ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської кваліфікаційної роботи

на ступінь вищої освіти бакалавр

на тему: **«РОЗРОБКА UI ДЛЯ ЕЛЕКТРОННОГО ЖУРНАЛУ НА МОВІ JAVASCRIPT»**

Виконав: студент 4 курсу, групи ПД-44

спеціальності 121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

Приндюк В.С.

(прізвище та ініціали)

Керівник Негоденко О.В.

(прізвище та ініціали)

Рецензент

(прізвище та ініціали)

Нормоконтроль

(прізвище та ініціали)

Київ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти - «Бакалавр»

Спеціальність -121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного
забезпечення
_____О.В.Негоденко
«____»__ _____2021 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

ПРИНДЮКУ ВЛАДИСЛАВУ СЕРГІЙОВИЧУ

1.Тема роботи: «Розробка UI для електронного журналу на мові JavaScript»
Керівник роботи Негоденко Олена Василівна, кандидат технічних наук, доцент
затверджені наказом вищого навчального закладу від — «12» березня 2021 року №65.

2. Строк подання студентом роботи 01.06.2021

3. Вхідні дані до роботи:

3.1. Середовище розробки WebShtorm 2021.1

3.2. Алгоритм авторизації клієнта

3.3. UI для клієнтів

3.4. Науково-технічна література, пов’язана з розробкою UI для електронного журналу

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).

4.1. Аналіз обов’язків студента і викладача

4.2. Аналіз та порівняння існуючих прототипів

4.3. Дослідження програмних засобів для розробки UI

4.4. Розробити функціонал для електронного

5. Перелік графічного матеріалу

5.1.1. Популярність електронного журналу

5.1.2. Переваги та недоліки електронного журналу

5.1.3. Програмні засоби реалізації

5.1.4. Огляд можливостей UI для електронного журналу

5.1.5. Апробація результатів досліджень

6. Дата видачі завдання 19.04.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	19.04.21 – 22.04.21	
2	Аналіз існуючих прототипів	22.04.21 – 23.04.21	
3	Дослідження програмних засобів	23.04.21 – 01.05.21	
4	Моделювання об’єкту проектування	01.05.21 – 05.05.21	
5	Розробка UI	05.05.21 – 17.05.21	
6	Вступ, висновки, реферат	17.05.21 – 18.05.21	
7	Розробка презентації застосунку	19.05.21 – 24.05.21	
8	Попередній захист роботи	25.05.21	

Студент

Керівник роботи

РЕФЕРАТ

Текстова частина бакалаврської роботи 54с., 11рис., 15 джерел.

Ключеві слова: WebShtorm, React, JavaScript, CSS, HTML, UI, електронний журнал, веб сайти, сайт.

Об'єкт дослідження – перенесення документообігу в електронний вигляд та автоматизація рутинних процесів.

Предмет дослідження – електронний журнал, для заміни паперового журналу.

Мета роботи – розробка UI для електронного журналу на мові програмування JavaScript.

Методи дослідження – методи теорії інформації, методи структурного аналізу і проектування, методи розробки програмного забезпечення, методи тестування, валідації та верифікації програмного забезпечення.

Досліджено основні особливості побудови та функціонування архітектури технології веб-сайтів та підкреслено необхідність широкого дослідження особливостей даної технології.

В роботі виконано аналіз існуючих UI для електронних журналів аналогів. Встановлено переваги та недоліки існуючих журналів. В результаті аналізу було визначено основні потреби користувачів. Проаналізовано можливості середовища розробки WebShtorm. Розроблено логіку та UI, загальну концепцію представлення інформації для користувачів.

Галузь використання – журнал можуть використовувати як вчителі так і студенти які мають зв'язок з інтернетом.

ЗМІСТ

ВСТУП

В сучасному світі кожного дня кількість інформації, яку необхідно зберігати росте в геометричній прогресії , набуваючи величезних розмірів. В певний момент старі варіанти зберігання інформації, такі як паперові носії, або навіть усна форма, віджили своє і більше не могли задовольняти потреби сучасного соціуму.

Саме в цей момент і з’явилося систем автоматизації, які до сьогоднішнього дня допомагають людям у структуруванні та зберіганні інформації різного роду, взаємодії між ролями користувачів, тощо.

На сьогоднішній день кожне підприємство та установа мають свої внутрішні клієнти, які допомагають зберігати і структурувати інформацію про роботи даної установи, автоматизувати деякі процеси, тощо.

Виходячи з актуальності явища систем автоматизації, об’єктом дослідження є перенесення документообігу в електронний вигляд та автоматизація рутинних процесів.

Предмет дослідження — UI для інформаційної системи електронного журналу.

Мета роботи — розроблення UI для інформаційної системи електронного журналу засобами мови програмування JavaScript.

Для досягнення поставленої мети слід виконати наступні завдання:
Для виконання поставленої мети слід виконати наступні завдання:

- Провести огляд поняття веб-сайту
- Провести аналіз методів та засобів розробки веб-сайтів
- Провести огляд поняття електронного журналу
- Обрати мову програмування
- Обрати додаткові інструменти
- Обрати середовище розробки
- Побудувати діаграму варіантів використання

- Розробити сторінку входу
- Розробити сторінку реєстрації
- Розробити сторінку виставлення оцінок
- Розробити сторінку статистики успішності
- Розробити сторінку розкладу
- Розробити сторінку завдань
- Розробити сторінку профілю

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Веб-сайти

Веб-сайт - це сукупність веб-сторінок та пов’язаного вмісту, які ідентифікуються загальним доменним ім’ям та публікуються принаймні на одному веб-сервері. Помітними прикладами є wikipedia.org, google.com та amazon.com.

Усі загальнодоступні веб-сайти в сукупності складають Всесвітню павутину. Існують також приватні веб-сайти, доступ до яких доступний лише в приватній мережі, наприклад, внутрішній веб-сайт компанії для її співробітників.

Веб-сайти, як правило, присвячені певній темі або меті, такі як новини, освіта, комерція, розваги чи соціальні мережі. Гіперпосилання між веб-сторінками спрямовує навігацію по сайту, яке часто починається з домашньої сторінки.

Користувачі можуть отримати доступ до веб-сайтів на різних пристроях, включаючи настільні, ноутбуки, планшети та смартфони. Програма, що використовується на цих пристроях, називається веб-браузером.

Всесвітня павутина (WWW) була створена в 1990 році британським фізиком ЦЕРН Тімом Бернерсом-Лі. [1] 30 квітня 1993 р. ЦЕРН оголосив, що Всесвітня павутина буде безкоштовною для використання будь-ким [2]. До впровадження протоколу передачі гіпертексту (HTTP) для отримання окремих файлів із сервера використовувались інші протоколи, такі як протокол передачі файлів та протокол gopher. Ці протоколи пропонують просту структуру каталогів, в якій користувач переходить і де він обирає файли для завантаження. Документи найчастіше подавалися у вигляді текстових файлів без форматування або кодувались у форматах текстового процесора.

Веб-сайти можуть використовуватися по-різному: персональний веб-сайт, корпоративний веб-сайт компанії, урядовий веб-сайт, веб-сайт організації тощо. Веб-сайти можуть бути роботою приватної особи, бізнесу чи іншої організації і, як правило, присвячені конкретна тема або мета. Будь-який веб-сайт може містити гіперпосилання на будь-який інший веб-сайт, тому різниця між окремими сайтами, як сприймається користувачем, може бути розмитою.

Деякі веб-сайти вимагають реєстрації користувачів або передплати для доступу до вмісту. Приклади веб-сайтів, на які здійснюється підписка, включають багато бізнес-сайтів, веб-сайти новин, веб-сайти академічних журналів, ігрові веб-сайти, веб-сайти спільного використання файлів, дошки оголошень, веб-адреси електронної пошти, веб-сайти соціальних мереж, веб-сайти, що надають дані про фондовий ринок у режимі реального часу, а також веб-сайти, що надають різні інші послуги.

Хоча "веб-сайт" був оригінальним написанням (іноді з великої літери "Веб-сайт", оскільки "Веб" є власним іменником, коли йдеться про Всесвітню павутину), цей варіант став рідкісним, а "веб-сайт" став стандартним написанням . Усі основні керівництва стилем, такі як Чиказький посібник стилю [3] та AP Stylebook [4], відображають цю зміну.

Статичний веб-сайт - це веб-сторінки, які мають веб-сторінки, що зберігаються на сервері у форматі, який надсилається клієнтському веб-браузеру. В основному він кодується мовою розмітки гіпертексту (HTML); Каскадні таблиці стилів (CSS) використовуються для контролю зовнішнього вигляду за базовим HTML. Зображення зазвичай використовуються для отримання бажаного вигляду та як частина основного змісту. Аудіо чи відео також можна вважати "статичним" вмістом, якщо він відтворюється автоматично або, як правило, є неінтерактивним. Цей тип веб-сайтів зазвичай відображає однакову інформацію для всіх відвідувачів. Подібно до роздачі друкованої брошури клієнтам або клієнтам, статичний веб-сайт, як правило, надає послідовну стандартну інформацію протягом тривалого періоду часу.

Незважаючи на те, що власник веб-сайту може періодично робити оновлення, редагування тексту, фотографій та іншого вмісту здійснюється вручну, і може знадобитися базові навички дизайну веб-сайту та програмне забезпечення. Прості форми або маркетингові приклади веб-сайтів, такі як класичний веб-сайт, веб-сайт із п'ятьма сторінками або веб-сайт брошури, часто є статичними веб-сайтами, оскільки вони представляють користувачеві заздалегідь визначену, статичну інформацію. Це може включати інформацію про компанію та її продукти та послуги через текст, фотографії, анімацію, аудіо / відео та навігаційні меню.

Статичні веб-сайти все ще можуть використовувати серверні компоненти (SSI) як зручність редагування, наприклад, спільний доступ до загального рядка меню на багатьох сторінках. Оскільки поведінка сайту до читача все ще залишається статичним, це не вважається динамічним сайтом.

Динамічний веб-сайт - це веб-сайт, який часто і автоматично змінюється або налаштовується. Динамічні сторінки на сервері генеруються "на льоту" за допомогою комп'ютерного коду, який виробляє HTML (CSS відповідає за зовнішній вигляд і, отже, статичні файли). Існує широкий спектр програмних систем, таких як CGI, Java-сервлети та Java Server Pages (JSP), Active Server Pages і ColdFusion (CFML), які доступні для створення динамічних веб-систем та динамічних веб-сайтів. Різні фреймворки веб-програм та системи веб-шаблонів доступні для загальновживаних мов програмування, таких як Perl, PHP, Python та Ruby, щоб полегшити та спростити створення складних динамічних веб-сайтів.

Сайт може відображати поточний стан діалогу між користувачами, відстежувати мінливу ситуацію або надавати інформацію певним чином персоналізовану відповідно до вимог окремого користувача. Наприклад, коли запитується головна сторінка сайту новин, код, що працює на веб-сервері, може поєднувати збережені фрагменти HTML із новинами, отриманими з бази даних або іншого веб-сайту за допомогою RSS, щоб створити сторінку, що включає найсвіжішу інформацію. Динамічні сайти можуть бути

інтерактивними, використовуючи HTML-форми, зберігаючи та зчитуючи файли cookie браузера, або створюючи ряд сторінок, що відображають попередню історію кліків. Інший приклад динамічного вмісту - це коли роздрібний веб-сайт з базою даних медіа-продуктів дозволяє користувачеві вводити запит на пошук, наприклад за ключовим словом "Бітлз". У відповідь вміст веб-сторінки спонтанно змінить те, як він виглядав раніше, а потім відобразить список продуктів "Бітлз", таких як компакт-диски, DVD-диски та книги. Динамічний HTML використовує код JavaScript, щоб вказувати веб-браузеру, як інтерактивно змінювати вміст сторінки. Одним із способів моделювання певного типу динамічного веб-сайту, уникаючи втрати продуктивності ініціювання динамічного механізму для кожного користувача або підключення, є періодична автоматична регенерація великої серії статичних сторінок.

1.2. Розробка веб-сайтів

Ранні веб-сайти мали лише текст, а незабаром і зображення. Потім плагіни веб-браузера використовувались для додавання аудіо, відео та інтерактивності (наприклад, для багатофункціональної програми Інтернету, яка відображає складність настільної програми, як текстовий процесор). Прикладами таких плагінів є Microsoft Silverlight, Adobe Flash, Adobe Shockwave та аплети, написані на Java. HTML 5 містить положення щодо аудіо та відео без плагінів. JavaScript також вбудований у більшість сучасних веб-браузерів і дозволяє творцям веб-сайтів надсилати код веб-браузеру, який вказує йому, як інтерактивно змінювати вміст сторінки та спілкуватися з веб-сервером, якщо це необхідно. Внутрішнє представлення вмісту браузера відомо як об’єктна модель документа (DOM), а техніка - динамічний HTML.

