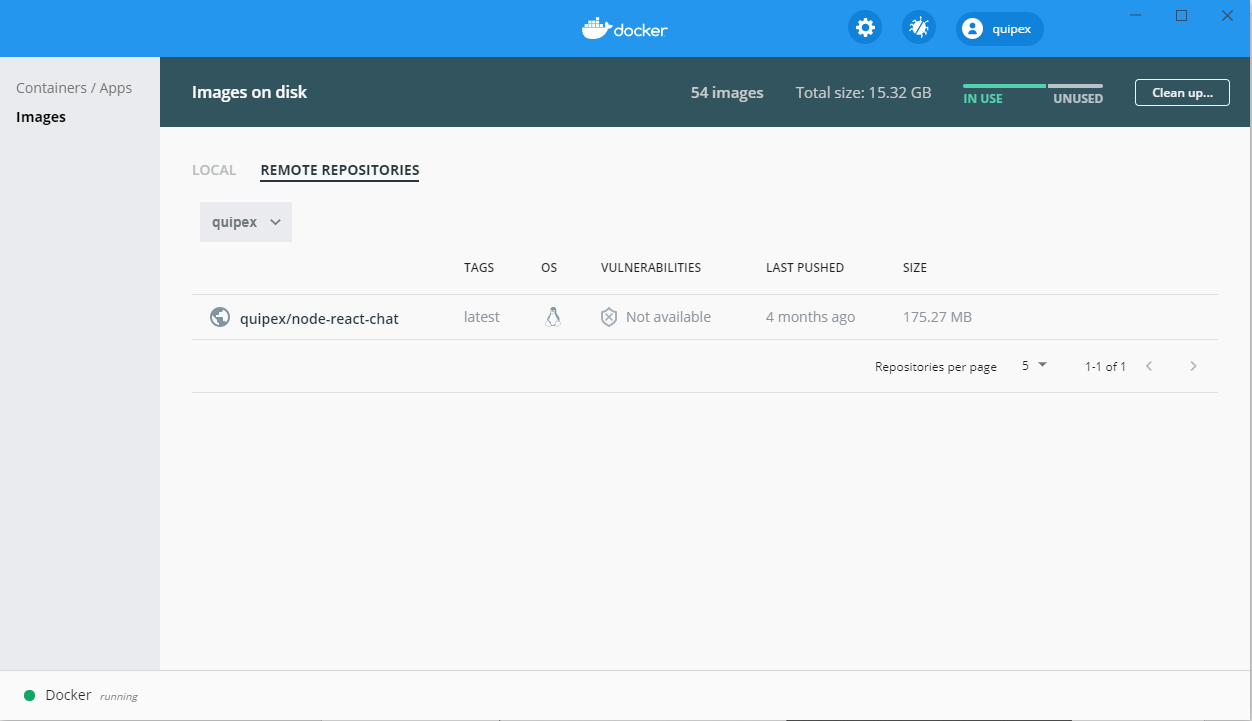
Для встановлення системи треба мати комп’ютер з операційною системою, що підтримує встановлення Docker-контейнерів. Порядок встановлення такий:

1. Встановити Docker.

2. Встановити Docker-image за тегом quipex:backlog\_constructor:latest (рис. 5.1).

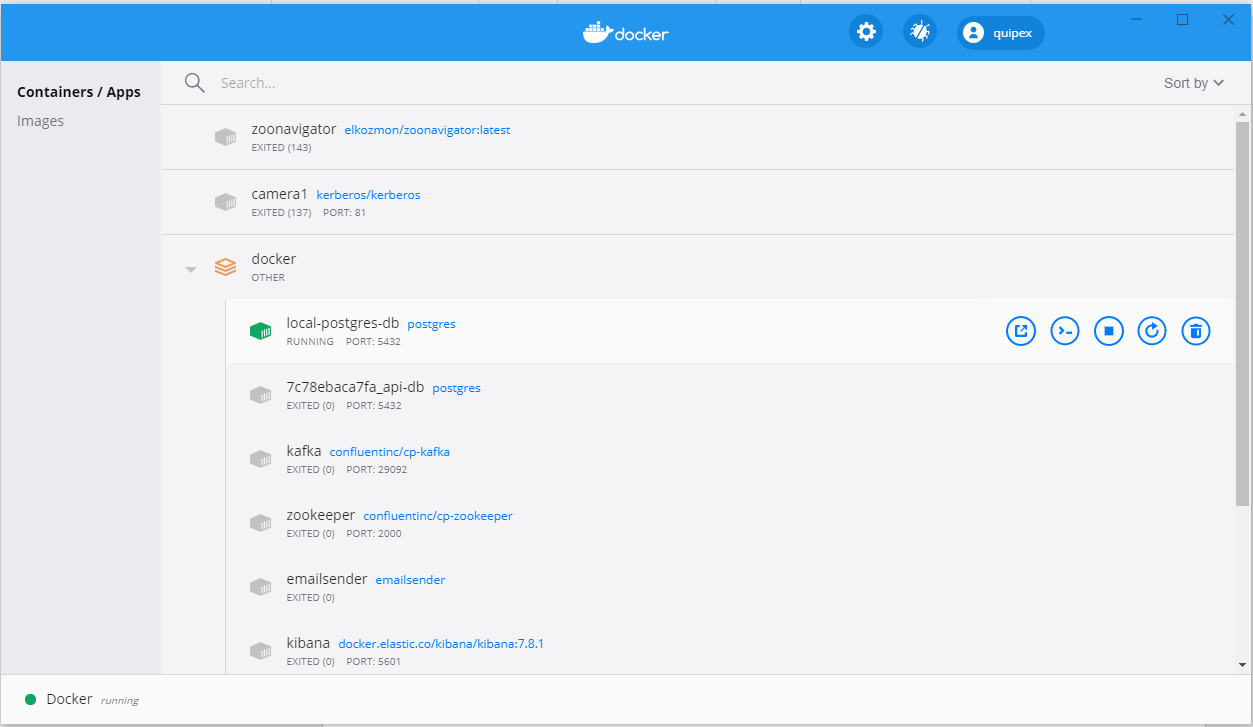
3. Запустити завантажений image у контейнері (рис. 5.2).

