КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теорії та технології програмування

Кваліфікаційна робота
на здобуття ступеня бакалавра

за спеціальністю 122 комп'ютерні науки: Інформатика
на тему:

РОЗРОБКА ЕЛЕКТРОННОГО ПІДРУЧНИКА З КУРСУ "ПРИКЛАДНА ЛОГІКА"

Виконав студент 4-го курсу
Нікіта ОРЛЯК

(підпис)

Науковий керівник
доктор фізико-математичних наук, професор
Степан ШКІЛЬНЯК

(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань
Студент

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теорії та технології програмування
«___» червня 2023 р.
протокол № ___
Завідувач кафедри
Микола НІКІТЧЕНКО

(підпис)

Київ – 2023

Реферат

Обсяг роботи – 55 сторінок, 15 ілюстрацій, 11 джерел посилань.

Тема: Розробка електронного підручника з курсу “Прикладна логіка”

Перелік ключових слів: Електронний підручник, JavaScript, HTML, CSS, React, DOM, WEB

Об’єктом роботи є процес інформатизації освіти з використанням сучасних інформаційних технологій. Предметом роботи є браузерний застосунок, який є електронним підручником за конкретною темою.

Метою роботи є розробка електронного підручника з курсу «Прикладна логіка», який дозволяв би користувачам оперативно отримати потрібну інформацію в галузі прикладної логіки, отримати поглиблені знання щодо програмно-орієнтованих логік часткових квазіарних предикатів та про секвенційні числення цих логік, про некласичні логіки, зокрема, модальні й темпоральні, та про їх застосування в інформатиці.

Для досягнення мети поставлено такі завдання:

- Аналіз існуючих електронних підручників.
- Аналіз підходів до розробки браузерних застосунків.
- Аналіз наявних технологій.
- Вибір технологій для реалізації.
- Вибір особливостей підручника.
- Розробка.
- Тестування.

Результати роботи: створено електронний підручник з курсу «Прикладна логіка» із розвинутими методами надання інформації користувачам; проаналізовано та опановано засоби для розробки браузерних застосунків, вироблено практичні навички щодо використання бібліотек та фреймворків; отримано знання щодо програмно-орієнтованих логік квазіарних предикатів та їх секвенційних числень, про модальні й темпоральні логіки та про їх застосування в інформатиці й програмуванні.

Зміст

Скорочення та умовні позначення

JS – JavaScript – це мова програмування, яка використовується для створення веб-додатків та динамічного змісту на веб-сторінках (див. [1]).

React – це бібліотека JavaScript, яка використовується для розробки інтерфейсів користувача веб-додатків (див. [2]). React дозволяє швидко створювати перевикористовувані компоненти та ефективно оновлювати вміст на сторінці без повного перезавантаження.

HTML – це мова розмітки гіпертексту, що використовується для створення веб-сторінок. HTML дозволяє визначати структуру та зміст веб-сторінки.

CSS – це мова таблиць стилів, яка використовується для задання зовнішнього вигляду веб-сторінок. CSS дозволяє задавати кольори, шрифти, розташування елементів та інші параметри.

DOM – це модель об'єктів документа, яка представляє структуру HTML документа та дозволяє змінювати його вміст та вигляд з використанням JavaScript.

WEB – World Wide Web – світова павутина, тобто система взаємопов'язаних веб-сторінок та ресурсів, доступних через Інтернет.

Redux – це відкрите програмне забезпечення для управління станом додатків у JavaScript, яке забезпечує передбачуваність та централізоване керування станом додатку (див. [3]). Він дозволяє зберігати стан додатку в одному місці (Store) та використовувати неінтерактивні функції (Reducers) для зміни стану. Redux найчастіше використовується разом з React, але може бути використаний з будь-яким іншим JavaScript-фреймворком або бібліотекою.

Вступ

Оцінка сучасного стану об'єкта дослідження або розробки.

Наш час характеризується розширенням сфери застосування дистанційної освіти, що передусім зумовлене пандемією коронавірусу та воєнним станом в Україні. Електронні навчальні матеріали стають все більш популярними та потрібними для навчання та самостійного вивчення різних дисциплін. Зокрема, підручники з прикладної логіки можуть бути корисними для студентів, які вивчають штучний інтелект, математичну логіку, теорію алгоритмів та обчислюваності, інформаційні системи та інші споріднені дисципліни.

Водночас існує багато електронних підручників з різних дисциплін, тому для розробки важливо дотримуватись сучасних стандартів та враховувати потреби та очікування користувачів. Зокрема, ефективність підручника буде залежати від його доступності, зручності використання, зрозумілості та охоплення необхідного матеріалу.

Актуальність роботи та підстави для її виконання.

Актуальність розробки електронного підручника з курсу "Прикладна логіка" полягає в тому, що з кожним роком все більше студентів вищих навчальних закладів вивчають дисципліни в галузі комп’ютерних наук та інформаційних технологій. При цьому, часто викладачі мають складнощі з підбором підручників, які б були цікавими, доступними та зрозумілими для студентів.

Електронний підручник є відмінним доповненням (а іноді й альтернативою) до традиційних підручників, оскільки він дає можливість студентам мати доступ до матеріалу у будь-який зручний для них час та місце. Крім того, електронний підручник може бути інтерактивним та

містити низку відео-, аудіо-, та інших матеріалів, використання яких дає змогу поліпшити якість та ефективність навчання.

Актуальність розробки електронного підручника з курсу "Прикладна логіка" також зумовлена розширенням сфери застосування популярних технологій веб-розробки, таких як HTML, CSS та JavaScript, які можуть бути використані для створення цього продукту. Знання цих технологій стає все більш важливим у сучасному інформатизованому світі, тому розробка електронного підручника з курсу "Прикладна логіка" є дуже актуальною та корисною для студентів, які бажають здобути та поглибити знання в цій галузі.

Таким чином, підставою для виконання даної дипломної роботи є потреба у створенні доступного, зрозумілого та ефективного електронного підручника з курсу "Прикладна логіка", що буде корисним для студентів та викладачів цієї дисципліни.

Мета й завдання роботи.

Метою пропонованої кваліфікаційної роботи бакалавра є розробка електронного підручника з курсу "Прикладна логіка", який дозволяв би користувачам оперативно отримати потрібну інформацію з теми «Прикладна логіка», отримати поглиблені знання щодо програмно-орієнтованих логік часткових квазіарних предикатів та про секвенційні числення цих логік, про некласичні логіки, зокрема, модальні й темпоральні, та про їх застосування в інформатиці. Використання підручника має на меті поліпшити процес навчання та зробити його більш доступним та зрозумілим для студентів.

Завдання кваліфікаційної роботи бакалавра наступні:

- Провести аналіз сучасних підручників та онлайн-курсів з математичних дисциплін та інформатики, щоб з'ясувати найефективніші методи навчання та найбільш вдалий формат подання матеріалу.
- Розробити дизайн та структуру електронного підручника, щоб забезпечити його зручне та зрозуміле використання.
- Розробити програмний код для створення електронного підручника, використовуючи такі технології, як HTML, CSS, JavaScript, React, та DOM.
- Заповнити електронний підручник необхідним матеріалом, який буде включати в себе теоретичні відомості та практичні завдання для закріплення знань.

У результаті виконання кваліфікаційної роботи буде створений ефективний та зручний електронний підручник з курсу "Прикладна логіка", який зможе бути успішно використаний для навчання студентів та сприяти поліпшенню якості навчального процесу.

Об’єкт, методи й засоби дослідження та розроблення.

Об’єктом дослідження та розробки є процес інформатизації освіти з використанням сучасних інформаційних технологій. Предметом роботи є браузерний застосунок, який є електронним підручником з курсу “Прикладна логіка”.