WebGL (Web Graphics Library) - це сучасний API JavaScript для візуалізації інтерактивної 3D-графіки без використання плагінів. Це дозволяє

інтерактивний контент, такий як 3D-анімація, візуалізація та відео-пояснення, представленим користувачам найбільш інтуїтивно зрозумілим способом. [5]

Тенденція 2010 року на веб-сайтах під назвою "адаптивний дизайн" забезпечила найкращий досвід перегляду, оскільки надає користувачам макет на основі пристрою. Ці веб-сайти змінюють свій макет відповідно до пристрою або мобільної платформи, що забезпечує багатий досвід користувачів. [6]

Веб-сайти можна розділити на дві великі категорії - статичні та інтерактивні. Інтерактивні сайти є частиною спільноти веб-сайтів Web 2.0 і дозволяють взаємодіяти між власником сайту та відвідувачами або користувачами сайту. Статичні сайти обслуговують або збирають інформацію, але не дозволяють взаємодіяти з аудиторією або користувачами безпосередньо. Деякі веб-сайти є інформаційними або виготовляються ентузіастами або для особистого користування чи розваги. Багато веб-сайтів мають на меті заробляти гроші, використовуючи одну або кілька бізнес-моделей, зокрема:

- Розміщення цікавого контенту та продаж контекстної реклами або шляхом прямих продажів, або через рекламну мережу.
- Електронна комерція: товари чи послуги купуються безпосередньо через веб-сайт
- Реклама товарів або послуг, доступних у цегельному та будівельному бізнесі
- Freemium: базовий вміст доступний безкоштовно, але преміум-вміст вимагає оплати (наприклад, веб-сайт WordPress, це платформа з відкритим кодом для створення блогу чи веб-сайту).

Деякі веб-сайти можуть бути включені в одну або кілька з цих категорій.

Наприклад, веб-сайт бізнесу може рекламувати продукцію бізнесу, але також може розміщувати інформативні документи, такі як технічні документи. Існують також численні підкатегорії до перерахованих вище. Наприклад, порносайт - це певний тип веб-сайту електронної комерції або бізнес-сайту

(тобто він намагається продати членство для доступу до свого сайту) або має можливості соціальних мереж. Фан-сайт може бути посвятою власника певній знаменитості. Веб-сайти обмежені архітектурними обмеженнями (наприклад, обчислювальна потужність, присвячена веб-сайту). Дуже великі веб-сайти, такі як Facebook, Yahoo !, Microsoft та Google, використовують багато серверів та обладнання для балансування навантажень, наприклад, комутатори служб вмісту Cisco, для розподілу навантажень відвідувачів на декілька комп'ютерів у різних місцях. На початок 2011 року Facebook використовував 9 центрів обробки даних із приблизно 63000 серверами.

У лютому 2009 року компанія Netcraft, компанія з моніторингу Інтернету, яка відслідковувала зростання Інтернету з 1995 р., Повідомила, що в 2009 р. Існувало 215 675 903 веб-сайтів з доменними іменами та вмістом на них, порівняно з лише 19 732 веб-сайтами в серпні 1995 р. [8] Після досягнення 1 мільярда веб-сайтів у вересні 2014 року, етап, підтверджений NetCraft в огляді веб-серверів за жовтень 2014 року, і те, що Інтернет-статистика стала першим, хто повідомив про це, про що свідчить цей твіт від самого винахідника Всесвітньої мережі, Тіма Бернерса Лі - кількість веб-сайтів у світі згодом зменшилась, повернувшись до рівня нижче 1 мільярда. Це пов'язано з щомісячними коливаннями кількості неактивних веб-сайтів. Кількість веб-сайтів до березня 2016 р. Продовжувала зростати до понад 1 млрд. І з тих пір продовжує зростати [9].

Веб-розробка - це робота, пов'язана з розробкою веб-сайту для Інтернету (Всесвітня павутина) або інтрамережі (приватна мережа). [1] Веб-розробка може варіюватися від розробки простої однієї статичної сторінки простого тексту до складних Інтернет-програм (Інтернет-додатків), електронного бізнесу та послуг соціальних мереж. Більш повний перелік завдань, до яких зазвичай відноситься веб-розробка, може включати веб-інженерію, веб-дизайн, розробку веб-вмісту, зв'язок з клієнтом, сценарії на стороні клієнта / сервера, налаштування безпеки веб-сервера та мережі та розвиток електронної комерції.

Серед веб-спеціалістів "веб-розробка" зазвичай відноситься до основних недизайнерських аспектів побудови веб-сайтів: написання розмітки та кодування. [2] Веб-розробка може використовувати системи управління вмістом (CMS), щоб зробити зміст вмістом простішим та доступнішим з базовими технічними навичками.

Для великих організацій та бізнесу групи веб-розробників можуть складатися з сотень людей (веб-розробників) і дотримуватися стандартних методів, таких як Agile, під час розробки веб-сайтів. Менші організації можуть вимагати лише одного постійного або підрядного розробника, або вторинне призначення на відповідні посади, такі як графічний дизайнер або технік інформаційних систем. Веб-розробка може бути спільним зусиллям між департаментами, а не областю призначеного департаменту. Існує три різновиди спеціалізації веб-розробників: розробник з інтерфейсу, розробник з інтерфейсу та розробник із повним стеком. Інтернетні розробники несуть відповідальність за поведінку та візуальні ефекти, які працюють у браузері користувача, тоді як інтерфейсні розробники мають справу з серверами.

З моменту комерціалізації Інтернету веб-розробка стала зростаючою галуззю. Зростання цієї галузі відбувається за рахунок підприємств, які бажають використовувати свій веб-сайт для реклами та продажу товарів та послуг споживачам. [3]

Існує багато інструментів з відкритим кодом для веб-розробки, таких як BerkeleyDB, GlassFish, стек LAMP (Linux, Apache, MySQL, PHP) та Perl / Plack. Це дозволило звести витрати на навчання веб-розробці до мінімуму. Ще одним фактором, що сприяє зростанню галузі, стало зростання простого у використанні програмного забезпечення для веб-розробки WYSIWYG, такого як Adobe Dreamweaver, BlueGriffon та Microsoft Visual Studio. Для використання такого програмного забезпечення все ще необхідні знання мови розмітки HyperText (HTML) або мов програмування, але основи можна швидко вивчити та впровадити.

Постійно зростаючий набір інструментів та технологій допоміг розробникам створювати більш динамічні та інтерактивні веб-сайти. Крім того, веб-розробники тепер допомагають надавати програми як веб-служби, які традиційно були доступні лише як програми на настільному комп'ютері. Це дало багато можливостей для децентралізації розповсюдження інформації та засобів масової інформації. Приклади можна побачити із зростанням хмарних сервісів, таких як Adobe Creative Cloud, Dropbox та Google Drive. Ці веб-служби дозволяють користувачам взаємодіяти з програмами з багатьох місць, замість того, щоб бути прив'язаними до певної робочої станції для свого середовища програм.

Прикладами кардинальних перетворень у спілкуванні та комерції на чолі з веб-розробкою є електронна комерція. Інтернет-сайти аукціонів, такі як eBay, змінили спосіб споживачів знаходити та купувати товари та послуги. Інтернет-магазини, такі як Amazon.com та Buy.com (серед багатьох інших), змінили досвід покупок та торгівлі для багатьох споживачів. Іншим прикладом трансформаційного спілкування, керованого веб-розробкою, є блог. Веб-програми, такі як WordPress та Movable Type, створили середовища для ведення блогів для окремих веб-сайтів. Поширене використання систем управління вмістом з відкритим кодом та систем управління корпоративним контентом розширило вплив веб-розробки на онлайн-взаємодію та спілкування.

Розробка веб-сайтів також вплинула на особисті мережі та маркетинг. Веб-сайти вже не є просто інструментами для роботи чи комерції, а ширше служать для спілкування та соціальних мереж. Такі веб-сайти, як Facebook та Twitter, надають користувачам платформу для спілкування, а організації - більш особистий та інтерактивний спосіб залучення громадськості.

Веб-розробка враховує багато міркувань безпеки, таких як перевірка помилок при введенні даних через форми, фільтрація вихідних даних та шифрування. Шкідливі практики, такі як введення SQL, можуть виконуватися користувачами з ненавмисними намірами, але лише з примітивними знаннями

про веб-розробку в цілому. Сценарії можна використовувати для використання веб-сайтів, надаючи несанкціонований доступ зловмисним користувачам, які намагаються збирати таку інформацію, як адреси електронної пошти, паролі та захищений вміст, наприклад номери кредитних карток.

Дещо з цього залежить від серверного середовища, на якому запущена мова сценаріїв, наприклад ASP, JSP, PHP, Python, Perl або Ruby, і тому не обов'язково підтримувати самого розробника. Однак рекомендується жорстке тестування веб-додатків перед публічним випуском, щоб запобігти подібним подвигам. Якщо на веб-сайті є якась контактна форма, вона повинна включати в неї поле captcha, яке не дозволяє комп’ютерним програмам автоматично заповнювати форми, а також розсилати пошту.

Захист веб-сервера від вторгнень часто називають зміцненням порту сервера. Багато технологій застосовуються для захисту інформації в Інтернеті, коли вона передається з одного місця в інше. Наприклад, сертифікати TLS (або "сертифікати SSL") видаються органами сертифікації для запобігання шахрайству в Інтернеті. Багато розробників часто використовують різні форми шифрування при передачі та зберіганні конфіденційної інформації. Базове розуміння питань безпеки інформаційних технологій часто є частиною знань веб-розробника.

Оскільки нові веб-програми виявляють нові діри в безпеці навіть після тестування та запуску, оновлення виправлень безпеки часто трапляються для широко використовуваних програм. Часто робота веб-розробників - постійно оновлювати програми, коли випускаються виправлення безпеки та виявляються нові проблеми безпеки.

1.3. Електронний журнал

Електронна книга оцінок - це онлайн-запис учителя про уроки, завдання, успішність та оцінки своїх учнів. Електронна книга оцінок взаємодіє з інформаційною системою учнів, в якій зберігаються записи учнів шкільного округу, включаючи оцінки, медичні картки відвідуваності, стенограми, графіки учнів та інші дані. Деякі книги з електронними оцінками роблять оцінки, домашні завдання та графіки учнів доступними в Інтернеті для батьків та учнів.

Деякі приклади послуг, що надають підручники, - TAMO, Gradealyzer, Spiral Universe, QuickSchools.com та GPA Teacher. Все це забезпечує простий спосіб оновлення інформації про оцінки кожного студента, а також можливість швидкого та легкого перенесення цієї інформації до звітних карток, що закінчуються.

У 2010 році британське урядове агентство з питань ІКТ в освіті, BECTA, запровадило вимогу до звітних карток для всіх учнів загальноосвітньої системи, щоб їхні звіти були доступними для батьків через Інтернет.

2. ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. JavaScript

JavaScript (/ ˈdʒɑːvəˌskrɪpt /), [8] часто скорочується як JS, є мовою програмування, що відповідає специфікації ECMAScript. [9] JavaScript є високорівневим, часто вчасно скомпільованим і багатопарадигмальним. Він має синтаксис фігурних дужок, динамічне введення тексту, орієнтацію на об’єкти на основі прототипу та функції першого класу.

Поряд з HTML та CSS, JavaScript є однією з основних технологій Всесвітньої мережі. [10] Понад 97% веб-сайтів використовують його на

стороні клієнта для поведінки веб-сторінок [11], часто включаючи сторонні бібліотеки. [12] Усі основні веб-браузери мають спеціальний механізм JavaScript для виконання коду на пристрої користувача.

Як мова багатопарадигми, JavaScript підтримує керовані подіями, функціональні та імперативні стилі програмування. Він має інтерфейси програмування програм (API) для роботи з текстом, датами, регулярними виразами, стандартними структурами даних та об’єктною моделлю документа (DOM).

Стандарт ECMAScript не включає введення / виведення (введення / виведення), наприклад, мережеві, сховища чи графічні засоби. На практиці веб-браузер або інша система виконання забезпечує API API для вводу-виводу.

Спочатку двигуни JavaScript використовувались лише у веб-браузерах, але зараз вони є основними компонентами інших програмних систем, зокрема серверів та різноманітних додатків.

Амбітна робота над мовою продовжувалась протягом декількох років, завершившись великою колекцією доповнень та уточнень, яка була оформлена публікацією ECMAScript 6 у 2015 році [26].