4. Перейти до localhost:3000 у веб-браузері (рис. 5.3).



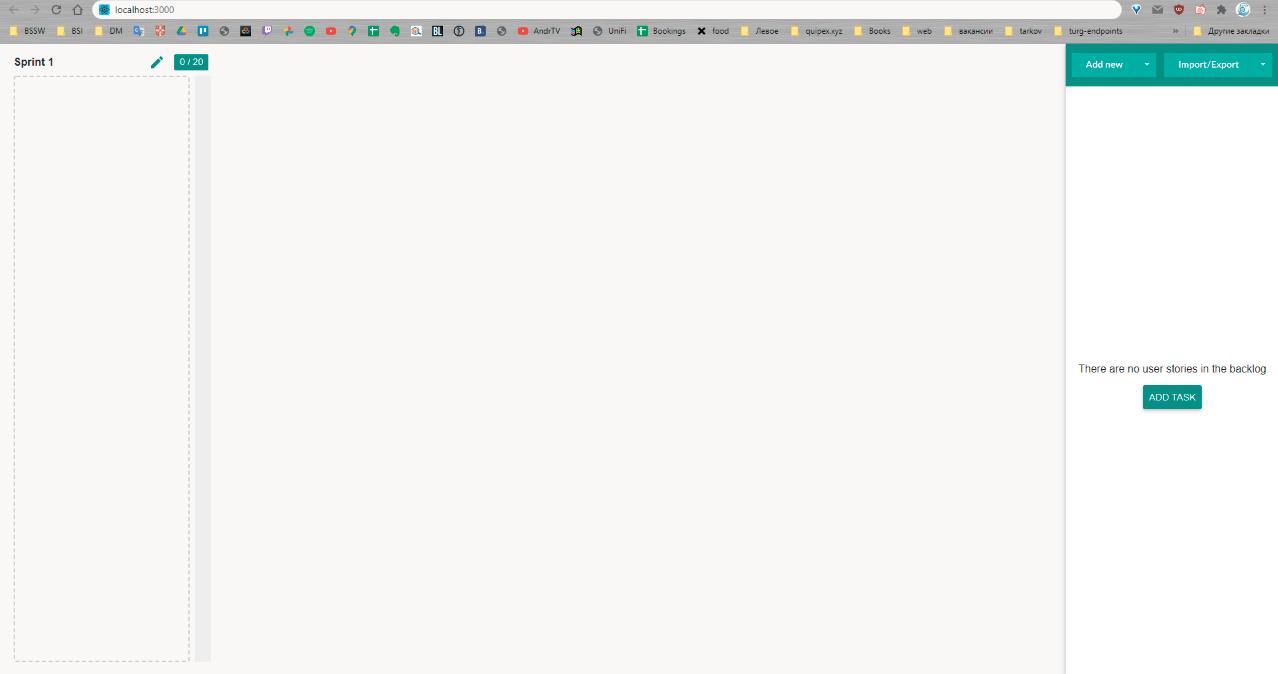
###### Рисунок 5.1 – Перший етап інсталяції – Docker image

Після завантаження образу, він з’явиться у списку образів Docker. Звідти його можна запустити двічі натиснувши на нього.



###### Рисунок 5.2 – Другий етап інсталяції – підняття образу

Запущений образ буде відображатися зеленим. Це свідчить про те, що він готовий до оперування ним. Образ може відображатися зеленим коли внутрішні процеси ще не завершилися, тоді, при переході за посиланням, потрібно буде зачекати.



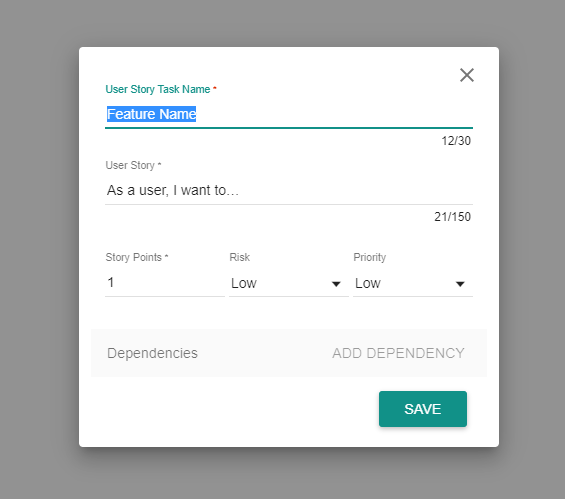
###### Рисунок 5.3 – Встановлений конструктор, що готовий до роботи

Після того, як контейнер буде запущений, а сторінка завантажиться, конструктором можна буде користуватися.

### 5.3 Опис використання системи

В попередньому підрозділі 5.2 на рис. 5.3 показано початковий екран конструктору. Він містить лише один спринт та випадаючі меню «Add new» та «Import/Export».