Для розробки електронного підручника будуть використані наступні методи і засоби:

- Метод аналізу та узагальнення наукової та методичної літератури з питань структури та змісту курсу “Прикладна логіка”.

- Метод проектування та розробки програмного забезпечення, включаючи веб-розробку, розробку баз даних та розробку інтерфейсу користувача.
- Засоби розробки програмного забезпечення, включаючи мови програмування JavaScript, HTML та CSS, фреймворк React та бібліотеку Material-UI.
- Метод тестування та валідації розробленого електронного підручника за допомогою відповідних інструментів, таких як Selenium WebDriver, Jest та React Testing Library.

Можливі сфери застосування.

Розробка електронного підручника з курсу "Прикладна логіка" може мати широкі сфери застосування в освітній галузі. Зокрема, підручник може бути використаний в навчальних закладах для покращення якості навчання, а також у процесі дистанційної освіти, що є особливо актуальним у зв'язку зі зростанням популярності онлайн-навчання. Цей підручник дає змогу користувачам оперативно отримати потрібну інформацію з теми «Прикладна логіка», набути поглиблені знання щодо програмно-орієнтованих логік та про секвенційні числення цих логік, знання про некласичні логіки, зокрема, модальні й темпоральні, та про їх застосування в інформатиці. Використання підручника має на меті поліпшити процес навчання та зробити його більш доступним та зрозумілим для студентів. Крім того, пропонований електронний підручник може бути використаний як допоміжний матеріал у підготовці студентів до різноманітних олімпіад, конкурсів та інших змагань у галузі комп'ютерних наук та програмування.

РОЗДІЛ 1. ЕЛЕКТРОННІ ПІДРУЧНИКИ

1.1 Що таке електронний підручник та історія його виникнення

Електронний підручник є новітнім засобом навчання, який поєднує у собі переваги інформаційних технологій та інтерактивності. Електронний підручник – це форма навчального матеріалу, який можна використовувати на комп'ютері або на іншому електронному пристрої, зокрема, такому як планшет, смартфон або електронна книга. Електронні підручники зазвичай містять інтерактивні елементи, які дозволяють студентам більш ефективно вивчати матеріал – відеоуроки, аудіофайли, тести та інші завдання. Такі підручники на сьогоднішній день успішно використовуються як доповнення чи заміна традиційним друкованим підручникам.

Історія електронного підручника починається у 1980-х роках, коли почали з'являтися перші комп'ютерні програми з навчальним матеріалом. У 1990-х роках розпочалося активне використання CD-ROM дисків як носіїв навчальної інформації. Компанії, такі як Microsoft та Apple, випустили програми з електронними книгами, що дозволяли користувачам читати книги на комп'ютерах. Водночас із поширенням Інтернету та розвитком технологій, електронні підручники стали більш доступними та популярними. У 2000-х роках з появою Інтернету стали популярними електронні підручники в онлайн форматі, доступні через різноманітні платформи та сайти.

Сьогодні електронні підручники широко використовуються в освіті. Вони дозволяють студентам отримувати доступ до навчального матеріалу у будь-який час і з будь-якого місця, зменшують вартість та витрати на

паперові підручники, а також забезпечують більш ефективне вивчення матеріалу завдяки використанню різноманітних інтерактивних елементів.

Багато навчальних закладів використовують електронні підручники як доповнення до традиційного навчання. Вони можуть містити відео, аудіо, інтерактивні завдання та тести, що дозволяє студентам вивчати матеріал більш ефективно та цікаво. Крім того, використання електронних підручників дозволяє істотно зменшити витрати на друк та зберігання підручників.

1.2 Наскільки змінилися електронні підручники за останні роки

За останні роки електронні підручники зазнали значних змін і покращень. Однією з головних тенденцій є перехід до більш інтерактивних та зручних форматів, які забезпечують більш глибоке розуміння матеріалу та підвищують ефективність навчання.

Наприклад, на сьогоднішній день електронні підручники можуть містити різноманітні мультимедійні елементи, такі як відео, анімація, зображення та інтерактивні діаграми. Це дозволяє студентам краще усвідомлювати й засвоювати матеріал та підвищувати інтерес до навчання.

Також відбувається постійний розвиток мобільних технологій, що дозволяє зручно користуватись електронними підручниками на різних пристроях. Багато видавництв тепер пропонують спеціальні додатки для смартфонів та планшетів, що забезпечують швидкий та зручний доступ до матеріалів.

Крім того, з'явилися нові можливості для авторів електронних підручників, що дозволяє забезпечувати більш ефективне навчання. Наприклад, автори можуть використовувати інтерактивні тести та

опитування для перевірки розуміння матеріалу та надання зворотного зв'язку. Також можливості віртуальної реальності дозволяють студентам відчути навчальний матеріал більш живою та реалістичною формою.

Водночас не зникає проблема доступності електронних підручників, особливо для тих, хто має обмежений доступ до Інтернету чи комп'ютерів. Також існують питання з приводу безпеки та конфіденційності даних студентів при використанні електронних підручників.

За останній час електронні підручники суттєво змінилися та стали більш доступними та зручними для використання. Розвиток технологій та зростання вимог до якості навчання ставлять нові виклики перед авторами підручників, що змушує їх постійно покращувати свої продукти.

Підсумовуючи, можна сказати, що в цілому електронні підручники стали невід'ємною частиною сучасного навчального процесу та продовжують розвиватися відповідно до потреб користувачів. Зміни в електронних підручниках за останні роки дозволяють навчанню бути більш доступним та ефективним, водночас при цьому потрібно розв'язувати проблеми доступності та безпеки для того, щоб дійсно досягти своєї мети – поліпшити процес навчання.

РОЗДІЛ 2. АРХІТЕКТУРА

2.1 Структура та архітектура електронних підручників

Структура та архітектура електронних підручників визначаються їх призначенням, функціональними вимогами та технічними можливостями. Зазвичай, електронні підручники мають подібну до паперових вербальну та графічну структуру. Вони складаються з титульного аркуша, змісту, розділів, підрозділів, тексту, ілюстрацій, таблиць та інших елементів.

Особливістю електронних підручників є їхній мультимедійний характер, що передбачає використання різних форматів візуалізації та відео- та аудіоелементів. Це дозволяє покращити процес засвоєння матеріалу, збільшити зацікавленість та мотивацію учнів, а також допомагає забезпечити доступ до навчального матеріалу особам з різними особливостями сприйняття.

Архітектура електронного підручника передбачає використання різноманітних засобів для зберігання, структурування та навігації навчального матеріалу. Зазвичай, підручник складається зі статичних та динамічних компонентів. Статичні компоненти - це ті, що не змінюються при взаємодії з користувачем (текст, зображення, таблиці), а динамічні - це ті, що можуть змінюватись, залежно від дій користувача (тестові завдання, інтерактивні симуляції).

У загальному, структура та архітектура електронних підручників залежить від їхньої мети та функціональних вимог, і розроблення їх потребує уважного аналізу та планування.

2.2 Архітектура електронних підручників з точки зору програмування

Архітектура електронних підручників є важливою складовою їх розробки. З погляду програмування, електронний підручник складається з різноманітних програмних компонентів, що взаємодіють між собою. Зазвичай, структура електронного підручника складається з декількох розділів або глав, кожен з яких може містити підрозділи та підпідрозділи. Кожен розділ може бути побудований за допомогою HTML-файлу, який містить відповідний контент та може бути оформлений за допомогою CSS. Загалом, розробка електронного підручника вимагає використання декількох мов програмування та технологій.