Наразі проект специфікації відкрито зберігається на GitHub, а видання ECMAScript видаються за допомогою регулярних щорічних знімків. [27] Потенційний перегляд мови перевіряється шляхом всебічного процесу подання пропозицій. [28] [29] Тепер замість номерів видань розробники перевіряють статус майбутніх функцій індивідуально. [27]

Поточна екосистема JavaScript має безліч бібліотек та фреймворків, усталену практику програмування та розширене використання JavaScript поза веб-браузерами. Плюс, із зростанням односторінкових додатків та інших веб-сайтів, важких для JavaScript, було створено ряд транспіляторів для сприяння процесу розробки. [30]

Використання JavaScript розширилось за межі своїх веб-браузерів. Двигуни JavaScript тепер вбудовані в безліч інших програмних систем, як для розгортання веб-сайтів на стороні сервера, так і для не браузерних програм.

Початкові спроби сприяти використанню JavaScript на стороні сервера були Netscape Enterprise Server та Інтернет-інформаційні служби Microsoft, [34] [35], але вони були невеликими нішами. [36] Використання на стороні сервера врешті-решт почало зростати в кінці 2000-х років, із створенням Node.js та інших підходів. [36]

Electron, Cordova, React Native та інші фреймворки додатків були використані для створення багатьох додатків з поведінкою, реалізованою в JavaScript. Інші програми, що не належать до браузера, включають підтримку Adobe Acrobat для створення сценаріїв документів PDF [37] та розширення оболонки GNOME, написані на JavaScript. [38]

JavaScript нещодавно почав з'являтися в деяких вбудованих системах, як правило, за допомогою Node.js. [39] [40] [41]

JavaScript і DOM дають можливість зловмисним авторам доставляти сценарії для запуску на клієнтському комп'ютері через Інтернет. Автори браузера мінімізують цей ризик, використовуючи два обмеження. По-перше, сценарії працюють у пісочниці, в якій вони можуть виконувати лише дії, пов'язані з Інтернетом, а не загальні завдання програмування, такі як створення файлів. По-друге, сценарії обмежуються політикою того самого походження: скрипти з одного веб-сайту не мають доступу до такої інформації, як імена користувачів, паролі чи файли cookie, що надсилаються на інший сайт. Більшість помилок безпеки, пов’язаних з JavaScript, є порушенням тієї самої політики походження або пісочниці.

Є підмножини загальних JavaScript - ADsafe, Secure ECMAScript (SES) - які забезпечують більший рівень безпеки, особливо для коду, створеного третіми сторонами (наприклад, реклами). [69] [70] Caja - ще один проект для безпечного вбудовування та ізоляції сторонніх JavaScript та HTML.

Політика безпеки вмісту - основний передбачуваний метод забезпечення того, щоб на веб-сторінці виконувався лише надійний код.

Поширеною проблемою безпеки, пов'язаної з JavaScript, є міжсайтовий сценарій (XSS), порушення політики того самого походження. Уразливості XSS виникають, коли зловмисник може змусити цільовий веб-сайт, наприклад веб-сайт Інтернет-банкінгу, включити шкідливий сценарій у веб-сторінку, представлену жертві. Потім сценарій у цьому прикладі може отримати доступ до банківської програми з привілеями жертви, потенційно розкриваючи секретну інформацію або переказуючи гроші без дозволу жертви. Рішенням вразливостей XSS є використання екрануючого HTML при відображенні ненадійних даних.

Деякі браузери включають частковий захист від відображених атак XSS, в якому зловмисник надає URL-адресу, включаючи шкідливий сценарій. Однак навіть користувачі цих браузерів вразливі до інших атак XSS, таких як ті, де шкідливий код зберігається в базі даних. Тільки правильний дизайн веб-додатків на стороні сервера може повністю запобігти XSS.

Уразливості XSS також можуть виникати через помилки впровадження авторами браузера. [71]

Ще однією уразливістю між сайтами є підробка між запитами (CSRF). У CSRF код на сайті зловмисника змушує браузер жертви вчинити дії, які користувач не передбачав на цільовому сайті (наприклад, переказувати гроші в банку). Коли цільові сайти покладаються виключно на файли cookie для автентифікації запитів, запити, що походять із коду на сайті зловмисника, можуть мати однакові дійсні дані для входу користувача-ініціатора. Загалом, рішення CSRF полягає у тому, щоб вимагати значення автентифікації у прихованому полі форми, а не лише у файлах cookie, для автентифікації будь-якого запиту, який може мати тривалі наслідки. Також може допомогти перевірка заголовка HTTP Referrer.

"Викрадення JavaScript" - це тип атаки CSRF, при якому тег <script> на сайті зловмисника експлуатує сторінку на сайті жертви, яка повертає приватну інформацію, таку як JSON або JavaScript. Можливі рішення включають:

вимагання маркера автентифікації в параметрах POST і GET для будь-якої відповіді, яка повертає приватну інформацію.

Розробники клієнт-серверних програм повинні усвідомити, що ненадійні клієнти можуть перебувати під контролем зловмисників. Автор програми не може припустити, що їх код JavaScript працюватиме належним чином (або взагалі), оскільки будь-яка таємниця, вбудована в код, може бути вилучена визначеним супротивником. Деякі наслідки:

Автори веб-сайтів не можуть ідеально приховати, як працює їх JavaScript, оскільки вихідний код повинен надсилатися клієнту. Код можна заплутати, але затухання можна змінити.

Перевірка форми JavaScript забезпечує лише зручність для користувачів, а не безпеку. Якщо сайт підтверджує, що користувач погодився з його умовами обслуговування, або фільтрує недійсні символи з полів, які повинні містити лише цифри, він повинен це робити на сервері, а не лише на клієнті.

Сценарії можна вибірково відключити, тому на JavaScript не можна покладатися, щоб запобігти таким операціям, як клацання правою кнопкою миші на зображенні, щоб зберегти його. [72]

Вбудовувати в JavaScript конфіденційну інформацію, таку як паролі, вважається дуже поганою практикою, оскільки її може отримати зловмисник [73].

Системи управління пакетами, такі як npm та Bower, популярні серед розробників JavaScript. Такі системи дозволяють розробнику легко управляти залежностями своєї програми від інших бібліотек програм розробника. Розробники вірять, що працівники бібліотек забезпечуватимуть їх безпеку та оновлення, але це не завжди так. Через цю сліпу довіру виникла вразливість. Бібліотеки, на які покладаються, можуть мати нові випуски, які спричиняють появу помилок або уразливостей у всіх програмах, які покладаються на

бібліотеки. І навпаки, бібліотека може не виправлятися з відомими вразливими місцями в дикій природі. В ході дослідження, проведеного на вибірці з 133 тис. Веб-сайтів, дослідники виявили, що 37% веб-сайтів включали бібліотеку з принаймні однією відомою вразливістю [74]. "Середнє відставання між найстарішою версією бібліотеки, що використовується на кожному веб-сайті, та найновішою доступною версією цієї бібліотеки становить 1177 днів у ALEXA, а розвиток деяких бібліотек, які все ще активно використовуються, припинився роками тому". [74] Інша можливість полягає в тому, що супровідник бібліотеки може повністю видалити її. Це сталося в березні 2016 року, коли Азер Кочулу видалив своє сховище з npm. Це призвело до поломки всіх десятків тисяч програм та веб-сайтів, залежно від його бібліотек. [75] [76]

JavaScript надає інтерфейс для широкого спектру можливостей браузера, деякі з яких можуть мати такі недоліки, як переповнення буфера. Ці недоліки можуть дозволити зловмисникам писати сценарії, які запускатимуть будь-який бажаний код в системі користувача. Цей код жодним чином не обмежується іншим додатком JavaScript. Наприклад, експлойт перевищення буфера може дозволити зловмиснику отримати доступ до API операційної системи з привілеями суперкористувача.

Ці недоліки вплинули на основні браузери, включаючи Firefox, [77] Internet Explorer, [78] і Safari. [79]

Плагіни, такі як відеоплеєри, Adobe Flash та широкий спектр елементів керування ActiveX, увімкнених за замовчуванням у Microsoft Internet Explorer, також можуть мати недоліки, які можна використати за допомогою JavaScript (такі недоліки використовувались у минулому). [80] [81]

У Windows Vista Microsoft намагалася стримувати ризики помилок, таких як переповнення буфера, запускаючи процес Internet Explorer з обмеженими привілеями. [82] Google Chrome аналогічним чином обмежує свої візуатори сторінок власною "пісочницею".

Веб-браузери здатні запускати JavaScript поза пісочницею, маючи привілеї, необхідні, наприклад, для створення або видалення файлів. Такі привілеї не призначені для надання коду з Інтернету.

Неправильне надання привілеїв JavaScript з Інтернету зіграло свою роль у вразливостях як в Internet Explorer [83], так і у Firefox. [84] У Windows XP з пакетом оновлень 2 Microsoft знизила привілеї JScript в Internet Explorer. [85]

Microsoft Windows дозволяє запускати вихідні файли JavaScript на жорсткому диску комп'ютера як загальні програми, що не містять середовища (див .: Windows Script Host). Це робить JavaScript (як і VBScript) теоретично життєздатним вектором для троянського коня, хоча троянські коні JavaScript на практиці трапляються рідко. [86]

2.2. React

React (також відомий як React.js або ReactJS) - це бібліотека JavaScript із відкритим кодом, інтерфейс, для створення інтерфейсів користувача або компонентів інтерфейсу. Він підтримується Facebook та спільнотою окремих розробників та компаній. [4] [5] [6] React можна використовувати як основу при розробці односторінкових або мобільних додатків. Однак React займається лише управлінням станом та наданням цього стану в DOM, тому для створення програм React зазвичай потрібно використовувати додаткові бібліотеки для маршрутизації, а також певну функціональність на стороні клієнта. [7]

Далі наведено елементарний приклад використання React у HTML із JSX та JavaScript.

```
<div id="myReactApp"></div>

<script type="text/babel">
  function Greeter(props) {
```

```
  return <h1>{props.greeting}</h1>
 }
 var App = <Greeter greeting="Hello World!" />;
 ReactDOM.render(App, document.getElementById('myReactApp'));
</script>
```

Функція Greeter - це компонент React, який приймає привітання властивості. Змінна App є екземпляром компонента Greeter, де для властивості привітання встановлено значення «Hello World!». Потім метод ReactDOM.render відображає наш компонент Greeter всередині елемента DOM з ідентифікатором myReactApp.

При відображенні у веб-браузері результат буде

```
<div id = "myReactApp">
  <h1> Hello World! </h1>
</div>
```

Код реагування складається з сутностей, які називаються компонентами. Компоненти можуть бути відтворені до певного елемента в DOM за допомогою бібліотеки React DOM. При візуалізації компонента можна передавати значення, які відомі як "реквізит": [8]

```
ReactDOM.render(<Greeter greeting="Hello World!" />, document.getElementById('myReactApp'));
```

Ще однією помітною особливістю є використання віртуальної об'єктної моделі документа або віртуальної DOM. React створює кеш-структуру даних в пам’яті, обчислює отримані різниці, а потім ефективно оновлює відображуваний DOM браузера. [9] Цей процес називається примиренням. Це дозволяє програмісту писати код так, ніби вся сторінка відображається при кожній зміні, тоді як бібліотеки React відображають лише підкомпоненти, які насправді змінюються. Цей вибірковий рендеринг забезпечує значне підвищення продуктивності. [Потрібне цитування] Це економить зусилля з перерахунку стилю CSS, макета для сторінки та рендерингу для всієї сторінки.

Методи життєвого циклу використовують форму підключення, яка дозволяє виконувати код у задані точки протягом життя компонента.

- shouldComponentUpdate дозволяє розробнику запобігати непотрібному повторному рендерингу компонента, повертаючи значення false, якщо візуалізація не потрібна.
- componentDidMount викликається, як тільки компонент "змонтований" (компонент створюється в інтерфейсі користувача, часто шляхом асоціювання його з вузлом DOM). Це зазвичай використовується для запуску завантаження даних з віддаленого джерела через API.
- componentWillUnmount викликається безпосередньо перед тим, як компонент буде зруйнований або "демонтований". Це зазвичай використовується для очищення залежностей від компонента, що вимагають ресурсів, і які не будуть просто видалені при демонтажі компонента (наприклад, видалення будь-яких екземплярів setInterval (), які пов'язані з компонентом, або "eventListener", встановленого на " документ "через наявність компонента)
- візуалізація - це найважливіший метод життєвого циклу і єдиний необхідний у будь-якому компоненті. Зазвичай він викликається кожного разу, коли стан компонента оновлюється, що повинно відображатися в інтерфейсі користувача.