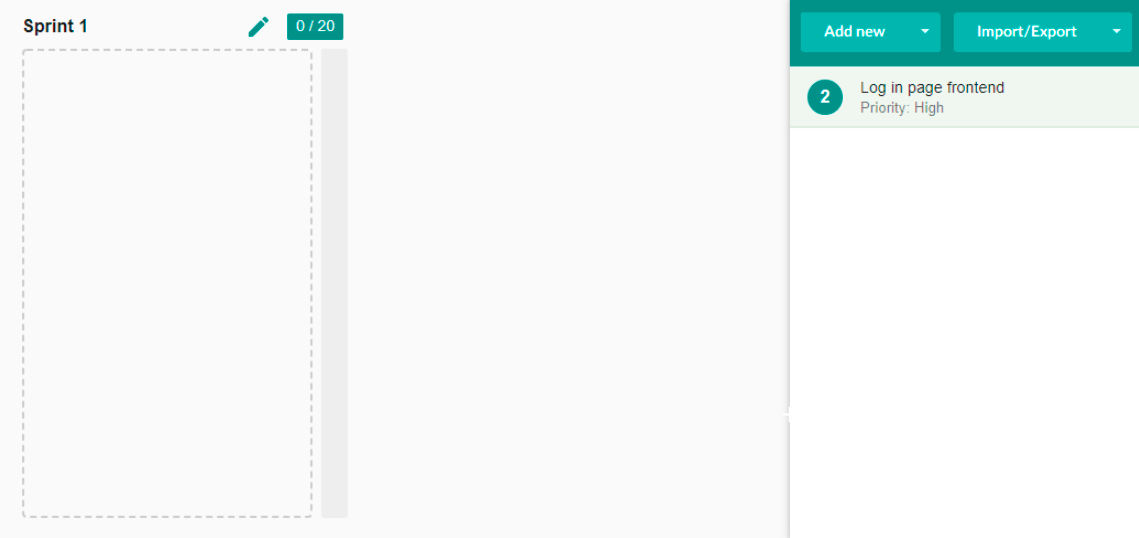
Розпочнемо роботу створивши першу юзер-сторі. Для цього натиснемо на випадаюче меню «Add new» та виберемо пункт «User Story». Перед нами з’явиться форма створення нової юзер-сторі (рис. 5.4).



###### Рисунок 5.4 – Форма для створення нової юзер-сторі

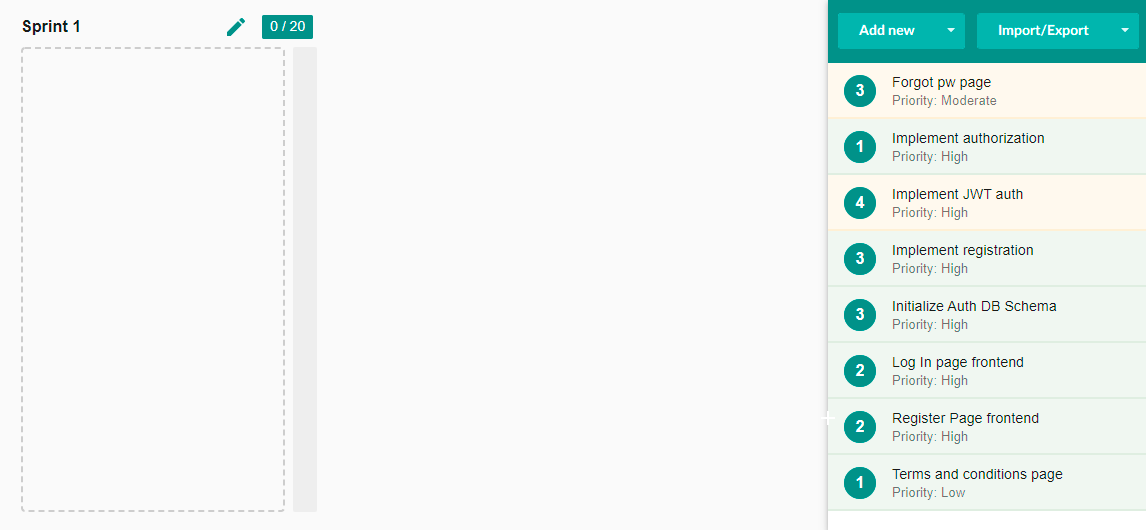
Заповнимо її необхідними нам даними: зазначимо назву, опис, кількість сторі поінтів, ризик та пріоритет. Залежності «Dependencies» ми вибрати не можемо через те, що інших юзер-сторі в беклозі поки не існує.

Створена юзер-сторі з’явиться в колонці беклогу, що видно на рис. 5.5



###### Рисунок 5.5 – Створена юзер-сторі в колонці беклогу

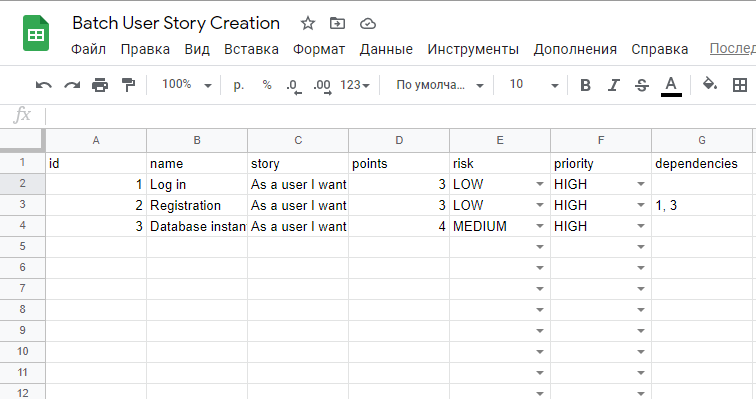
Створимо ще декілька юзер-сторі для більш наглядної демонстрації розташування великої кількості юзер-сторі у беклозі (рис. 5.6).