Основним елементом архітектури електронного підручника є його структура. Структура може бути представлена у вигляді дерева, де кожен вузол відповідає розділу підручника. Кожен розділ може містити підрозділи, які в свою чергу містять навчальний матеріал. Така структура дозволяє користувачеві зручно орієнтуватися в підручнику та швидко знаходити необхідну інформацію.

Отже, ще одним не менш важливим елементом архітектури електронного підручника є зручне та логічне навігаційне меню, яке дозволяє користувачам легко переходити між розділами та підрозділами. Тому, добре організований пошуковий інструмент значно полегшить користувачу пошук потрібної інформації.

Однією з важливих складових архітектури є система управління контентом. Система управління контентом (Content Management System – CMS) забезпечує можливість редагування та оновлення вмісту підручника. CMS також відповідає за збереження та організацію матеріалів, керування доступом користувачів до різних розділів та інші функції.

Для розробки електронних підручників також використовуються різні технології та мови програмування, наприклад, HTML, CSS, JavaScript, PHP, Python тощо. Для забезпечення більшої взаємодії з користувачем, можуть використовуватися різноманітні інтерактивні елементи. Для реалізації цих функцій часто використовуються мови програмування, такі як JavaScript та Python.

Отже, архітектура електронних підручників з погляду програмування дуже важлива, оскільки вона визначає зручність та ефективність користування підручником. Добре структурований та логічно побудований підручник забезпечує більш якісне та ефективне навчання.

2.3 Опис архітектури електронного підручника з теми «Прикладна логіка»

Задля створення електронного підручника з теми «Прикладна логіка» я розробив наступну архітектуру:

- **Header –** шапка електронного підручника є головною та незмінною на усьому сайті (в підручнику). Вона демонструє основні розділи підручника, що дозволяє швидко та легко знайти потрібну тему та перейти до неї.
- **Side Panel –** бокова панель не менш важлива, бо саме вона показує користувачу основні підтеми обраної теми. Частіш за все на цій панелі ми побачимо основну інформацію про обрану тему, її пояснення, значення, приклади, тощо.
- **Additional information –** якщо користувач захоче отримати більше інформації з потребуючої його теми, то натиснувши на кнопку «Джерела» він отримає повний перелік джерел з додатковою інформацією.

- **Main menu –** головне меню електронного підручника з описом самого підручника, коротким описом його складової а також інформацією як користуватися підручником.
- **Information about author –** інформація про автора підручника, а саме: загальна інформація, його контактні дані та попередні роботи. Щоб у разі помилок або проблем – користувач зміг зв’язатися та повідомити або допомогти покращенню підручника.

Задля цього я використовував наступні технології:

- **HTML –** зручна мова розмітки, яка допомогла мені побудувати структуру та дизайн підручника, а також наповнити його потрібним змістом.
- **CSS –** мова таблиць стилів, яка надала мені змогу покращити візуальну складову підручника задля більш приємного користування іншими людьми.
- **JavaScript –** мова програмування, завдяки якій я зміг «оживити» підручник, зробивши його більш зручним та легшим у використанні для користувача.
- **React –** зручний та популярний фреймворк, який дозволив мені «зв’язати» сторінки підручника, полегшити навантаження та прискорити завантаження сторінок, що зумовлює більш швидке та приємне користування.
- **Redux –** бібліотека для керування станом додатків у JavaScript, яка покращила чистоту та простоту коду, а також дозволила легко проводити тестування.

РОЗДІЛ 3. СТРУКТУРА ЕЛЕКТРОННОГО ПІДРУЧНИКА З КУРСУ "ПРИКЛАДНА ЛОГІКА"

3.1 Прикладна логіка

Основною темою пропонованого електронного підручника стала саме прикладна логіка.

Прикладна логіка (Applied logic) – це розділ математичної логіки, який займається розробкою та використанням формальних логіко-математичних методів для моделювання та вирішення різноманітних задач. Методи й засоби прикладної логіки успішно використовуються в різних галузях штучного інтелекту, інформатики й програмування. Апарат прикладної логіки застосовується для проектування комп'ютерних та інформаційних систем, розробки програмного забезпечення, специфікації та верифікації програм, побудови систем пошуку виведень, створення баз даних та баз знань, низки інших задач.

Прикладна логіка є важливою складовою багатьох галузей, таких як інформатика, математика, філософія, філологія, фізика, біологія та інші. Вона допомагає вирішувати різні задачі, які потребують формалізації, наприклад, перевірку правильності аргументації в лінгвістичних теоріях, розробку формальних засобів при створенні систем штучного інтелекту, в задачах аналізу даних тощо. В цілому прикладна логіка дає змогу використовувати формальні логіко-математичні методи для вирішення важливих проблем у науці та технологіях (див. [5]).

Як приклади використання прикладної логіки можна вказати розробку мов програмування Prolog та Lisp, які базуються на апараті математичної логіки; розробку низки алгоритмів для штучного інтелекту та обробки природних мов. Апарат прикладної логіки успішно використовується для створення систем пошуку доведень теорем, що дозволяє отримувати

теоретичні твердження за допомогою формальних методів. Інший приклад - розробка формальних граматик для створення комп'ютерних програм, які здатні аналізувати та розуміти природну мову. Також прикладна логіка може бути використана для розробки баз даних та баз знань, які забезпечують точність та надійність при збереженні та обробці даних.

Таким чином, прикладна логіка є дуже важливою з практичного погляду галуззю знань. Розвиток штучного інтелекту, інформаційних технологій та програмування зумовлює невпинне розширення сфери її застосування.

3.2 Програмно-орієнтовані моделі логічних систем

В першому розділі посібника розглядаються питання побудови адекватних логічних формалізмів, орієнтованих на різноманітні задачі програмування й моделювання. Вказано особливе місце класичної логіки предикатів серед логічних формалізмів, описано її позитивні вартості. Водночас наведено принципові обмеження класичної логіки, які ускладнюють її використання в програмуванні та моделюванні (див. [4, 5]). Такі обмеження мотивують необхідність побудови нових логік, які більше орієнтовані на потреби інформатики й програмування. Описано основні аспекти логік, орієнтованих на дослідження програм. Зроблено висновок, що за основу побудови нових, програмно-орієнтованих логік природно взяти спільний для логіки й програмування композиційно-номінативний підхід. Описано основні принципи композиційно-номінативного підходу. Логіки, збудовані основі цього підходу, названо композиційно-номінативними (КНЛ).

Застосування композиційно-номінативного підходу дозволило побудувати низку логічних моделей різноманітних предметних областей, що знаходяться на різних рівнях абстрактності та загальності [4, 5].

Виділено (див. [4, 5, 9]) рівні розгляду КНЛ, такі рівні відрізняються трактуванням рівня розгляду даних:

1. Пропозиційний рівень.

2. Сингулярний рівень.

3. Номінативно-безквантoрні рівні:

– реномінативний (рівень РНЛ);

– реномінативний з слабкою рівністю (рівень $РНЛ_{=}$);

– реномінативний з строгою рівністю (рівень $РНЛ_{\equiv}$);

– безквантoрно-функціональний (рівень БФЛ);

– безквантoрно-функціональний з слабкою рівністю (рівень $БФЛ_{=}$);

– безквантoрно-функціональний з строгою рівністю (рівень $БФЛ_{\equiv}$).

4. Першопорядкові рівні:

– квантoрний, або чистий першопорядковий рівень (рівень ЧКНЛ);

– чистий першопорядковий рівень з слабкою рівністю (рівень ЧКНЛ);

– чистий першопорядковий рівень з строгою рівністю (рівень $ЧКНЛ_{\equiv}$);

– функціональний рівень (рівень ФКНЛ);

– функціональний рівень з слабкою рівністю (рівень $ФКНЛ_{=}$);

– функціональний рівень з строгою рівністю. (рівень $ФКНЛ_{\equiv}$).