Для підтримки концепції React про односпрямований потік даних (який може бути протиставлений двонаправленому потоку AngularJS), архітектура Flux була розроблена як альтернатива популярній архітектурі модель-вигляд-контролер. Flux має дії, які надсилаються через центральний диспетчер до магазину, а зміни в магазині передаються назад до подання. [25] При використанні з React це розповсюдження здійснюється завдяки властивостям компонентів. З моменту своєї концепції Flux був замінений такими бібліотеками, як Redux та MobX. [26]

Потік можна вважати варіантом моделі спостерігача. [27]

Компонент React під архітектурою Flux не повинен безпосередньо модифікувати будь-який переданий йому реквізит, але повинен передавати функції зворотного виклику, які створюють дії, які відправляються диспетчером для модифікації сховища. Дія - це об'єкт, відповідальність якого полягає в описі того, що відбулося: наприклад, дія, що описує одного користувача, "що слідує" за іншим, може містити ідентифікатор користувача, цільовий ідентифікатор користувача та тип USER_FOLLOWED_ANOTHER_USER. [28] Магазини, які можна сприймати як моделі, можуть змінити себе у відповідь на дії, отримані від диспетчера.

Ця закономірність іноді виражається як "властивості стікають вниз, дії течуть вгору". З моменту його створення було створено багато реалізацій Flux, мабуть, найвідомішим є Redux, який має єдиний магазин, який часто називають єдиним джерелом істини. [29]

React був створений Джорданом Уолком, інженером-програмістом у Facebook, який випустив ранній прототип React під назвою "FaxJS". [33] [34] На нього вплинув XHP, бібліотека HTML-компонентів для PHP. Вперше він був розміщений на стрічці новин Facebook у 2011 році, а пізніше в Instagram у 2012 році [35]. Він був відкритий у JSConf US у травні 2013 р. [34]

React Native, який дозволяє розробляти власні Android, iOS та UWP разом з React, був анонсований на React Conf у Facebook у лютому 2015 року та відкритим у березні 2015 року.

18 квітня 2017 року Facebook оголосив React Fiber, новий базовий алгоритм бібліотеки React для побудови користувальницьких інтерфейсів. [36] React Fiber повинен був стати основою будь-яких майбутніх удосконалень та розвитку функцій бібліотеки React. [37] [потребує оновлення]

26 вересня 2017 року React 16.0 був випущений для широкого загалу. [38]

16 лютого 2019 року React 16.8 був оприлюднений для громадськості. [39] Випуск представив React Hooks. [40]

10 серпня 2020 року команда React оголосила першого кандидата на випуск React v17.0, який відзначається першим серйозним випуском без серйозних змін в API-інтерфейсі React для розробників. [41]

2.3. Вибір середовища розробки

JetBrains s.r.o. (раніше IntelliJ Software s.r.o.) - чеська [2] компанія, що розробляє програмне забезпечення, інструменти якої орієнтовані на розробників програмного забезпечення та менеджерів проектів. [3] [4] Станом на 2019 рік компанія має офіси в Празі, Санкт-Петербурзі, Москві, Мюнхені, Бостоні, Новосибірську, Амстердамі, Фостер-Сіті та Марлтоні, Нью-Джерсі. [5] [6] [7] [8] [9]

Компанія пропонує розширене сімейство інтегрованих середовищ розробки (IDE) для мов програмування Java, Groovy, Kotlin, Ruby, Python, PHP, C, Objective-C, C ++, C #, Go, [10] JavaScript і SQL. Компанія увійшла в нову сферу в 2011 році, коли представила Kotlin, мову програмування, яка може працювати на віртуальній машині Java (JVM).

Журнал InfoWorld присудив фірмі "Нагорода за технологію року" у 2011 та 2015 роках.

WebStorm IDE — інтегроване середовище розробки для розробки веб, JavaScript та TypeScript. Багато інших IDE JetBrains включають набір функцій WebStorm через плагіни.

2.4. Каскадні таблиці стилів CSS

Каскадні таблиці стилів (CSS) - це мова таблиці стилів, що використовується для опису презентації документа, написаного мовою розмітки, такою як HTML. CSS - це наріжна технологія Всесвітньої павутини, поряд із HTML та JavaScript.

CSS призначений для розділення презентації та вмісту, включаючи макет, кольори та шрифти. Це розділення може покращити доступність вмісту, забезпечити більшу гнучкість та контроль у специфікації характеристик презентації, дозволити спільному форматуванню декілька веб-сторінок, вказавши відповідний CSS в окремому файлі .css, що зменшує складність та повторюваність структурного вмісту, а також дозволяє файл .css, який буде кешовано, щоб покращити швидкість завантаження сторінки між сторінками, що ділять файл, та його форматування.

Поділ форматування та вмісту також робить можливим подання однієї і тієї ж сторінки розмітки в різних стилях для різних методів візуалізації, таких як екран, друк, голосом (через мовний браузер або зчитувач екрана) та на основі Брайля тактильні пристосування. CSS також має правила альтернативного форматування, якщо доступ до вмісту здійснюється на мобільному пристрої. Каскадне ім'я походить із зазначеної пріоритетної схеми, щоб визначити, яке правило стилю застосовується, якщо більше одного правила відповідає певному елементу. Ця схема каскадного пріоритету передбачувана.

Специфікації CSS підтримуються Консорціумом World Wide Web (W3C). Текст / css типу Інтернет-носія (тип MIME) зареєстровано для використання з CSS RFC 2318 (березень 1998 р.). W3C забезпечує безкоштовну службу перевірки CSS для документів CSS.

На додаток до HTML, інші мови розмітки підтримують використання CSS, включаючи XHTML, звичайний XML, SVG та XUL.

CSS має простий синтаксис і використовує ряд англійських ключових слів, щоб вказати імена різних властивостей стилю.

Таблиця стилів складається зі списку правил. Кожне правило або набір правил складається з одного або декількох селекторів та блоку оголошень.

У CSS селектори оголошують, до якої частини розмітки застосовується стиль, збігаючи теги та атрибути в самій розмітці.

Селектори можуть застосовувати такі документи:

- всі елементи певного типу, напр. заголовки другого рівня h2
- елементи, визначені атрибутом, зокрема:
    - id: унікальний ідентифікатор у документі, ідентифікований префіксом хешу, наприклад #id
    - клас: ідентифікатор, який може анотувати кілька елементів у документі, ідентифікований префіксом точки, наприклад .classname
- елементи залежно від того, як вони розміщені щодо інших у дереві документів.

Класи та ідентифікатори чутливі до регістру, починаються з літер і можуть містити буквено-цифрові символи, дефіси та підкреслення. Клас може застосовуватися до будь-якої кількості екземплярів будь-яких елементів. Ідентифікатор може застосовуватися лише до одного елемента.

Псевдокласи використовуються в селекторах CSS, щоб дозволити форматування на основі інформації, яка не міститься в дереві документів. Одним із прикладів широко використовуваного псевдокласу є: hover, який визначає вміст лише тоді, коли користувач "вказує" на видимий елемент, як правило, утримуючи над ним курсор миші. Він додається до селектора, як у: hover або #elementid: hover. Псевдо-клас класифікує елементи документа, такі як: link або: відвідується, тоді як псевдо-елемент робить виділення, яке може складатися з часткових елементів, таких як :: first-line або :: first-letter.

Селектори можуть поєднуватися різними способами для досягнення великої специфічності та гнучкості. Кілька селекторів можуть бути об’єднані в рознесений список, щоб вказати елементи за розташуванням, типом елемента, ідентифікатором, класом або будь-якою їх комбінацією. Порядок селекторів важливий. Наприклад, div .myClass {color: red;} застосовується до всіх елементів класу myClass, що знаходяться всередині елементів div, тоді як .myClass div {color: red;} застосовується до всіх елементів div, що знаходяться всередині елементів класу myClass. Цього не слід плутати з об'єднаними

ідентифікаторами, такими як div.myClass {color: red;}, який застосовується до елементів div класу myClass.

Блок оголошень складається зі списку оголошень у фігурних дужках. Кожна декларація сама складається з властивості, двокрапки (:) та значення. Якщо в блоці є кілька оголошень, для відокремлення кожного оголошення потрібно вставити крапку з комою (;). Може бути використана необов'язкова крапка з комою після останньої (або одиничної) декларації. Властивості вказані в стандарті CSS. Кожна властивість має набір можливих значень. Деякі властивості можуть впливати на будь-який тип елементів, а інші стосуються лише певних груп елементів.

Значеннями можуть бути ключові слова, такі як "центр" або "успадкувати", або числові значення, такі як 200px (200 пікселів), 50vw (50 відсотків ширини області перегляду) або 80% (80 відсотків ширини батьківського елемента). Значення кольору можна вказати за допомогою ключових слів (наприклад, "червоний"), шістнадцяткових значень (наприклад, # FF0000, також скорочено # F00), значень RGB за шкалою від 0 до 255 (наприклад, rgb (255, 0, 0)), значень RGBA які вказують як кольорову, так і альфа-прозорість (наприклад, rgba (255, 0, 0, 0.8)), або значення HSL або HSLA (наприклад, hsl (000, 100%, 50%), hsla (000, 100%, 50%, 80 %)).

Ненульові числові значення, що представляють лінійні міри, повинні включати одиницю довжини, яка є або алфавітним кодом, або абревіатурою, як у 200px або 50vw; або знак відсотка, як у 80%. Деякі одиниці - см (сантиметр); дюйми (дюйми); мм (міліметр); шт (піка); і pt (точка) - абсолютні, що означає, що розмір, що відображається, не залежить від структури сторінки; інші - em (em); ex (ex) та px (pixel) - відносні, що означає, що такі фактори, як розмір шрифту батьківського елемента, можуть впливати на відтворене вимірювання. Ці вісім одиниць були властивістю CSS 1 і зберігались у всіх наступних переробках. Запропонований модуль значень та одиниць CSS Рівень 3, якщо буде прийнятий як рекомендація W3C, надасть ще сім одиниць довжини: ch; Q; rem; vh; vmax; vmin; та vw.

До CSS майже всі презентаційні атрибути HTML-документів містилися в HTML-розмітці. Всі кольори шрифту, стилі тла, вирівнювання елементів, межі та розміри мали бути чітко описані, часто неодноразово, в HTML. CSS дозволяє авторам перенести більшу частину цієї інформації в інший файл, таблицю стилів, в результаті чого HTML стає значно простішим.

Наприклад, заголовки (елементи h1), підзаголовки (h2), підзаголовки (h3) тощо визначаються структурно за допомогою HTML. У друку та на екрані вибір шрифту, розміру, кольору та акценту для цих елементів є презентаційним. До CSS автори документів, які хотіли присвоїти такі типографічні характеристики, скажімо, всім заголовкам h2, повинні були повторювати розмітку презентації HTML для кожного входження цього типу заголовка. Це зробило документи більш складними, більшими, більш схильними до помилок і важкими в обслуговуванні. CSS дозволяє відокремлювати презентацію від структури. CSS може визначати колір, шрифт, вирівнювання тексту, розмір, межі, інтервали, макет та багато інших типографічних характеристик, і може робити це самостійно для екранних та друкованих подань. CSS також визначає невізуальні стилі, такі як швидкість читання та акценти для звукових читачів тексту. Зараз W3C не підтримує використання всієї презентаційної розмітки HTML.

Наприклад, під HTML до CSS елемент заголовка, визначений червоним текстом, буде записаний як:

<h1> Глава 1. </h1>

Використовуючи CSS, один і той же елемент можна закодувати, використовуючи властивості стилю замість презентаційних атрибутів HTML:

<h1 style = "color: red;"> Глава 1. </h1>

Переваги цього можуть бути не одразу ясними, але сила CSS стає більш очевидною, коли властивості стилю розміщуються у внутрішньому елементі стилю або, ще краще, у зовнішньому файлі CSS. Наприклад, припустимо, документ містить елемент стилю:

< style >

```
    h1 {
        колір: червоний;
    }
</style>
```

Тоді всі елементи h1 у документі автоматично стануть червоними, не вимагаючи явного коду. Якщо згодом автор хотів зробити елементи h1 синіми, це можна зробити, змінивши елемент стилю на:

```
< style >
    h1 {
        колір: синій;
    }
</style>
```

а не шляхом копіткого перегляду документа та зміни кольору для кожного окремого елемента h1.

Стилі також можна помістити у зовнішній файл CSS, як описано нижче, і завантажити, використовуючи синтаксис, подібний до:

<link href = "path / to / file.css" rel = "stylesheet" type = "text / css">

Це додатково відокремлює стиль від документа HTML і дає змогу переробляти стилі декількох документів, просто редагуючи спільний зовнішній файл CSS.