###### Рисунок 5.6 – Створення великої кількості юзер-сторі в беклозі

Виявляється, додавати дуже велику кількість юзер-сторі за допомогою форми не дуже зручно. Що як раптом нам знадобиться створити 10, 20, або більше юзер-сторі? Уявімо, що це потрібно буде ще й відтворити на різних комп’ютерах для різних студентів. Для таких ситуацій було реалізовано імпорт набору юзер-сторі з файлу. Ми можемо скористатися сторонніми сервісами, наприклад Google Sheets, для створення списку юзер-сторі, що буде експортовано в файл формату csv.

Для комфортного створення нових юзер-сторі за допомогою Google Sheets, я створив шаблон (рис. 5.7), який можна скопіювати до себе на обліковий запис google для подальшого редагування. Скористатися шаблоном можна за посиланням[[1]](#footnote-2), або скороченою його версією[[2]](#footnote-3)

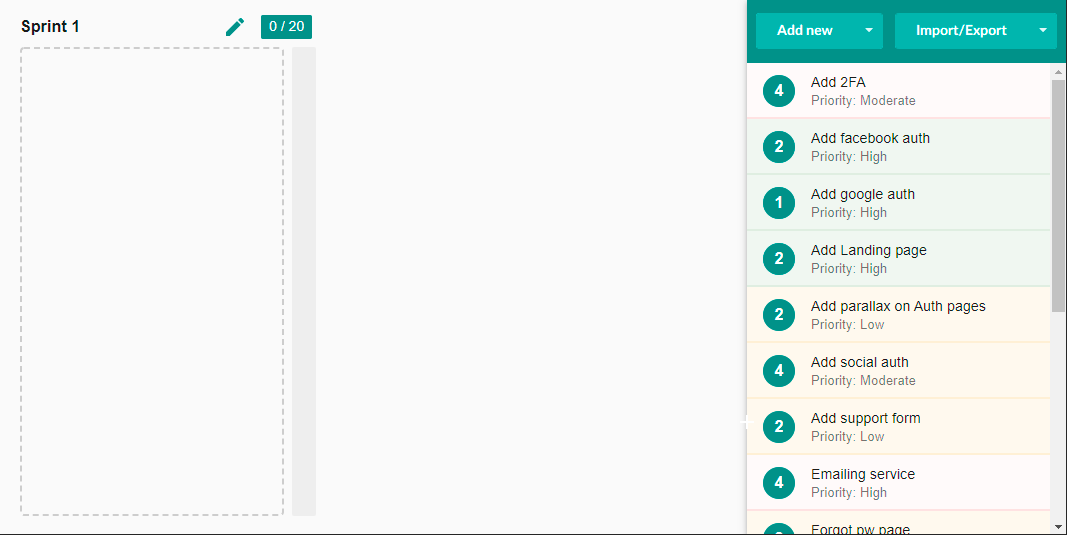


###### Рисунок 5.7 – Шаблон Google Sheets для створення файлу імпорту юзер-сторі

Для того аби скористатися шаблоном, необхідно буде перейти за посиланням, натиснути «Файл» після чого «Створити копію». На Вашому обліковому записі буде створена копія документу, яку можна буде редагувати.

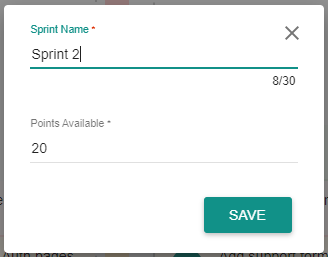
Вже створені юзер-сторі з ідентифікаторами (id) 1, 2 і 3 можна видалити та створити свої. Для подальшого імпорту набору юзер-сторі в систему потрібно буде натиснути «Файл», «Завантажити» та вибрати пункт «CSV файл (поточний лист)». Після чого, якщо потрібно, вибрати місце збереження файлу та зберегти його. Коли файл завантажено необхідно перейти до вікна з конструктором беклогів.

В вікні конструктору треба буде натиснути випадаюче меню «Import/Export» та вибрати «Import to backlog…». У вікні, що з’явилося, вибрати файл що було щойно скачано з Google Sheets та дочекатися імпорту юзер-сторі. Щойно імпорт завершиться, у беклозі з’являться нові юзер-сторі поряд з вже створеними (рис. 5.8).



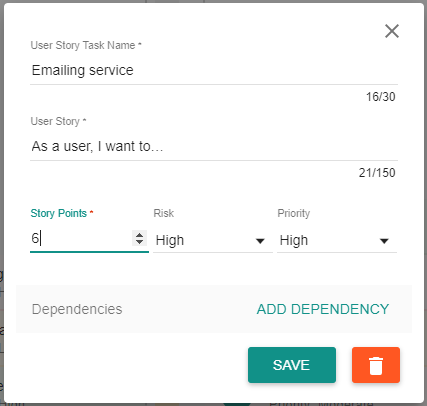
###### Рисунок 5.8 – Стан беклогу після імпорту юзер-сторі з файлу

Тепер, коли існує достатньо юзер-сторі для складання беклогу, розпочнемо його планування. Юзер-сторі досить багато тому створимо ще один спринт (рис. 5.9).