5. Ієрархічно-номінативний рівень. Варто зауважити, що цей рівень дуже багатий, детальне його дослідження лише розпочалось.

Наведено спектр композиційно-номінативних логік за рівнем абстракції розгляду. Описано різновиди КНЛ за обмеженнями на клас квазіарних предикатів.

Далі в підручнику основна увага приділена вивченню чистих першопорядкових логік (ЧКНЛ) без обмежень монотонності предикатів.

3.3 Першопорядкові логіки квазіарних предикатів

В другому розділі “Предикатні композиційні системи” описано композиційні системи та алгебри квазіарних предикатів. Квазіарні функції та квазіарні предикати – це часткові відображення, задані на номінативних (іменних) даних. Базовим класом номінативних даних є іменні множини (ІМ). Вони генерують найважливіші типи даних для програмування та програмно-орієнтованих логік. Дано визначення ІМ, описано операції над ними. Особливу увагу приділено операціям реномінації та розширеної реномінації.

Дано визначення квазіарної функції та квазіарного предиката. Описано різновиди часткових неоднозначних квазіарних предикатів реляційного типу, або *R*-предикатів.

Визначено композиції квазіарних предикатів пропозиційного рівня (логічні зв’язки), реномінативного рівня (реномінації та розширені реномінації), першопрядкового рівня (квантори). Описано властивості цих композицій. Визначено композиційні системи та алгебри квазіарних предикатів на реномінативному та чистому першопорядковому рівнях.

Розглянуто особливості логік квазіарних предикатів. Показано, що для таких логік перестають бути вірними деякі важливі закони класичної логіки. Зокрема, в логіці квазіарних предикатів без обмежень монотонності спростовуються предикати вигляду $Q \to \exists xQ$ та $\forall xQ \to Q$.

В третьому розділі “Першопорядкові логіки квазіарних предикатів” вивчаються семантичні властивості чистих першопорядкових КНЛ квазіарних предикатів – ЧКНЛ (див. [5, 6, 9, 10]). Описано мови L^Q – чистих першопорядкових логік з традиційними реномінаціями, та мови $L_\perp{}^Q$ – чистих першопорядкових логік з композицією розширеної реномінації та предикатами-індикаторами. Виділено класи інтерпретацій

таких мов, або семантики. Визначено важливі поняття неспростовної (істинної) та виконуваної формули.

На основі різних співвідношень між областями істинності та хибності предикатів введено низку відношень логічного наслідку на множині формул мови – відношення $^{P}|=_{IR}$, $^{P}|=_{T}$, $^{P}|=_{F}$, $^{P}|=_{TF}$, $^{R}|=_{TF}$. Досліджено властивості цих відношень, наведено співвідношення між ними. Це зроблено для мови L^{Q} та для мови $L_{\perp}{}^{Q}$.

Відношення логічного наслідку поширюються на пари множин формул. Описано властивості цих відношень на пропозиційному та на номінативних рівнях. Можна виділити властивості:

– декомпозиції формул

– спрощення та еквівалентних перетворень

– елімінації кванторів

– гарантованої наявності відповідного відношення логічного наслідку.

Властивості відношень логічного наслідку для множин формул є семантичною основою побудови відповідних числень Ґенценівського, або секвенційного типу [4, 6].

3.4 Секвенційні числення

Секвенційні числення формалізують відношення логічного наслідку для множин формул (див. [4, 6, 8–10]). На даний момент побудовано низку секвенційних числень для різних класів КНЛ часткових квазіарних предикатів. В підручнику детально описано числення першопорядкових логік квазіарних предикатів із розширеними реномінаціями (див. [8]), які формалізують відношення логічного наслідку $^{P}|=_{IR}$, $^{P}|=_{T}$, $^{P}|=_{F}$, $^{P}|=_{TF}$, $^{R}|=_{TF}$.

Такі секвенційні числення будуються в стилі семантичних таблиць, секвенції трактуються як множини формул, специфікованих символами $\vdash$ та $\dashv$. Секвенції позначаємо у вигляді $\vdash\Gamma\dashv\Delta$, специфікація $\vdash$ відповідає

семантичному трактуванню формул із Γ як істинних, а специфікація ⊣ відповідає семантичному трактуванню формул із Δ як хибних.

Секвенційні числення будуються за таким принципом:

секвенція $\vdash \Gamma \dashv \Delta$ має виведення $\Leftrightarrow \Gamma \models \Delta$.

Виведення в секвенційних численнях має вигляд дерева. Вершинами цього дерева є секвенції, тому такі дерева названо секвенційними.

Правилами виведення секвенційних числень є секвенційні форми – це синтаксичні аналоги семантичних властивостей відношень |=.

Аксіомами секвенційного числення є замкнені секвенції. Це поняття уточнюють по-різному в секвенційних числень для різних відношень |= . При цьому має виконуватись умова: секвенція $\vdash \Gamma \dashv \Delta$ замкнена $\Rightarrow \Gamma \models \Delta$.

Для формалізації відношень логічного наслідку в $L_{\perp}{}^{Q}$ запропоновано такі секвенційні числення:

– відношення $^{P}\models_{IR}$ формалізує числення $C_{\perp}{}^{QIR}$;

– відношення $^{P}\models_{T}$ формалізує числення $C_{\perp}{}^{QT}$;

– відношення $^{P}\models_{F}$ формалізує числення $C_{\perp}{}^{QF}$;

– відношення $^{P}\models_{TF}$ формалізує числення $C_{\perp}{}^{QTF}$;

– відношення $^{R}\models_{TF}$ формалізує числення $C_{\perp}{}^{QTFR}$.

Наведено базові секвенційні форми зазначених числень та умови замкненості секвенції.

Описано побудову секвенційного дерева (виведення) для заданої секвенції.

Для збудованих секвенційних числень доведена теорема коректності, вона формулюється однотипно для розглянутих числень:

якщо секвенція $\vdash \Gamma \dashv \Delta$ вивідна, то $\Gamma \models \Delta$.

Доведення теорем про повноту секвенційних числень базується на теоремах про побудову контрмоделі за незамкненим шляхом в секвенцій-

ному дереві. Для доведення теорем про контрмоделі використовують метод модельних (Хінтікківських) множин

Теорема повноти формулюється однотипно для розглянутих числень:

якщо секвенція $\vdash\Gamma\dashv\Delta$ вивідна, то $\Gamma \models \Delta$.

Далі в розділі наведено низку прикладів побудови виведень в секвенційних численнях $C_{\perp}^{QIR}$ (12 прикладів) та в численнях $C_{\perp}^{QT}$, $C_{\perp}^{QF}$, $C_{\perp}^{QTF}$, $C_{\perp}^{QTFR}$ (11 прикладів).

3.5 Логіки предикатів над номінативними даними

В п’ятому розділі вивчаються класи КНЛ часткових предикатів над номінативними даними. Дані номінативного рівня будуємо індуктивно із множини імен та множини базових значень.

Розглядаються спеціальні логіки, над скінченними номінативними даними (див. [5]), що виникають як формалізми специфікацій програм. Такі скінченні багатозначні номінативні дані названі метаномінативними. Аксіоматична система специфікацій програм над метаномінативними даними дає змогу дослідити властивості часткових багатозначних функцій та провести конкретизацію специфікацій у мовах програмування нижчого рівня.