Інформація про CSS може бути надана з різних джерел. Такими джерелами можуть бути веб-браузер, користувач та автор. Інформацію від автора можна класифікувати вбудовано, тип носія, важливість, специфічність селектора, порядок правил, успадкування та визначення властивостей. Інформація про стиль CSS може бути в окремому документі або вбудована в документ HTML. Можна імпортувати декілька таблиць стилів. Залежно від використовуваного вихідного пристрою можуть застосовуватися різні стилі; наприклад, екранна версія може сильно відрізнятися від друкованої версії, так що автори можуть адаптувати презентацію відповідно до кожного носія.

Таблиця стилів з найвищим пріоритетом контролює відображення вмісту. Декларації, не встановлені у джерелі з найвищим пріоритетом, передаються джерелу з нижчим пріоритетом, наприклад стилю агента користувача. Процес називається каскадним.

Однією з цілей CSS є надання користувачам більшого контролю над презентацією. Хтось, кому важко читати червоні курсивні заголовки, може застосувати іншу таблицю стилів. Залежно від браузера та веб-сайту, користувач може вибирати з різних таблиць стилів, наданих дизайнерами, або може видаляти всі додані стилі та переглядати сайт, використовуючи стиль браузера за замовчуванням, або може замінювати лише стиль червоного курсиву, не змінюючи атрибути.

Спадкування є ключовою особливістю CSS; він покладається на взаємозв'язок предок-нащадок. Спадкування - це механізм, за допомогою якого властивості застосовуються не лише до зазначеного елемента, але і до його нащадків. Спадщина спирається на дерево документів, яке є ієрархією елементів XHTML на сторінці на основі вкладеності. Нащадкові елементи можуть успадковувати значення властивостей CSS від будь-якого елемента-предка, що їх включає. Як правило, нащадні елементи успадковують властивості, пов’язані з текстом, але властивості, пов’язані з вікнами, не успадковуються. Властивості, які можна успадкувати, - це колір, шрифт, інтервал між літерами, висота рядка, стиль списку, вирівнювання тексту, відступ тексту, перетворення тексту, видимість, пробіли та пробіли між словами. Властивості, які неможливо успадкувати, це фон, межа, відображення, плаваюче та очищення, висота та ширина, поле, мінімальна та максимальна висота та -ширина, контур, переповнення, відступ, положення, оформлення тексту, вертикальне вирівнювання та z -індекс.

Спадкування може бути використана, щоб уникнути неодноразового декларування певних властивостей у таблиці стилів, дозволяючи коротший CSS.

Спадкування в CSS - це не те саме, що успадкування в мовах програмування на основі класів, де можна визначити клас B як "як клас A, але з модифікаціями". За допомогою CSS можна стилізувати елемент з "класом A, але з модифікаціями". Однак неможливо визначити такий клас CSS B, який потім можна було б використовувати для стилізації декількох елементів без необхідності повторювати модифікації.

CSS вперше був запропонований Håkon Wium Lie 10 жовтня 1994 р. У той час Лі працювала з Тімом Бернерс-Лі у ЦЕРНі. Кілька інших мов таблиць стилів для Інтернету було запропоновано приблизно в той же час, і обговорення публічних списків розсилки та всередині Консорціуму всесвітньої павутини призвели до того, що перша Рекомендація W3C CSS (CSS1) [24] була опублікована в 1996 році. Зокрема, пропозиція Берт Бос мав вплив; він став співавтором CSS1 і вважається співавтором CSS.

Таблиці стилів існують в тій чи іншій формі з часів заснування Стандартної узагальненої мови розмітки (SGML) у 1980-х роках, а CSS був розроблений для надання таблиць стилів для Інтернету. Однією з вимог до мови веб-таблиць стилів було те, щоб таблиці стилів надходили з різних джерел у мережі. Тому існуючі мови таблиць стилів, такі як DSSSL та FOSI, не підходили. CSS, з іншого боку, дозволяє стилю документа впливати на декілька таблиць стилів шляхом "каскадного" стилю.

По мірі зростання HTML він охоплював все більше різноманітних стилістичних можливостей, щоб задовольнити запити веб-розробників. Ця еволюція дала дизайнеру більше контролю над зовнішнім виглядом сайту за рахунок більш складного HTML. Варіації реалізації веб-браузера, такі як ViolaWWW та WorldWideWeb, ускладнювали послідовний зовнішній вигляд сайту, а користувачі мали менший контроль над тим, як відображався веб-вміст. Браузер / редактор, розроблений Тімом Бернерсом-Лі, мав таблиці стилів, які були жорстко закодовані в програмі. Тому таблиці стилів не можуть бути пов'язані з документами в Інтернеті. Роберт Кайо, також із CERN, хотів

відокремити структуру від презентації, щоб різні таблиці стилів могли описувати різні презентації для друку, презентації на екрані та редактори.

Поліпшення можливостей веб-презентацій було темою, яка цікавила багатьох у веб-спільноті, і дев'ять різних мов таблиць стилів було запропоновано в списку розсилки у стилі www. З цих дев’яти пропозицій дві особливо вплинули на те, що стало CSS: каскадні таблиці стилів HTML та пропозиція таблиці стилів на основі потоку (SSP). Два браузери служили тестами для початкових пропозицій; Лі працював з Івом Лафоном над впровадженням CSS у браузері Arena Дейва Реггетта. Берт Бос реалізував власну пропозицію SSP у браузері Argo. Після цього Lie і Bos спільно працювали над розробкою стандарту CSS ("H" було вилучено з назви, оскільки ці таблиці стилів також можна застосовувати до інших мов розмітки, крім HTML).

Пропозиція Лі була представлена на конференції "Мозаїка та Інтернет" (згодом названа WWW2) в Чикаго, штат Іллінойс, у 1994 році, і знову разом з Бертом Босом у 1995 році. [23] Приблизно в цей час W3C вже був створений і зацікавився розробкою CSS. З цією метою він організував семінар-практикум під головуванням Стівена Пембертона. Це призвело до того, що W3C додав роботу над CSS до результатів роботи редакційної комісії HTML-перегляду (ERB). Лі і Бос були основним технічним персоналом у цьому аспекті проекту, в якому брали участь також додаткові члени, в тому числі Томас Рірдон з Microsoft. У серпні 1996 року компанія Netscape Communication Corporation представила альтернативну мову таблиць стилів, що називається JavaScript Sheets Style (JSSS). Специфікація ніколи не була закінчена, і вона застаріла. До кінця 1996 р. CSS був готовий стати офіційним, а Рекомендація CSS рівня 1 була опублікована в грудні.

Розробка HTML, CSS та DOM велася в одній групі - редакційній комісії HTML (ERB). На початку 1997 року ЄБР було розділено на три робочі групи: Робоча група HTML, головою якої став Дан Конноллі з W3C; Робоча група

DOM під головуванням Лорен Вуд із SoftQuad; та робоча група CSS під головуванням Кріса Ліллі з W3C.

Робоча група CSS розпочала вирішення питань, які не були вирішені на рівні CSS 1, в результаті чого рівень CSS рівня 2 був створений 4 листопада 1997 р. Він був опублікований як рекомендація W3C 12 травня 1998 р. Рівень CSS 3, який був розпочате в 1998 році, все ще знаходиться в стадії розробки станом на 2014 рік.

У 2005 році Робочі групи CSS вирішили суворіше виконувати вимоги до стандартів. Це означало, що вже опубліковані стандарти, такі як CSS 2.1, CSS 3 Selectors та CSS 3 Text, були повернуті з Рекомендацій кандидатів на рівень робочого проекту.

Фреймворки CSS - це заздалегідь підготовлені бібліотеки, які призначені для спрощення стилю веб-сторінок, що відповідають стандартам, з використанням мови каскадних таблиць стилів. Структури CSS включають Blueprint, Bootstrap, Cascade Framework, Foundation та Materialize. Як і бібліотеки мови програмування та сценаріїв, фреймворки CSS зазвичай включаються як зовнішні .css-аркуші, на які посилається HTML-код <head>. Вони надають ряд готових варіантів проектування та розміщення веб-сторінки. Незважаючи на те, що багато з цих фреймворків опубліковано, деякі автори використовують їх здебільшого для швидкого створення прототипів або для вивчення, і вважають за краще "виготовляти" CSS, який підходить для кожного опублікованого сайту, без проектування, обслуговування та завантаження, що має багато невикористаних функцій в стилі сайту.

У міру збільшення розміру ресурсів CSS, що використовуються в проекті, команді розробників часто потрібно прийняти рішення про загальну методологію проектування, щоб тримати їх організованими. Цілями є простота розробки, простота співпраці під час розробки та продуктивність розгорнутих таблиць стилів у браузері. Популярні методології включають OOCSS (об'єктно-орієнтований CSS), ACSS (атомний CSS), oCSS (органічний

каскадний аркуш стилів), SMACSS (масштабована та модульна архітектура для CSS) та BEM (блок, елемент, модифікатор).

3. РОЗРОБКА ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1. Діаграма варіантів використання

Найпростіша діаграма використання - це представлення взаємодії між користувачем та системою, яка показує взаємозв'язок між користувачем та різними видами використання, що беруть участь у користувачі. Діаграми випадків використання можуть ідентифікувати різні типи користувачів системи та різні випадки використання, і часто супроводжуються іншими типами діаграм. Використовуйте кола або еліпси для позначення використання.

Незважаючи на те, що сам випадок використання може детально вивчити кожну можливість, діаграма прикладів може допомогти забезпечити огляд системи більш високого рівня. Я вже говорив, що "план використання - це принцип вашої системи".

Завдяки спрощеному характеру, план використання може бути хорошим інструментом комунікації для зацікавлених сторін. Креслення намагаються імітувати реальний світ та надати зацікавленим сторонам ідеї щодо розвитку системи. Сіау та Лі провели дослідження, щоб визначити, чи взагалі існувала дійсна ситуація для схем використання або вони були непотрібними. Було виявлено, що діаграми випадків використання передають намір системи більш спрощеним чином зацікавленим сторонам і що вони "інтерпретуються більш повно, ніж діаграми класів".

Мета використання діаграм - показати динамічні аспекти системи. Інші схеми та документи можуть бути використані для забезпечення повного функціонального та технічного представлення системи. Вони забезпечують

спрощене графічне представлення того, що система насправді повинна робити.

Елементи:

- рамки системи (англ. system border) - прямоугольник із назвою у верхніх частинах та еліпсами (прецедентами) всередині. Часто може бути опущено без корисної інформації про полезну інформацію,
- актор (англ. актор) - стилізований людський персонаж, обозначаючий набір ролей користувача (розуміється в широкому змісті: людина, внешняя сутність, клас, інша система), взаємодіючого з деякою сутністю (системною, підсистемою, класом). Актори не можуть бути пов'язані між собою з іншим (за вимкнення відносин щодо обробки / дослідження),
- прецедент - еліпс із надписом, що обозначає виконувану систематичну дію (може включати можливі варіанти), що призводить до спостерігаємих акторами результатів. Надпис може бути ім'ям або описом (з точки зору актора) того, "що" робить система (а не "як"). Ім прецедента зв’язано з неперервним (атомарним) сценарієм - конкретною послідовністю дій, ілюстріруючою повіддю. Під час сценарію актори обмінюються із систематичними повідомленнями. Сценарій може бути приведений на діаграмі прецедентів у відео UML-коментарі. З одним прецедентом може бути пов'язано кілька різних сценаріїв

На рисунку 3.1 зображено діаграму варіантів використання, яка описує можливі дії студента в системі.