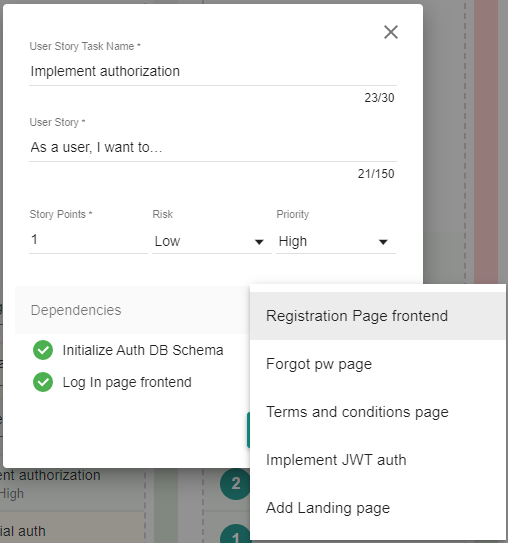
###### Рисунок 5.9 – Форма створення нового спринта

Виявляється, що юзер-сторі «Emailing service» більш трудомістка ніж ми її оцінили. Змінимо кількість сторі поінтів з 4 на 6. Для цього натиснемо на неї та змінимо дані в формі (рис. 5.10).



###### Рисунок 5.10 – Зміна кількості сторі поінтів для юзер-сторі «Emailing service»

Тепер, необхідно встановити залежності між юзер-сторі, адже без реалізації базових юзер-сторі неможливо реалізувати більш складні. Для цього натиснемо на юзер-сторі в яких треба встановити залежності, у формі натиснемо на кнопку «Add dependency» та оберемо залежності з випадаючого меню (рис. 5.11).



###### Рисунок 5.11 – Встановлення залежностей юзер-сторі

Тепер, коли юзер-сторі відредаговані як нам потрібно, розпочнемо їх переносити до спринтів (рис. 5.12).



Рисунок 5.12 – Перенесення юзер-сторі до спринту

Коли користувач намагається перенести юзер-сторі, спринти, в які перенос можливий, виділяються зеленою штриховою межею. Натомість, ті спринти, в які неможна перенести юзер-сторі, мають червону штрихову межу. Якщо тримати юзер-сторі над певним спринтом, кольором буде виділятися і його фон, що свідчить про те, що якщо відпустити юзер-сторі, вона перенесеться до цього спринта.

Юзер-сторі, що мають залежності, які ще не були розставлені, будуть виділені більш тьмяним кольором в беклозі (рис. 5.13), що свідчить про те, що вони неактивні доки їх залежності не будуть розміщені.

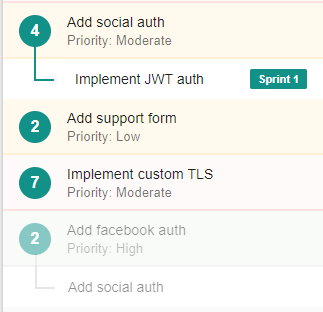


Рисунок 5.13 – Неактивні юзер-сторі через нерозташовані залежності

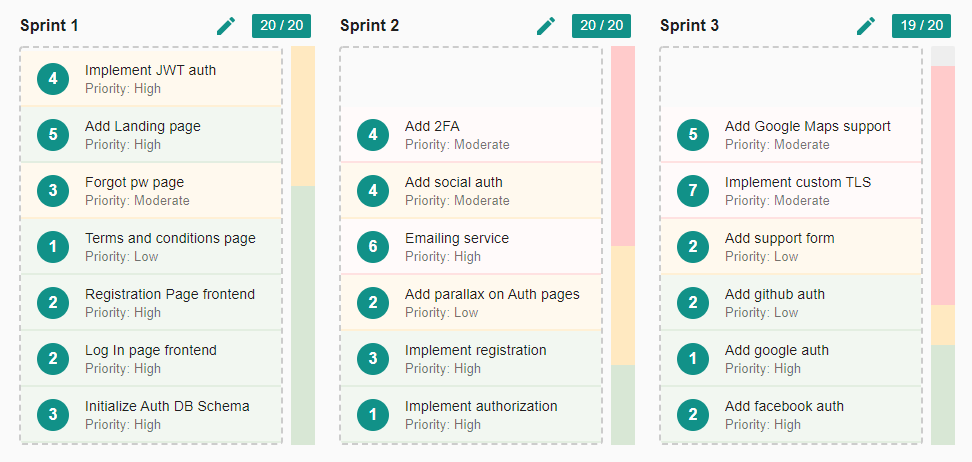
Неактивна на рис. 5.13 юзер-сторі під назвою «Add facebook auth». Від неї йде смужка, що вказує на її залежність – юзер-сторі під назвою «Add social auth». Як тільки вона буде розташована у спринті, юзер-сторі «Add facebook auth» стане активною. Проте, залежні юзер-сторі не можна розмістити в той самий спринт, або раніше від юзер-сторі, від якої вона залежить. Це показано на рис. 5.14.



###### Рисунок 5.14 – Обмеження залежних юзер-сторі

На рис. 5.14 у другого спринта є 4 вільні сторі-поінти, проте він не дає розмістити юзер-сторі через обмеження залежності.