Досліджено абстрактну обчислюваність над номінативними даними (див. [5]). Описано логіку часткових неоднозначних предикатів над скінченними номінативними даними, її названо логікою номінативних даних. Ця логіка є конкретизацією традиційної логіки квазіарних предикатів 1-го порядку. Для логіки номінативних даних побудовано формально-аксіоматичну систему Гільбертівського типу – теорію номінативних даних.

Основну увагу в розділі приділено вивченню логік над ієрархічними номінативними даними зі складеними іменами (див. [11]). Дано визначення таких даних, описано операції над ними. Введено нормальні форми

операції видалення компонент із заданими префікс-іменами та операції реномінації.

Визначено предикати над даними зі складеними іменами – *CND*-предикати, введено композиції *CND*-предикатів. Описано особливості композицій реномінації та квантифікації. Визначено композиційні системи та алгебри першопорядкових логік *CND*-предикатів.

Описано мови чистих першопорядкових логік *CND*-предикатів. На множинах формул такої мови визначено низку відношень логічного наслідку, наведено їх властивості. Відношення логічного наслідку поширено на множини формул, описано властивості цих відношень.

На основі властивостей відношень логічного наслідку для множин формул побудовано низку секвенційних числень логіки *CND*-предикатів. Для таких числень доведено теореми коректності та повноти.

3.6 Нетрадиційні логіки

Традиційна логіка предикатів має дві визначальні особливості:

– множина $\{T, F\}$ її істиннісних значень є двоелементною;

– вона базується на фундаментальному заміни еквівалентних (рівних); це означає можливість заміни еквівалентних (рівних) незважаючи на контексти.

Втіленням принципу заміни еквівалентних (рівних) в логіці є теореми еквівалентності та рівності.

Класична логіка розглядає ситуації як незмінні в часі та просторі, вона не працює, коли нас цікавить розвиток понять, а не істинність в конкретній ситуації. Вона малопридатна для формалізації незнання. Все це зумовлює необхідність вивчення нетрадиційних, некласичних логік.

В шостому розділі розглядаються такі відомі різновиди нетрадиційних логік, як багатозначні та інтуїціоністські (див. [5, 9]).

Першою системою багатозначних логік є тризначні логіки Я. Лукасєвича. При цьому найважливішим є доведення Я. Лукасєвичем самої *можливості* існування некласичних логік, що за значимістю стоїть поряд із відкриттям неевклідових геометрій. Далі Я. Лукасєвич запропонував чотиризначні, багатозначні та нескінченнозначні логіки. Концепція нескінченнозначної логіки лягла в основу побудови ймовірнісних та нечітких логік.

Серед тризначних логік найбільш відомими є сильна та слабка логіки С. Кліні. Вони використовуються, зокрема, в теорії рекурсії, в системах алгоритмічних алгебр, в мовах табличних баз даних.

Із чотиризначних логік найбільш відомою є логіка Белнапа. Ця логіка знайшла застосування для опису інформаційних систем із неповною та суперечливою інформацією.

В розділі наведено логічні зв’язки тризначної логіки Лукасєвича, сильної та слабкої тризначних логік Кліні, чотиризначної логіки Белнапа.

Далі в розділі досліджено зв'язки між композиційно-номінативними логіками часткових однозначних та часткових неоднозначних квазіарних предикатів і тризначними й чотиризначними логіками традиційних однозначних предикатів. Встановлено, що логіці часткових однозначних квазіарних предикатів відповідає сильна тризначна логіка Кліні, а логіці часткових неоднозначних квазіарних предикатів відповідає чотиризначна логіка Белнапа. Доведено ізоморфізми відповідних предикатних алгебр. Це демонструє особливе місце сильної логіки Кліні серед тризначних та логіки Белнапа серед чотиризначних.

Далі в розділі розглянута інтуїціоністська логіка. Така логіка описує математичні твердження не як абстрактні істину чи фальш, а як твердження про можливість виконання деякої побудови, тому кожне математичне доведення має давати таку побудову та її обґрунтування. Для інтуїціоністської логіки вже не діє закон виключеного третього й пов'язані з ним закони де

Моргана, не діє і закон зняття подвійного заперечення. Пропозиційні зв'язки $\neg$, $\vee$, $\&$, $\rightarrow$ тут незалежні, квантори $\exists x$ та $\forall x$ теж незалежні.

В розділі описано мови інтуїціоністської пропозиційної логіки та першопорядкової інтуїціоністської логіки предикатів, розглянуто реляційну семантику інтуїціоністської логіки.

Наведено формально-аксіоматичні системи гільбертівського типу для інтуїціоністської логіки – інтуїціоністські числення пропозиційного та першопорядкового рівнів. Також розглянуто оригінальні секвенційні числення інтуїціоністської логіки.

3.7 Модальні логіки

Сьомий розділ електронного підручника присвячений розгляду модальних логік. Такі логіки доцільно використовувати для опису мінливого світу, який змінюється та розвивається.

Під модальностями розуміють властивості тверджень, які в тому чи іншому аспекті характеризують міру їх істинності чи наше ставлення до них (див. [5, 9]). Модальності "необхідно" й "можливо" називають загальними, або алетичними. Модальності, які мають часовий зміст, називають темпоральними (часовими). Основними темпоральними модальностями є "завжди було", "колись було", "завжди буде", "колись буде". Алетичні та темпоральні модальності відомі з античних часів. Міркування з твердженнями, що містять алетичні модальності, вивчає алетична модальна логіка, а міркування з твердженнями, що містять темпоральні модальності, вивчає темпоральна, логіка. Можна також виділити модальності, що характеризують міру обґрунтованості знання, їх називають епістемічними. Це модальності "доведено", "підтверджено", "обґрунтовано", "спростовано" тощо. Модальна логіка, яка вивчає міркування з твердженнями, що містять епістемічні модальності, називається

епістемічною. Розглядають також інші різновиди модальних логік, зокрема, деонтична логіка вивчає міркування з модальностями, які характеризують норми й нормативні поняття: "обов'язково", "дозволено", "заборонено".

Модальні логіки ефективно використовуються для моделювання різноманітних предметних областей і аспектів діяльності людини. Потужного імпульсу розвитку модальних логік надало створення сучасних інформаційних та програмних систем. Із цього погляду найважливішими є темпоральні та епістемічні логіки. Темпоральні логіки використовуються для опису й моделювання складних динамічних систем, для специфікації та верифікації програм. Епістемічні логіки знайшли використання для опису баз даних та баз знань, інтелектуальних інформаційних систем та систем штучного інтелекту.

В розділі описано такі різновиди традиційних модальних логік: алетичні, темпоральні, епістемічні, деонтичні. Наведено аксіоматичні системи таких логік на пропозиційному рівні, описано реляційну семантику (семантику можливих світів) для цих логік. Особливу увагу приділено темпоральним логікам. Розглянуто класифікацію систем темпоральної логіки, описано застосування темпоральних логік для специфікації та верифікації програм.

Можливості логік часткових квазіарних предикатів (див. розділи 2–4) і традиційних модальних логік поєднують композиційно-номінативні модальні логіки часткових предикатів (див. [5, 7, 9]). Найважливішим їх класом є транзиційні модальні логіки (див. [7]), які адекватно відбивають аспект зміни й розвитку предметних областей, описуючи переходи від одного стану світу до іншого. Традиційні модальні логіки (алетичні, темпоральні, епістемічні тощо) можна природним чином розглядати в межах транзиційних модальних логік (ТМЛ).

В підручнику основну увагу приділено чистим першопорядковим ТМЛ. Семантичною основою ТМЛ є транзиційні модальні системи (ТМС), в таких системах відношення на станах світу трактуються як відношення переходу на станах. Виділено такі різновиди ТМС: загальні транзиційні (ЗМС), темпоральні (ТмМС), мультимодальні (ММС) системи. Описано мови першопорядкових ЗМС, ТмМС, ММС. Залежно від умов, накладених на відношення переходу, можна визначити різні класи таких ТМС. Розглянуто випадки, коли це відношення може бути рефлексивним, симетричним чи транзитивним.