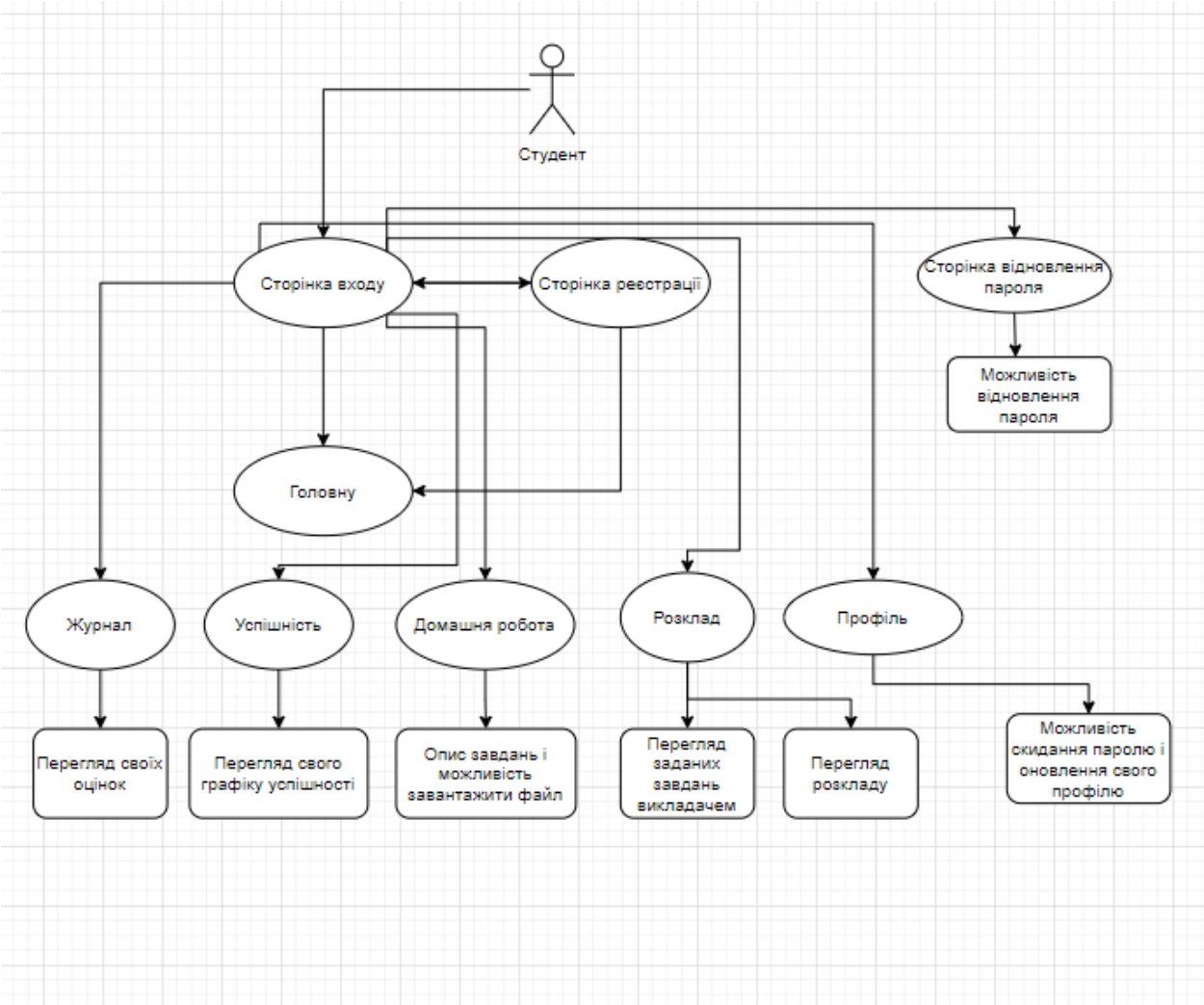


Рис. 3.1 — Діаграма варіантів використання для студента

На рисунку 3.2 зображено діаграму варіантів використання, яка описує можливі дії вчителя в системі.

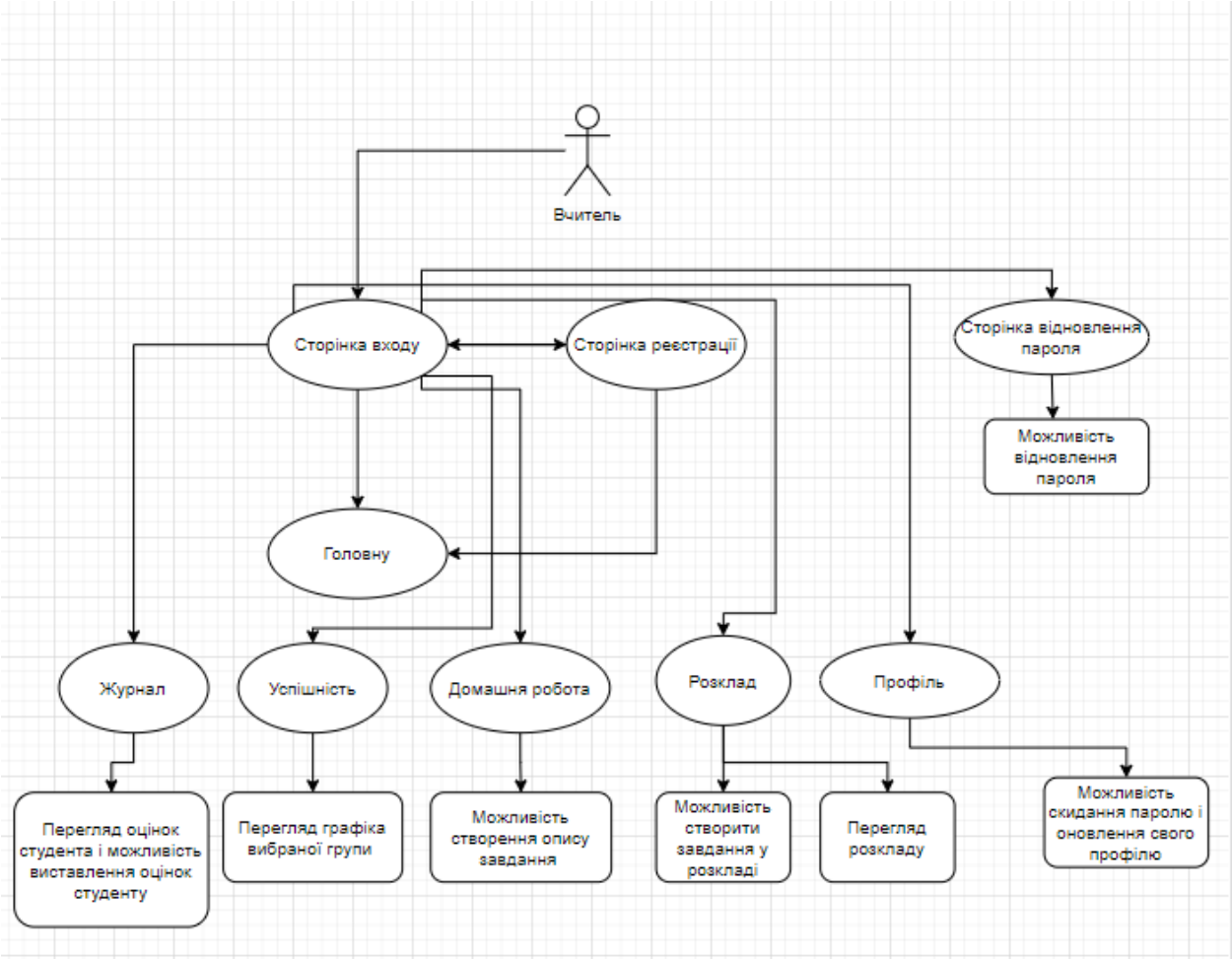


Рис. 3.2 — Діаграма варіантів використання для вчителя

3.2. Сторінка входу

Сторінка входу представляє собою форму авторизації, яка допомагає відсіяти незареєстрованих користувачів, тобто, підвищити безпечність конфіденційної інформації.

В загальному понятті, авторизація - це функція визначення прав / привілеїв доступу до ресурсів, яка пов’язана із загальною інформаційною та комп’ютерною безпекою, а також з управлінням доступом, зокрема. Більш формально "авторизувати" - це визначати політику доступу. Наприклад, кадровий персонал, як правило, уповноважений отримувати доступ до записів працівників, і ця політика часто оформляється як правила контролю доступу в комп'ютерній системі. Під час роботи система використовує правила контролю доступу, щоб вирішити, чи будуть запити на доступ від (аутентифікованих) споживачів схвалені (надані) чи відхилені (відхилені). Ресурси включають окремі файли або дані елемента, комп'ютерні програми, комп'ютерні пристрої та функціональні можливості, що надаються комп'ютерними програмами. Прикладами споживачів є користувачі комп'ютерів, комп'ютерне програмне забезпечення та інше обладнання на комп'ютері.

Зовнішній вигляд фінальної реалізації зображено на рисунках 3.3 (порожня форма) і 3.4 (заповнена форма).

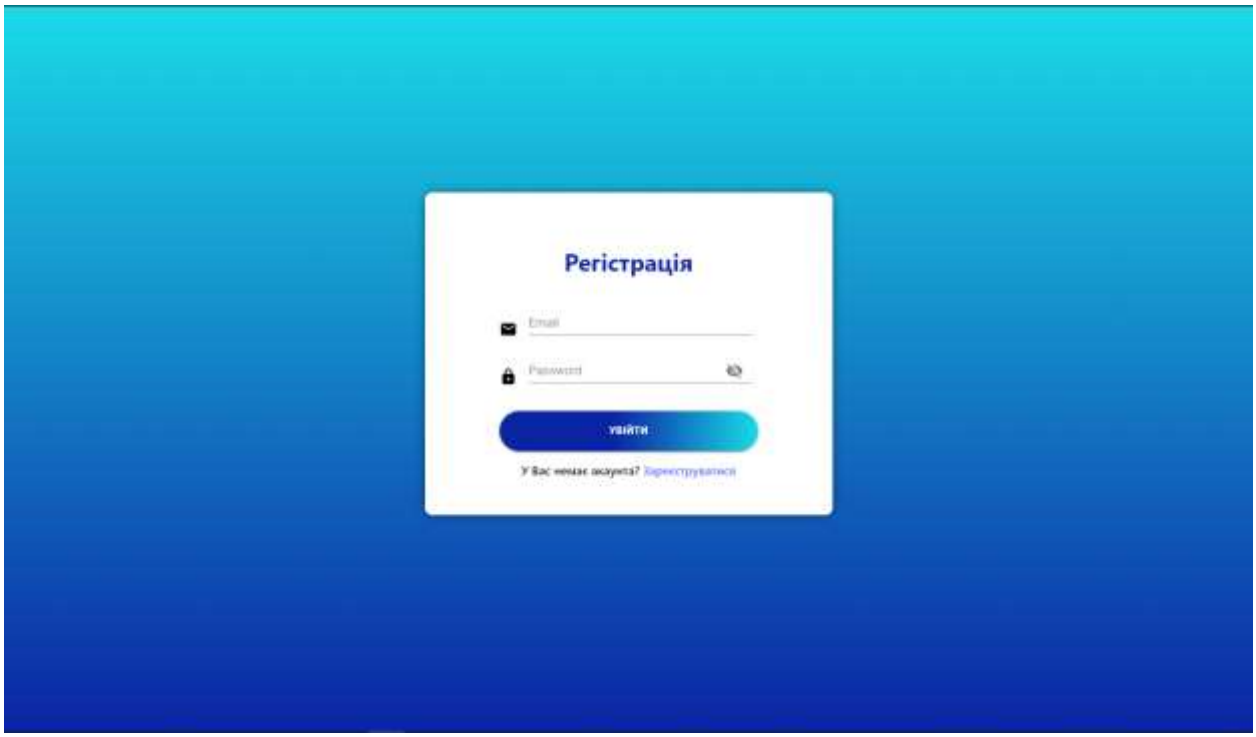


Рис. 3.3 — Порожня форма

Рис. 3.4 — Заповнена форма

3.3. Сторінка реєстрації

Сторінка реєстрації представляє собою форму, яка несе в собі функціонал створення нового облікового запису шляхом додавання його в систему.

В загальному розумінні користувач - це особа, яка користується комп’ютером або послугою мережі. Користувачам комп'ютерних систем та програмних продуктів, як правило, не вистачає технічного досвіду, необхідного для повного розуміння їх роботи. [1] Потужні користувачі використовують розширені функції програм, хоча вони не обов’язково здатні до комп’ютерного програмування та системного адміністрування. [2] [3]

Користувач часто має обліковий запис і ідентифікується системою за допомогою імені користувача (або імені користувача). Інші терміни імені користувача включають ім'я для входу, ім'я екрана (або ім'я екрана), ім'я облікового запису, псевдонім (або псевдонім) та дескриптор, що походить від ідентичного терміна радіостанції для громадян.

Деякі програмні продукти надають послуги іншим системам і не мають прямих кінцевих користувачів.

Зовнішній вигляд фінальної реалізації форми реєстрації зображено на рисунках 3.5 (порожня форма) і 3.6 (заповнена форма).