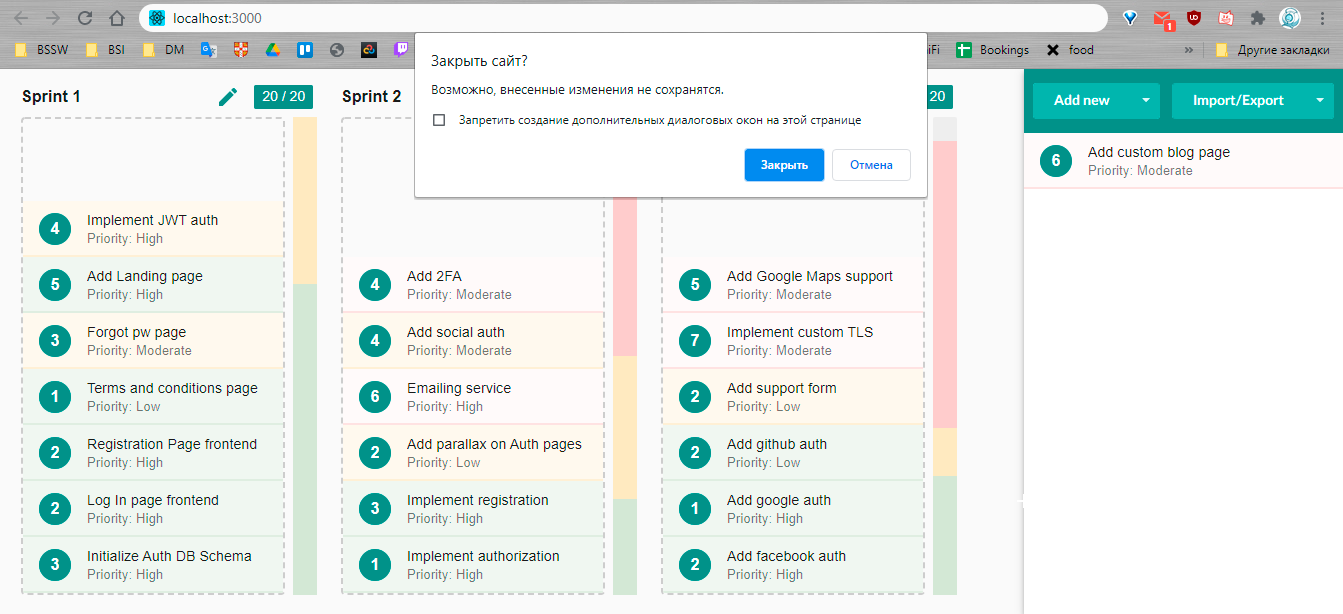
Продовжимо розташовувати решту юзер-сторі для завершення планування. Результат планування можна побачити на рис. 5.15.



###### Рисунок 5.15 – Завершене планування спринтів

Смужка справа від спринтів 2 та 3 свідчать про те, що ми мали б перепланувати спринти, бо ризик їх невиконання досить високий, проте суть прикладу не в ідеальному плануванні, а в демонстрації роботи конструктору, тому залишимо як є.

Нарешті, коли ми завершили планування, спробуємо закрити сторінку з конструктором. Браузер не дасть нам зробити це та попередить про те, що дані можуть бути втрачені (рис 5.16).



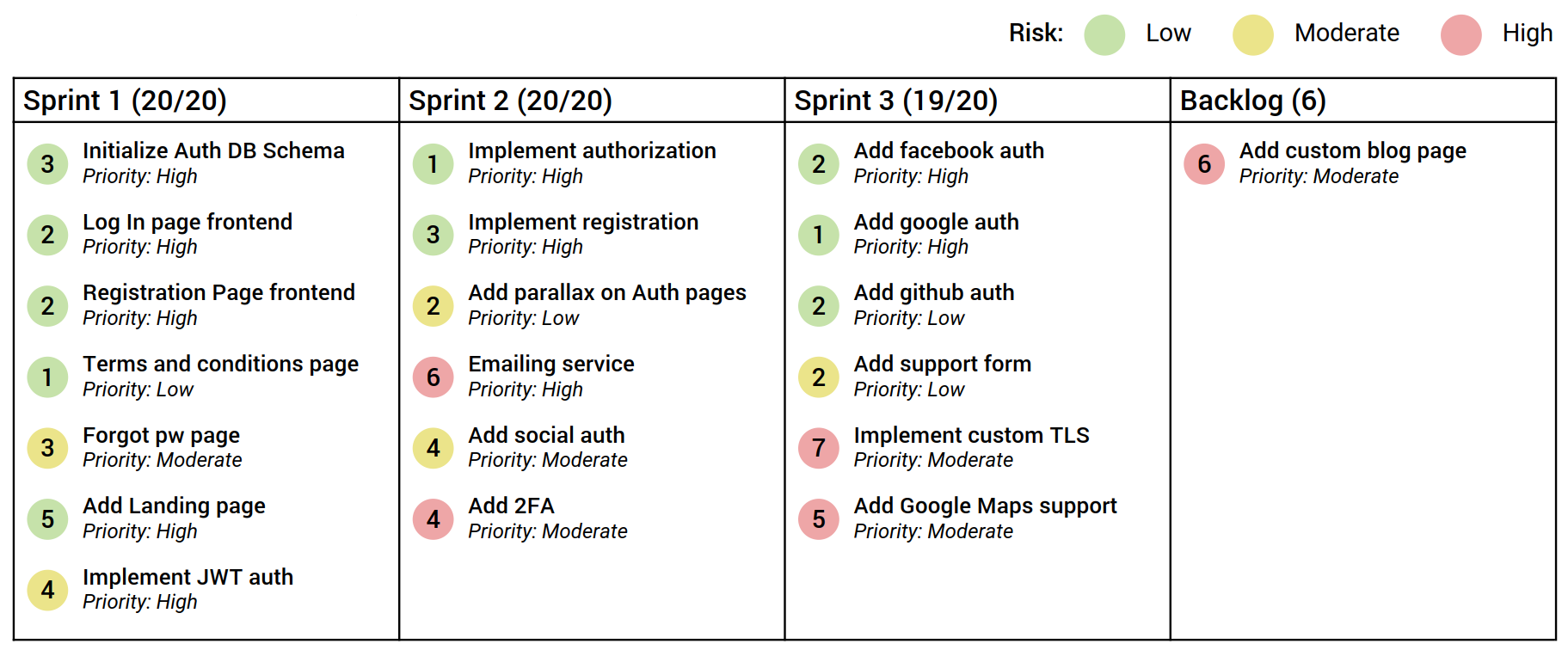
###### Рисунок 5.16 – Попередження браузеру про те, що дані не збережені

Чудово, що браузер попередив нас про те, що дані не були збережені, адже необхідно їх експортувати в файл для подальшого редагування. Для цього натиснемо на випадаюче меню «Import/Export» та виберемо «Save as…». Після цього оберемо шлях до куди ми хочемо зберегти файл та натиснемо кнопку «Зберегти».