Описано основні властивості ТМС. Розглянуто взаємодію модальних композицій ТМС із реносмінаціями та кванторами.

Відношення логічного наслідку в ТМЛ задаються (див. [7]) на множині формул, специфікованих іменами станів (множині специфікованих станами формул). Немодальні властивості таких відношень фактично успадковані від традиційної логіки квазіарних предикатів. Характерними для ТМЛ особливостями є властивості цих відношень, пов'язані з модальними композиціями, зокрема, властивості елімінації модальностей.

Властивості відношень логічного наслідку для множин специфікованих станами формул є семантичною основою для побудови секвенційних числень різних класів ТМЛ. Наведено базові секвенційні форми та умови замкненості секвенції таких числень. Для цих числень доведено теореми коректності та повноти.

РОЗДІЛ 4. ПРОБЛЕМИ ТА ЇХ ВИРІШЕННЯ

4.1 Наявні проблеми

За останні місяці електронний підручник був дороблений та доведений до готового продукту. На проміжному етапі розробки (першій презентації електронного підручника) продукт мав явні недоліки, а саме:

- **Дизайн:** підручник мав наявні проблеми з візуальною частиною. Перш за все – дуже яскраві та насичені кольори, які заважали простому користуванню. Яскраві анімації та переходи, які сильно кидалися у вічі користувачу. Ця проблема також могла призвести до дискомфорту у людей з певними ускладненнями зору. По-друге – гострі кути та краї зображення, які при довгому користуванні робили вид підручника «важким» для певних користувачів. По-третє – відсутність динаміки та простої анімації в багатьох частинах підручника, яка полегшила б його користування.
- **Матеріал:** продукт мав певні «пробіли» у інформації, яку надавав користувачу. В деяких місцях інформація була неповна, а в деяких взагалі відсутня, що змушувало користувача шукати додаткову інформацію на сторонніх джерелах.
- **Джерела:** відсутність наявних джерел з додатковою та повною інформацією, яка б дозволила користувачу поглибити свої знання з конкретної області.
- **Програмна частина:** підручник мав проблеми з завантаженням інформації та її обробкою. Сторінки довго завантажувалися, деяка інформація зникала при завантаженні.

4.2 Вирішення проблем

Задля вирішення даних проблем мною було вирішено поглибити знання стосовно основних використаних технологій, а саме:

- **React –** опановано використання Hooks, Link, NavLink, react-dom та Route. Впроваджено нові технології та покращено їх використання.
- **JavaScript –** поглиблено знання та навички з використанням даної мови програмування. Впроваджено нові методи взаємодії з користувачем.
- **Redux –** опанована бібліотека. Впроваджено технології та загальне використання бібліотекою.
- **HTML, CSS –** покращено знання з даних технологій, опановано нові нововведення, впроваджено їх у вже існуючу структуру продукту.

Завдяки новим технологіям та їх функціоналу вдалося вирішити наявні проблеми та усунути недоліки продукту. Проблеми були вирішені наступним чином:

- Покращення дизайну – було вирішено змінити кольорову гаму деяких частин підручника; це сприяло покращенню загального вигляду продукту та полегшило користування продуктом користувачами з певними ускладненнями зору.
- Структуровано, покращено та доповнено матеріал – було поглиблено знання за темою підручника та додано нову, краще структуровану інформацію.
- Додано нові джерела – наразі користувач може ознайомитися з додатковим матеріалом, посилання на які було додано до продукту.

- Було покращено програмну компоненту продукту – завдяки новим технологіям та їх функціоналу було покращено загальну програмну частину. За допомогою тестування були виявлені баги та недоліки, які також вдалось успішно виправити та покращити функціонал підручника.

РОЗДІЛ 5. ПРОЕКТ. ЙОГО СТРУКТУРА ТА ВИКОРИСТАНІ ТЕХНОЛОГІЇ

5.1 Мета та ідея проекту

Головна ідея проекту полягала в розробці інтерактивного електронного підручника, який буде містити основний матеріал з обраної теми, який міг би повністю задовольнити широке коло користувачів, які мали б потребу в освоєні теми «Прикладна логіка».

Важливою метою стало освоєння теоретичних та практичних навичок роботи з технологіями, які були використані в процесі розробки електронного підручника. Це мова програмування, розмітки, фреймворки, а також теоретичні знання з основних розділів підручника – програмно-орієнтовані логіки часткових квазіарних предикатів та секвенційні числення таких логік, нетрадиційні та модальні логіки.

5.2 Обрані технології, їх переваги та недоліки. Порівняння з іншими схожими технологіями

- **HTML** (Hypertext Markup Language) є основною мовою розмітки для створення веб-сторінок та веб-додатків. Введена в 1991 році, HTML постійно розвивалась та удосконалювалась, щоб забезпечити підтримку нових можливостей та функціональності веб-додатків.

Переваги HTML:

- Простота вивчення та розуміння.
- Відкритий стандарт з відкритим кодом.
- Можливість відображення тексту, зображень, відео та інших мультимедійних елементів.

- Широко підтримується веб-браузерами та різними платформами.

Недоліки HTML:

- Не має вбудованих можливостей для динамічного взаємодії з користувачем.
- Для створення динамічних веб-сторінок потрібно використовувати додаткові технології, такі як CSS та JavaScript.

Порівняння з іншими схожими технологіями:

- XHTML: XHTML є більш суворим варіантом HTML, що вимагає більш строго використання правильної розмітки. Однак, HTML має більш широку підтримку та менш складне використання.
- XML: XML – це загальна мова розмітки, що не обмежена при використанні веб-додатками. Вона є більш гнучкою та може бути використана для обміну даними між різними системами. Водночас вона не підтримує вбудовані можливості для відображення веб-сторінок, які має HTML.

Рис 1. Мови розмітки

У своїй дипломній роботі я використовував HTML, щоб створити основну структуру веб-сторінок та забезпечити їх основну функціональність. Ця технологія є найпоширенішою мовою розмітки веб-сторінок та є стандартом для розробки веб-сайтів. HTML дозволяє

відображати текст, зображення, відео та інші мультимедійні елементи на веб-сторінках, що забезпечує їх більшу інформативність та привабливість для користувачів.

- **CSS** (Cascading Style Sheets) є мовою опису стилів, яка використовується для зовнішнього оформлення веб-сторінок, включаючи відображення тексту, зображень, таблиць, форм та інших елементів. Вперше введена у 1996 році, CSS дозволяє веб-розробникам відокремлювати дизайн та стиль веб-сторінок від їх структури та контенту.

Переваги CSS:

- Відділення відмінностей у відображенні та змісті: CSS дозволяє відокремлювати візуальні елементи від HTML структури, що забезпечує кращу структуру коду та полегшує зміну візуальних елементів.
- Можливість створення стилів для різних пристроїв: CSS дозволяє розробникам створювати стилі, які можуть змінюватися залежно від пристрою, на якому відображається веб-сторінка, що забезпечує кращу оптимізацію та придатність до використання.
- Ефективність та повторне використання: CSS дозволяє створювати стилі для кількох сторінок веб-сайту, що зменшує обсяг коду та полегшує його підтримку та редагування.

Недоліки CSS:

- Складність вивчення: CSS вимагає певного рівня знань та вмінь від розробників, щоб створювати візуально привабливі та ефективні веб-сторінки.