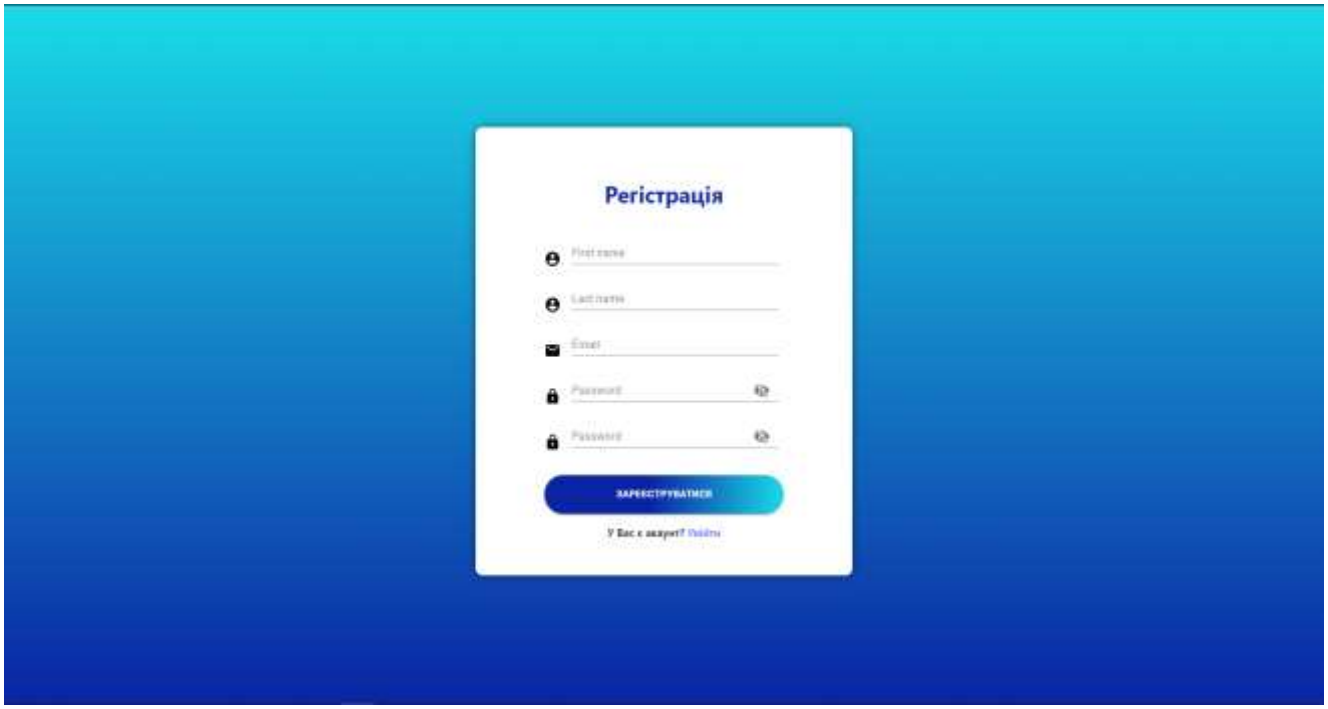


Рис. 3.5 — Порожня форма реєстрації

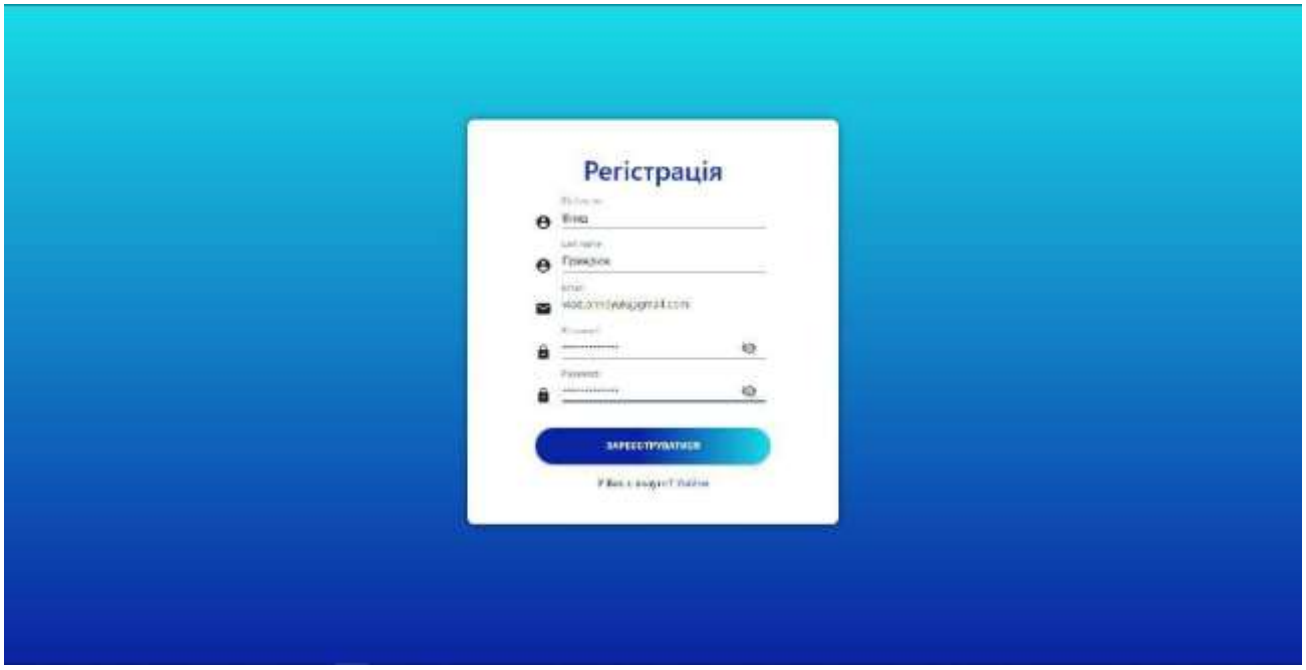


Рис. 3.6 — Заповнена форма реєстрації

3.4. Виставлення оцінок

Сторінка виставлення оцінок представляє собою журнальний список учнів і комірок для оцінок по кожному окремому предмету в табличному вигляді, як це зображенно на рисунку 3.7 (сторінка виставлення оцінок). На сорінці можна подивитися журнал оцінок і дізнатися яка оцінка була виставленна на конкретний проект. А також можна вибрати період за який можна дізнатися свої оцінки, а також середній бал. Вчитель має змогу переглядати оцінки по свому предмету, а також виставляти їх.

Рис. 3.7 — Сторінка виставлення оцінок

3.5. Статистика успішності

Сторінка успішності представляє собою графічне відображення статистики успішності учнів по предметах, представлене у вигляді гістограми, яка зображенна на рисунку 3.8 (сторінка успішності). Також студент має можливість вибору проміжуток дати і конкретного предмету по якому буде відображатися графік. Результати відображаються у вигляді барів – це вертикальні стовпці у графіку. Для швидкого знаходження певного бара, кожний бар має свій колір.

Рис. 3.8 — Сторінка успішності

3.6. Розклад

Сторінка розкладу представляє собою перегляд розкладу на кожний день, тиждень, робочий тиждень і місяць, як це зображенно на рисунку 3.9 (сторінка розкладу). Студент має можливість переглядати яка пара має бути і в якій аудиторії, а також є можливість вибіру розкладу за певним періодом. Вчитель має право виставляти коли, на яку годину, по яких днях і де мають проводитися заннятя. Також має можливість зробити нагадування яке буде відображатися для студентів, щоб не пропустити пару, а ще можна написати опис до пари, тобто що буде або потрібно взяти з собою на пару.

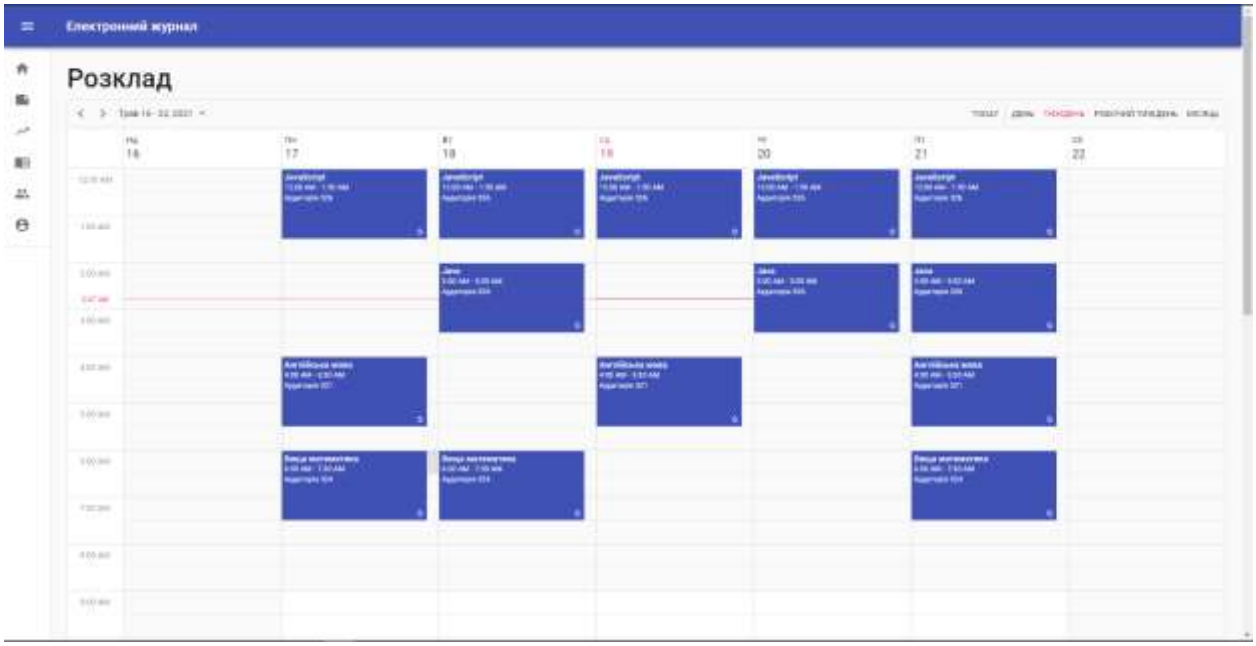


Рис. 3.9 — Сторінка розкладу

3.7. Завдання

Сторінка завдань представляє собою табличне представлення списку завдань, розбите по категоріям, таким як день здачі, предмет, тощо. Студент має можливість переглядати свої завдання на певний період. А вчтель має можливість створювати завдання для студентів.

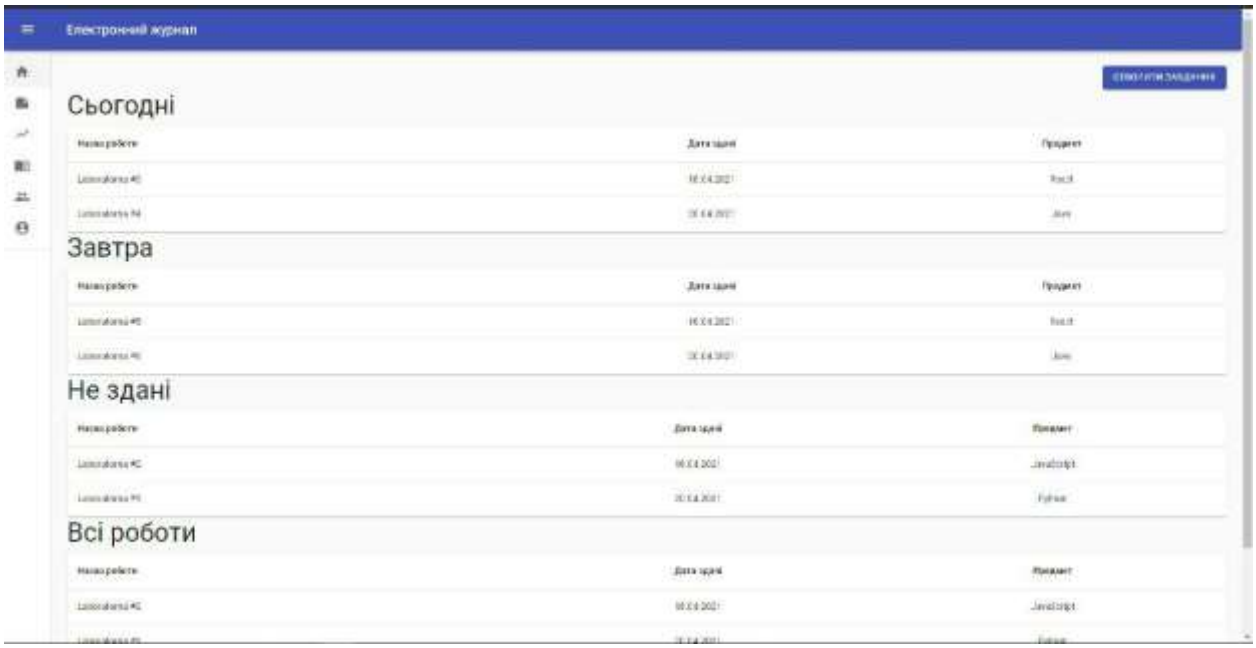


Рис. 3.10 — Сторінка завдань

3.8. Профіль

Сторінка профілю представляє собою графічне відображення облікового записку користувача в читабельній формі. Користувач має змогу змінювати своє ПІБ, почту і пароль, а також фото.