Тепер, якщо закрити сторінку з конструктором, браузер не буде попереджати про те, що дані будуть втрачені, адже вони були збережені на комп’ютері. Закриємо сторінку та відкриємо наново. Стан конструктору змінився до початкового, який показано на рис. 5.3.

Відновімо збережений прогрес за допомогою меню «Import/Export» та пункту «Load from…». Виберемо файл, до якого ми зберегли прогрес, та натиснемо кнопку «Відкрити». Поточні дані зміняться на ті, що були збережені до файлу.

Наостанок, експортуємо зроблену роботу у файл PDF для зручного друку. Для цього натиснемо «Import/Export» та «Export to PDF». Оберемо місце та ім’я експортованого файлу та натиснемо кнопку «Зберегти». Нарешті, відкриємо збережений файл будь-яким PDF переглядачем. Результат можна побачити на рис. 5.17.



###### Рисунок 5.17 – Вміст експортованого плану спринтів у PDF

## 6 ОХОРОНА ПРАЦІ

Охорона праці (ОП) – це система нормативно-правових, організаційно-технічних, соціально-економічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини під час виконання трудової діяльності [26].

Правовою основою законодавства щодо ОП є Конституція України, в якій підкреслюється, що людина, її життя і здоров'я, недоторканість і безпека визначаються в Україні найвищою соціальною цінністю.

Основоположним законодавчим документом у галузі ОП є Закон України «Про охорону праці», дія якого поширюється на всі підприємства, установи та організації незалежно від форм власності та видів їх діяльності. «Кодекс законів про працю (КЗпП) України» проголошує правові засади та гарантії здійснення громадянами України права розпоряджатися своїми здібностями до продуктивної та творчої праці; регулює трудові відносини працівників усіх підприємств, установ незалежно від форм власності, виду діяльності й галузевої належності.

Найнижчим рівнем нормативно-правових актів з ОП є стандарти, норми, правила, інструкції, тощо, які в тому числі регулюють умови безпосередньо робочого місця. Держава формує політику охорони здоров'я та забезпечує її реалізацію, використовуючи світовий досвід роботи з поліпшення умов і безпеки праці.

Організація заходів із забезпечення безпечних умов праці та поведінки у надзвичайних ситуаціях засновується на наступних нормативних документах:

* ДСТУ 2293-99. Охорона праці. Терміни та визначення [27];
* НАПБ А.01.001-2015. Правила пожежної безпеки в Україні [28];
* ДСН 3.3.6.037-99. Санітарні норми виробничого шуму, ультразвуку та інфразвуку;
* НПАОП 0.00-1.28-10. Правила охорони праці при експлуатації ЕОМ;
* ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень;
* НАПБ Б.03.001-2204. Типові норми належності вогнегасників;
* ДБН В.2.5-28-2006. Природне і штучне освітлення.

Вивчення найважливіших заходів з охорони праці та безпеки у надзвичайних ситуаціях для працівників галузі інженерія програмного забезпечення наведено у Додатку Б та висвітлює наступні питання:

* аналіз умов праці і вибір заходів і засобів захисту від небезпечних і шкідливих виробничих факторів;
* аналіз техногенних небезпек і вибір заходів і засобів забезпечення безпеки у надзвичайних ситуаціях;
* аналіз робочого приміщення на пожежонебезпечність.

В ході роботи проаналізовано приміщення роботи інженера програміста на предмет пожежонебезпечності.

## ВИСНОВКИ

В кваліфікаційній роботі було розроблено програмну реалізацію інтерактивного навчального конструктору беклогів спринтів.

Для цього спочатку були проаналізовані існуючі рішення щодо потреб кінцевого користувача в контексті навчання. Наведено порівняння функціоналу існуючих рішень.

На основі результатів порівняльного аналізу існуючих рішень, були сформульовані вимоги до програмної системи та розроблена специфікація вимог, до якої входять функціональні та нефункціональні вимоги. Сформовані вимоги націлені на навчальний процес для досягнення мети модернізації процесу навчання.

Далі була спроектована архітектура майбутньої системи для задоволення функціональних та нефункціональних вимог. Також були спроектовані інтерфейс користувача, структури даних для експорту та імпорту, обрана реалізація сховища даних. Зроблені проектні рішення націлені на досягнення мети та мінімізацію затрат часу при реалізації програмного продукту.

Після завершення проектування настала черга програмної реалізації. Було обрано стек технологій та інструменти розробки для реалізації сформованих вимог, описані реалізовані методи, що використовуються в системі.

Нарешті, коли була реалізована програмна частина, були створені тест-кейси для перевірки задоволення функціональних та нефункціональних вимог. Система була протестована в повному циклі її роботи і показала задовільний результат, що є достатнім для повноцінного її функціонування і є підтвердженням досягнення мети кваліфікаційної роботи.