- Сумісність з браузерами: Різні браузери можуть по-різному інтерпретувати CSS, що може призвести до проблем з відображенням веб-сторінок на різних платформах та браузерах.

Порівняння з іншими схожими технологіями:

- SASS є препроцесором CSS, що дозволяє використовувати змінні, функції та інші програмні конструкції, які допомагають зменшити кількість коду та зробити його більш структурованим. SASS також має підтримку для математичних операцій, умов та інших програмних конструкцій, які полегшують розробку складних стилів. Однак, використання SASS потребує додаткового кроку компіляції, щоб перетворити SASS-код в звичайний CSS.
- LESS є іншим препроцесором CSS, який також дозволяє використовувати змінні та програмні конструкції для зменшення обсягу коду. LESS також підтримує ієрархічну структуру, яка допомагає зберегти стилі більш організованими. LESS відрізняється від SASS тим, що він використовує звичайний синтаксис CSS, що полегшує вивчення та використання для тих, хто вже знайомий з CSS.
- Stylus є ще одним препроцесором CSS, який дозволяє використовувати змінні та програмні конструкції для зменшення кількості коду та зробити стилі більш структурованими. Відрізняється він від SASS та LESS тим, що має більш гнучкий синтаксис та дозволяє використовувати звичайний CSS разом зі своїми розширеннями. Також має вбудовану підтримку для математичних операцій, умов та інших програмних конструкцій.

Рис 2. Мови опису стилів

Було прийнято рішення обрати саме CSS, тому що він дозволяє розмістити елементи на сторінці саме так, як потрібно, додати анімацію та інші ефекти, змінювати стиль та вигляд сторінки в залежності від розміру екрану та пристрою користувача. Завдяки цьому я зміг створити зручні та естетично привабливі, на мій погляд, веб-сторінки, які добре пристосовуються до різних пристроїв та розмірів екрану.

- **JavaScript** є однією з найпоширеніших мов програмування, яка використовується для розробки веб-додатків та веб-сторінок. Запропонована в 1995 році, JavaScript (див. [1]) стала невід'ємною частиною веб-розробки, забезпечуючи динамічність та інтерактивність на стороні клієнта.

Переваги JavaScript:

- Динамічність та інтерактивність: JavaScript дозволяє створювати динамічні та інтерактивні елементи на сторінці, такі як анімація, валідація форм, взаємодія з користувачем та інше.
- Переносність: JavaScript може працювати на будь-якій платформі та в будь-якому веб-браузері, що робить його високо переносним.
- Швидкодія: JavaScript є високопродуктивною мовою програмування, що дозволяє створювати швидкі та ефективні веб-додатки.

Недоліки JavaScript:

- Безпека: JavaScript може бути вразливим до атак на стороні клієнта, таких як віруси та шкідливі скрипти.
- Залежність від веб-браузера: деякі функції JavaScript можуть не підтримуватися в старіших веб-браузерах, що може призвести до проблем з сумісністю.
- Складність: JavaScript є високорівневою мовою програмування, що може вимагати більшої кількості коду для досягнення певних функцій.

Порівняння з іншими схожими технологіями:

- Java: Java є повномасштабною мовою програмування, яка використовується для створення різноманітних додатків. Незважаючи на те, що JavaScript і Java мають спільне слово в назві, вони не мають багато спільного. JavaScript є мовою програмування, яка використовується виключно для розробки веб-додатків, тоді як Java може використовуватись для розробки різних типів додатків, таких як мобільні додатки та настільні додатки.
- Python: Python є іншою популярною мовою програмування, яка використовується для розробки веб-додатків та програмного

забезпечення загалом. Python зазвичай використовується для розробки більш складних додатків, ніж JavaScript, і має більш широкий діапазон застосування. Однак, JavaScript є найбільш популярною мовою програмування для розробки веб-додатків.

- Ruby: Ruby є ще однією мовою програмування, яка використовується для розробки веб-додатків. Вона має дуже схожий синтаксис з Python, але працює швидше за нього. Ruby має широкий діапазон застосування, але він менш популярний у порівнянні з JavaScript.
- TypeScript: TypeScript є суперсетом JavaScript, що дозволяє додати сильну типізацію до JavaScript-коду. TypeScript розроблений компанією Microsoft та має широку підтримку у веб-розробці. TypeScript дозволяє виправити багато проблем, пов'язаних з JavaScript, таких як невизначеність типів даних, помилки через неправильну передачу параметрів функцій, труднощі у відлагодженні складних програмних проектів тощо.

Рис 3. Мови програмування

Для даного продукту була обрана мова JavaScript тому, що ця технологія дозволяє розробникам створювати динамічні веб-сторінки з інтерактивними елементами. Я використовував JavaScript для створення скриптів, які забезпечують взаємодію користувача з моєю веб-сторінкою. JavaScript також дозволяє забезпечити валідацію форм, анімацію, додаткові можливості для роботи з DOM та інші функції, що покращують користувацький досвід. Крім того, JavaScript є однією з найпопулярніших технологій для розробки веб-додатків, тому я мав можливість знайти багато корисних ресурсів та підтримки спільноти для своєї роботи.

- **React** (див. [2]) – це відкрита JavaScript бібліотека для розробки інтерфейсів користувача. Вона створена Facebook в 2011 для внутрішнього використання. Вперше він був опублікований у 2013 році і з того часу здобув широку популярність серед розробників. У 2015 році Facebook оголосив про випуск React Native – фреймворку для розробки мобільних додатків на React. Навіть зараз він використовується для створення веб-додатків та мобільних додатків з високим рівнем взаємодії користувача.

Переваги React:

- React використовує Virtual DOM, що дозволяє зменшити кількість маніпуляцій з DOM-елементами і забезпечує більш швидку роботу додатків.
- React є декларативним, що дозволяє зменшити кількість коду, необхідного для створення складних інтерфейсів.
- React має велику спільноту розробників, що забезпечує швидку підтримку та поширення додатків.

- React дозволяє використовувати компоненти, що полегшує розробку та підтримку додатків.

Недоліки React:

- React не є повноцінним фреймворком, тому для повноцінної розробки додатків може бути необхідно використовувати додаткові бібліотеки.
- React не має вбудованої можливості роботи зі станом додатка, тому для цього необхідно використовувати додаткові засоби.
- Використання React може призвести до проблем з оптимізацією додатків через використання Virtual DOM.

Порівняння з іншими технологіями:

- Angular: Angular – це повноцінний фреймворк для розробки веб-додатків, який пропонує свій підхід до розробки інтерфейсу користувача. Angular використовує своє внутрішнє рішення для оновлення DOM, а також пропонує різні інструменти для роботи зі структурою додатку та управління станом.
- Vue.js: Vue.js – це бібліотека, яка надає можливості для розробки користувацьких інтерфейсів та легко інтегрується з іншими бібліотеками та фреймворками. Vue.js має просту API, що дозволяє швидко створювати компоненти та використовувати їх у проекті.
- Ember.js: Ember.js – це фреймворк для розробки веб-додатків, який має свій підхід до розробки інтерфейсу користувача. Ember.js пропонує своє рішення для роботи зі станом та використовує шаблонізатор Handlebars для генерації HTML-коду.