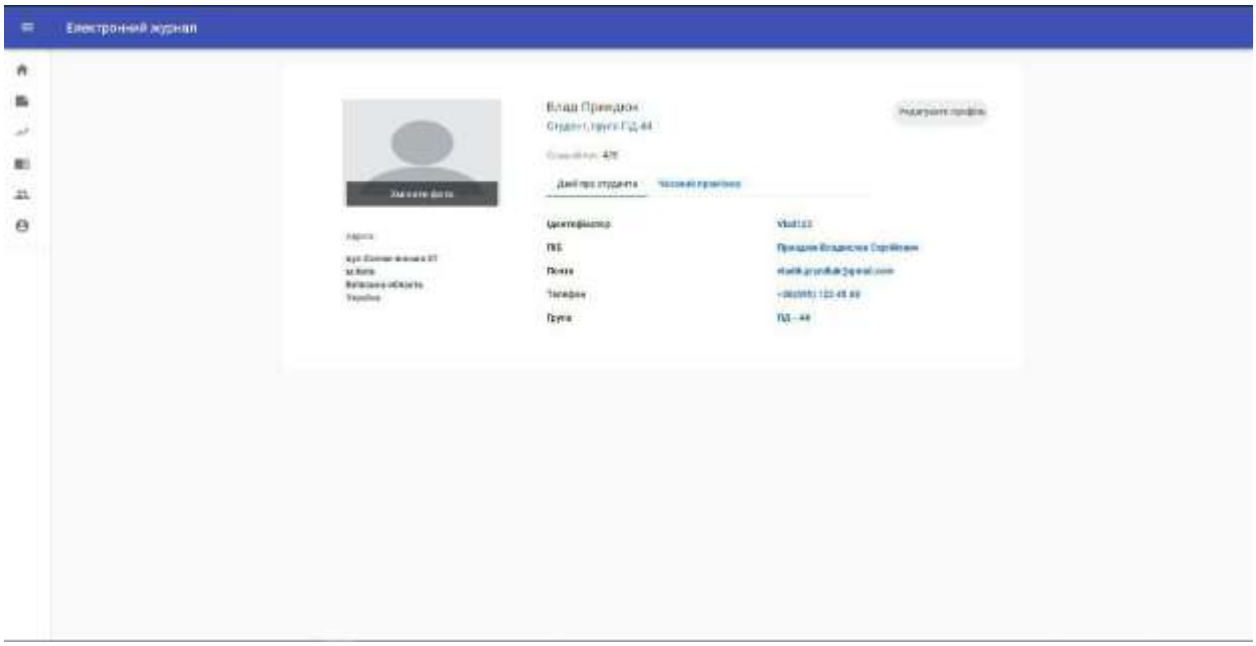


Рис. 3.11 — Сторінка профілю

ВИСНОВКИ

Метою даної роботи стало розроблення UI для інформаційної системи електронного журналу засобами мови програмування JavaScript.

Для досягнення поставленої мети в процесі виконання роботи наступні завдання:

Було проаналізовано загальне розуміння поняття «веб-сайт», виявлено його основні структурні частини, особливості та призначення.

Проведено повний аналіз засобів розробки веб-сайтів, виявлено методи і технології, які використовуються сучасними веб-розробниками для створення і проектування веб-сайтів.

Проведено огляд поняття електронного журналу, виявлено його суть та історичні витоки даного явища.

В якості мови програмування було обрано JavaScript, як мову яка найбільш повною мірою підходить під поставлені задачі. Проведено огляд мови програмування JavaScript.

Проведено огляд технології каскадних таблиць стилів CSS, виявлено основні функції, особливості синтаксису та призначення технології.

В якості середовища розробки обрано WebStorm IDE, інтегроване середовище розробки компанії JetBrains.

Побудовано діаграму варіантів використання для двох ролей користувачів, яка повністю описує можливий функціонал системи для ролей студента і вчителя.

- Розробити сторінку входу
- Розробити сторінку реєстрації
- Розробити сторінку виставлення оцінок
- Розробити сторінку статистики успішності
- Розробити сторінку розкладу
- Розробити сторінку завдань

- Розробити сторінку профілю

Завдяки чіткому виконанню завдань, поставлених на початку роботи, в результаті було отримано повноцінний UI для інформаційної системи електронного журналу, який готовий до використання в реальних умовах у зв’язці з сервером.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "The website of the world's first-ever web server". Retrieved 30 August 2008.
2. Cailliau, Robert. "A Little History of the World Wide Web". Retrieved 16 February 2007.
3. "Internet, Web, and Other Post-Watergate Concerns". University of Chicago. Retrieved 18 September 2010.
4. AP Stylebook [@APStylebook] (16 April 2010). "Responding to reader input, we are changing Web site to website. This appears on Stylebook Online today and in the 2010 book next month" (Tweet). Retrieved 18 March 2019 – via Twitter.
5. "OpenGL ES for the Web". khronos.org. Retrieved 1 April 2019.
6. Pete LePage. "Responsive Web Design Basics | Web". Google Developers. Retrieved 13 March 2017.
7. Perrin, Andrew; Anderson, Monica (10 April 2019). "Social media usage in the U.S. in 2019 | Pew Research Center". PewResearch.Org. Pew Research. Retrieved 20 July 2019. graphic *Study was quoted in Forbes.
8. "Web Server Survey". Netcraft. Retrieved 13 March 2017.
9. A total number of Websites | Internet live stats. internetlivestats.com. Retrieved on 14 April 2015.
10. "Web Server Survey". Netcraft News. Retrieved 17 May 2021.
11. Deon (26 May 2020). "How Many Websites Are There Around the World? [2021]". Siteefy. Retrieved 17 May 2021.
12. "Internet 2009 in numbers". Pingdom. Retrieved 17 May 2021.
13. "Number of internet users worldwide". Statista. Retrieved 17 May 2021.
14. "Internet audiences worldwide 2020". Statista. Retrieved 17 May 2021.
15. "Facebook MAU worldwide 2020". Statista. Retrieved 17 May 2021.

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

“Розробка UI для електронного журналу на мові JavaScript”

Виконав студент 4 курсу
групи ПД-44
Приндюк Владислав Сергійович
Керівник роботи
Негоденко Олена Василівна

Київ – 2021

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** (розроблення UI для інформаційної системи електронного журналу засобами мови програмування JavaScript.)
- **Об’єкт дослідження** (перенесення документообігу в електронний вигляд та автоматизація рутинних процесів.)
- **Предмет дослідження** (UI для інформаційної системи електронного журналу.)

ТЕХНІЧНІ ЗАВДАННЯ

- *1. Провести огляд поняття веб-сайту*
- *2. Провести аналіз методів та засобів розробки веб-сайтів*
- *3. Провести огляд поняття електронного журналу*
- *4. Обрати мову програмування*
- *5. Обрати додаткові інструменти*
- *6. Обрати середовище розробки*
- *7. Побудувати діаграму варіантів використання*
- *8. Розробити сторінку входу*
- *9. Розробити сторінку реєстрації*

3

ТЕХНІЧНІ ЗАВДАННЯ

- *10. Розробити сторінку виставлення оцінок*
- *11. Розробити сторінку статистики успішності*
- *12. Розробити сторінку розкладу*
- *13. Розробити сторінку завдань*
- *14. Розробити сторінку профілю*

4

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

JavaScript - це мова сценаріїв, який дозволяє створювати динамічно оновлюваний контент, управляти мультимедіа, анімувати зображення і практично все інше.

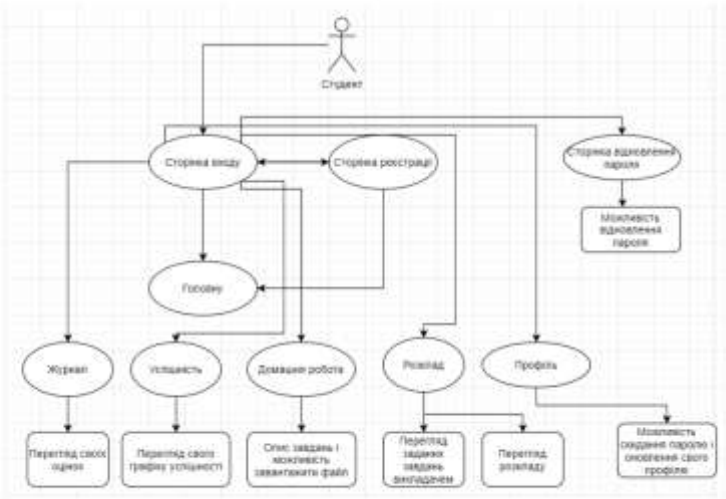
React (також відомий як React.js або ReactJS) - це бібліотека JavaScript із відкритим кодом, інтерфейс, для створення інтерфейсів користувача або компонентів інтерфейсу

WebStorm IDE — інтегроване середовище розробки для розробки веб, JavaScript та TypeScript. Багато інших IDE JetBrains включають набір функцій WebStorm через плагіни

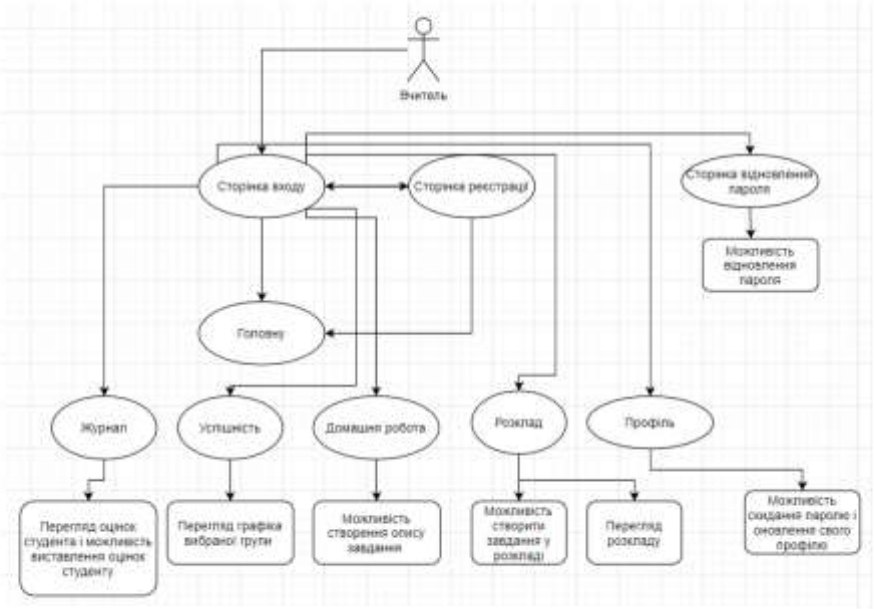
Каскадні таблиці стилів (CSS) - це мова таблиці стилів, що використовується для опису презентації документа, написаного мовою розмітки, такою як HTML

5

МЕТОДИ ТА КЛАСИ ПРОГРАМИ



6



7

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Приндюк В.С. Застосування програмного забезпечення в інфокомунікаційних технологія. Збірник тез. 12.02.2021, ДУТ, м. Київ — К.: ДУТ, 2021. — С. 37.

8

ВИСНОВКИ

Метою даної роботи стало розроблення UI для інформаційної системи електронного журналу засобами мови програмування JavaScript.

Для досягнення поставленої мети в процесі виконання роботи наступні завдання:

Було проаналізовано загальне розуміння поняття «веб-сайт», виявлено його основні структурні частини, особливості та призначення.

Проведено повний аналіз засобів розробки веб-сайтів, виявлено методи і технології, які використовуються сучасними веб-розробниками для створення і проектування веб-сайтів.

Проведено огляд поняття електронного журналу, виявлено його суть та історичні витоки даного явища.

В якості мови програмування було обрано JavaScript, як мову яка найбільш повною мірою підходить під поставлені задачі. Проведено огляд мови програмування JavaScript.

Проведено огляд технології каскадних таблиць стилів CSS, виявлено основні функції, особливості синтаксису та призначення технології.

В якості середовища розробки обрано WebStorm IDE, інтегроване середовище розробки компанії JetBrains.

Побудовано діаграму варіантів використання для двох ролей користувачів, яка повністю описує можливий функціонал системи для ролей студента і вчителя.

- Розробити сторінку входу
- Розробити сторінку реєстрації
- Розробити сторінку виставлення оцінок
- Розробити сторінку статистики успішності
- Розробити сторінку розкладу
- Розробити сторінку завдань
- Розробити сторінку профілю

Завдяки чіткому виконанню завдань, поставлених на початку роботи, в результаті було отримано повноцінний UI для інформаційної системи електронного журналу, який готовий до використання в реальних умовах у зв'язці з сервером.

ДЯКУЮ ЗА УВАГУ!