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Мартін, Д. Rapid Application Development [Текст] / Мартін Д. – Macmillan, 1991. — 228 с.
2. Agile project management statistics [Електронний ресурс] / M. Langley, PMI – Режим доступу : www / URL : https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017 – Жовтень 2017 р. – Загл. з екрану.
3. Agile methodologies [Електронний ресурс] / B. Kent, Blueprint Inc. – Режим доступу : www / URL : https://www.blueprintsys.com/agile-development-101/agile-methodologies – квітень 2013 р. – Загл. з екрану.
4. Systems and software engineering – Requirements engineering [Електронний ресурс] / ISO/IEC JTC 1/SC 7 Software and systems engineering – Режим доступу : www / URL : https://www.iso.org/ru/standard/45171.html – грудень 2011 р. – Загл. з екрану.
5. Фаулер, М. UML Основи Короткий посібник зі стандартної мови об’єктного моделювання 3-е видання [Текст] / М. Фаулер – СПб. : Символ плюс, 2004. – 192 с.
6. Басс, Л. Архітектура програмного забезпечення на практиці 2-е видання [Текст] / Л. Басс, П. Клементс, Р. Кацман – СПб. : Питер, 2006. – 575 с.
7. Docker [Електронний ресурс] / R. Zwitserloot, R. Spilker – Режим доступу : www / URL : https://www.docker.com/ – грудень 2020 р. – Загл. з екрану.
8. Мацяшек, Л.А. Аналіз вимог і проектування систем. Розробка інформаційних систем з використанням UML [Текст] / Л.А. Мацяшек – М. : Вильямс, 2002. – 432 с.
9. Узагальнений Model-View-Controller [Електронний ресурс] С. Рогачев – Режим доступу : www / URL : http://rsdn.org/article/patterns/generic-mvc.xml — безерень 2007 р. – Загл. з екрану.
10. Язык UML. Руководство пользователя [Текст] / Г. Буч, Д. Рамбо, И. Якобсон. — М. : ДМК Пресс, 2006. — 496 с.
11. Введение в системы баз данных [Текст] / К. Дейт — М. : Вильямс, 2005. — 1328 с.
12. NoSQL: новая методология разработки нереляционных баз данных [Текст] / М. Фаулер, П. Садаладж. — М. : Вильямс, 2013. — 192 с.
13. Common Format and MIME Type for Comma-Separated Values (CSV) Files [Електронний ресурс] / Y. Shafranovich, SolidMatrix Technologies, Inc. – Режим доступу : www / URL : https://www.ietf.org/rfc/rfc4180.txt – жовтень 2005 р. – Загл. з екрану.
14. The application/json Media Type for JavaScript Object Notation (JSON) [Текст] / D. Crockford — US. : Internet Engineering Task Force, 2006. – 10с.
15. Single-page applications vs. multiple-page applications: pros, cons, pitfalls [Електронний ресурс] / D. Snoop, BLAKIT – Режим доступу : www / URL : https://ozitag.com/blog/spa-advantages – грудень 2015 р. – Загл. з екрану.
16. React – JavaScript-библиотека для создания пользовательских интерфейсов [Електронний ресурс] / Facebook Inc. – Режим доступу : www / URL : https://ru.reactjs.org/ – грудень 2020 р. – Загл. з екрану.
17. Semantic UI React – The official Semantic-UI-React integration [Електронний ресурс] / Open Source – Режим доступу : www / URL : https://react.semantic-ui.com/ – грудень 2020 р. – Загл. з екрану.
18. Redux – A Predictable State Container for JS Apps [Електронний ресурс] / Open Source – Режим доступу : www / URL : https://redux.js.org/ – грудень 2020 р. – Загл. з екрану.
19. React Redux – Official React bindings for Redux [Електронний ресурс] / Open Source – Режим доступу : www / URL : https://react-redux.js.org/ – грудень 2020 р. – Загл. з екрану.
20. Csv-parse – CSV Parser for Node.js [Електронний ресурс] / Open Source – Режим доступу : www / URL : https://csv.js.org/ – грудень 2020 р. – Загл. з екрану.
21. Jest – Delightful Javascript testing [Електронний ресурс] / Open Source – Режим доступу : www / URL : https://jestjs.io/ – грудень 2020 р. – Загл. з екрану.
22. React Testing Library – Simple and complete testing utilities that encourage good testing practices [Електронний ресурс] / Open Source – Режим доступу : www / URL : https://testing-library.com/ – грудень 2020 р. – Загл. з екрану.
23. Visual Studio Code – Code editing. Redefined. [Електронний ресурс] / Microsoft – Режим доступу : www / URL : https://code.visualstudio.com/ – грудень 2020 р. – Загл. з екрану.
24. Redux fundamentals – Redux flow [Електронний ресурс] / Facebook – Режим доступу : www / URL : https://redux.js.org/tutorials/fundamentals/part-2-concepts-data-flow – грудень 2020 р. – Загл. з екрану.
25. Системне програмне тестування [Електронний ресурс] / А. Олексієнко, QA-Media – Режим доступу : www / URL : https://qalight.com.ua/baza-znaniy/sistemnoe-testirovanie/ – червень 2018 р. – Загл. з екрану.
26. Керб Л.П. Основи охорони праці: Навчально-методичний посібник для самостійного вивчення дисципліни / Л.П. Керб. – К.: КНЕУ, 2001. – 252 с.
27. Охорона праці. Терміни та визначення: ДСТУ 2293-99. – Режим доступу : www / URL : https://dnaop.com/html/34095/doc-%D0%94%D0%A1  
    %D0%A2%D0%A3\_2293-99
28. НАПБ А.01.001-2015. Правила пожежної безпеки в Україні – Режим доступу : www / URL : http://deos-release.com/image/catalog/img/pdf/  
    NAPB%20A.01.001-2014.pdf

1. Посилання на шаблон Google Sheets – https://docs.google.com/spreadsheets/d/1LY8xNjRP  
   A8EayW9Gm4NhX96gR4Q1eSr4wOhh5mH-5JM [↑](#footnote-ref-2)
2. Скорочена версія посилання на шаблон Google Sheets – https://cutt.ly/IhTXQxW [↑](#footnote-ref-3)