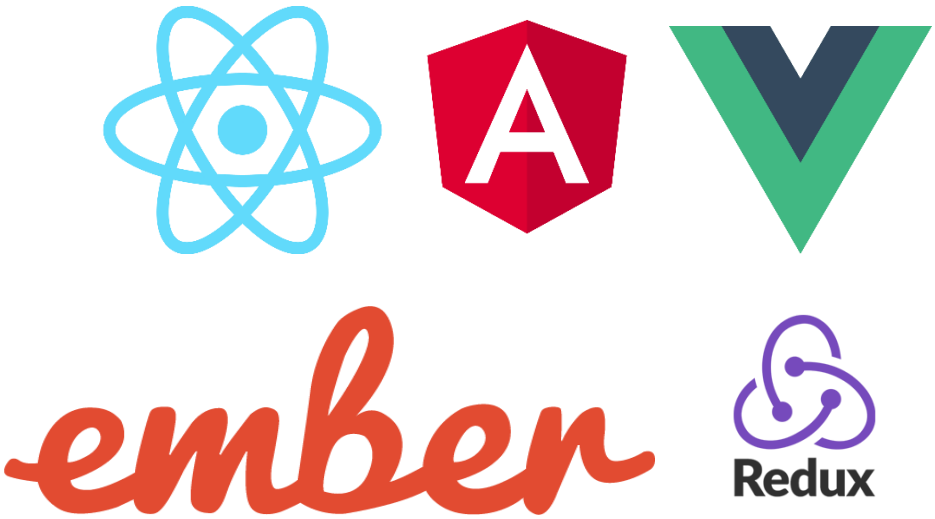

Рис 4. Бібліотеки та Фреймворки

В своїй дипломній роботі я використовував саме React, тому що ця технологія є однією з найпопулярніших веб-фреймворків на сьогоднішній день. React дозволяє розробникам будувати високопродуктивні веб-додатки з великою кількістю інтерактивності та динаміки. Крім того, React забезпечує зручний інтерфейс для створення компонентів, що дозволяє забезпечувати перевикористання коду та зменшення часу розробки. Я також використовував Redux - доповнення до React, яке забезпечує ефективну роботу зі станом додатку та зменшує складність управління станом.

РОЗДІЛ 6. ДИЗАЙН ТА ПРОГРАМНА ЧАСТИНА ПІДРУЧНИКА

6.1 Дизайн

При першому заході у підручник ми побачимо головну сторінку – привітання, де буде описано про що підручник. З лівого боку також побачимо вкладку «FAQ» – де стисло описано як користуватися підручником та яку саме інформацію можна знайти на певних сторінках.

На шапці побачимо основні надрозділи підручника – вступний «Прикладна логіка», «Логіки квазіарних предикатів» та «Нетрадиційні та модальні логіки»; із них користувач отримає основну інформацію з підручника про конкретні розділи та підрозділи.

У вступній вкладці «Прикладна логіка» описується основна тема підручника – прикладна логіка. Вона має за мету дослідження та формалізацію законів мислення, орієнтованих на застосування в різних прикладних областях. До прикладних відносять розділи математичної логіки, орієнтовані на розробку та використанням формальних логіко-математичних методів для моделювання та вирішення різноманітних задач найперше в галузях штучного інтелекту, інформатики, програмування. Водночас ідеї та методи математичної логіки з кожним роком усе глибше пронизують математику, філософію, лінгвістику, психологію.

Наступна вкладка на шапці – «Логіки квазіарних предикатів». Вона охоплює наступні розділи підручника:

- **Програмно-орієнтовані моделі логічних систем** – розділ присвячено проблемі побудови логічних формалізмів, орієнтованих на різноманітні задачі інформатики й програмування. Така побудова ведеться на базі спільного для логіки й програму-

вання композиційно-номінативного підходу. Наведено спектр композиційно-номінативних логік квазіарних предикатів.

- **Предикатні композиційні системи** – в цьому розділі описано композиційні системи та алгебри квазіарних предикатів. Визначено композиції цих предикатів, описано їх властивості.
- **Першопорядкові логіки квазіарних предикатів** – в цьому розділі вивчаються семантичні властивості чистих першопорядкових логік квазіарних предикатів. На множині формул мови цих логік введено низку відношень логічного наслідку.
- **Секвенційні числення** - вона охоплює розділ «Секвенційні числення першопорядкових логік із розширеними реномінаціями». Детально описано такі числення, наведено низку прикладів побудови виведень в цих численнях.

Вкладка «Нетрадиційні та модальні логіки» охоплює такі розділи:

- **Логіки предикатів над номінативними даними** – в цьому розділі основну увагу приділено вивченню логік над ієрархічними номінативними даними зі складеними іменами.
- **Нетрадиційні логіки** – в розділі вивчаються багатозначні логіки (зокрема, 3-значні й 4-значні) та інтуїціоністські логіки.
- **Модальні логіки** – в розділі описано традиційні модальні логіки (алетичні, темпоральні, епістемічні, деонтичні) і транзиційні модальні логіки часткових предикатів. Описано застосування темпоральних логік для специфікації та верифікації програм.

Остання вкладка підручника – «Інформація про автора» – сторінка з інформацією про автора, а також контактні дані та пошта, щоб користувач мав змогу зв’язатися з автором підручника та повідомити про проблеми, баги, або надати інформацію щодо оновлення змісту підручника.

Наявні скріншоти з дизайном розділів можна переглянути в додатку Б.

Рис 5. Головна сторінка

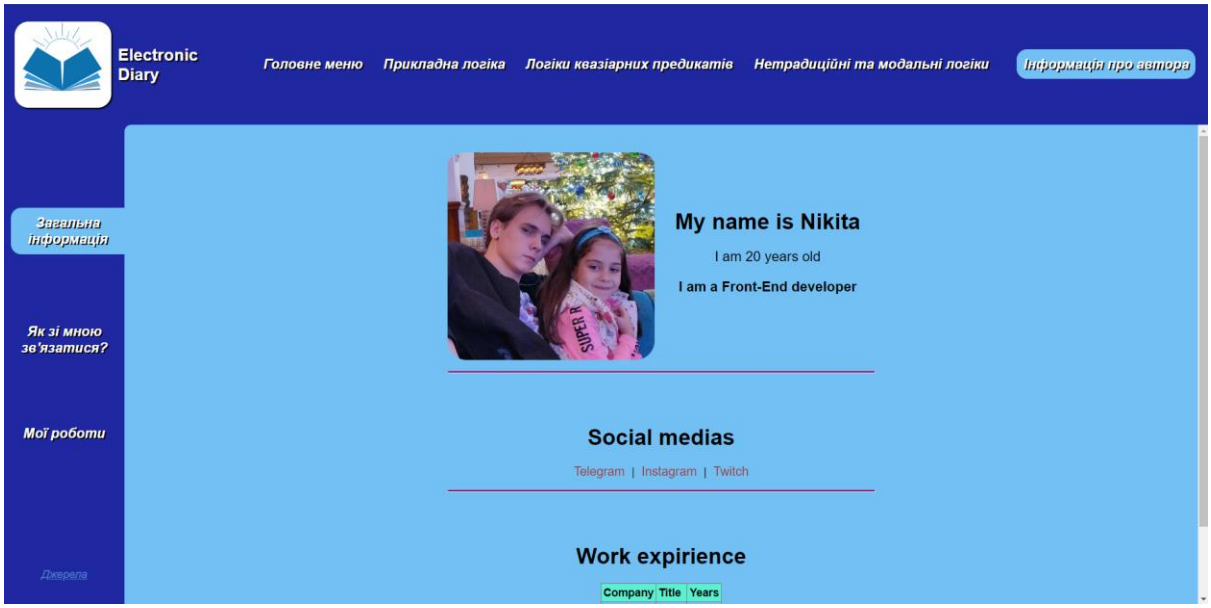

Рис 6. Сторінка «Інформація про автора»

Рис 7. Перелік джерел

6.2 Програмна частина

Важливою ідеєю щодо структури електронного підручника стало розбиття його на надрозділи, тому в коді було вирішено розробити окремі папки з окремими вкладками підручника:

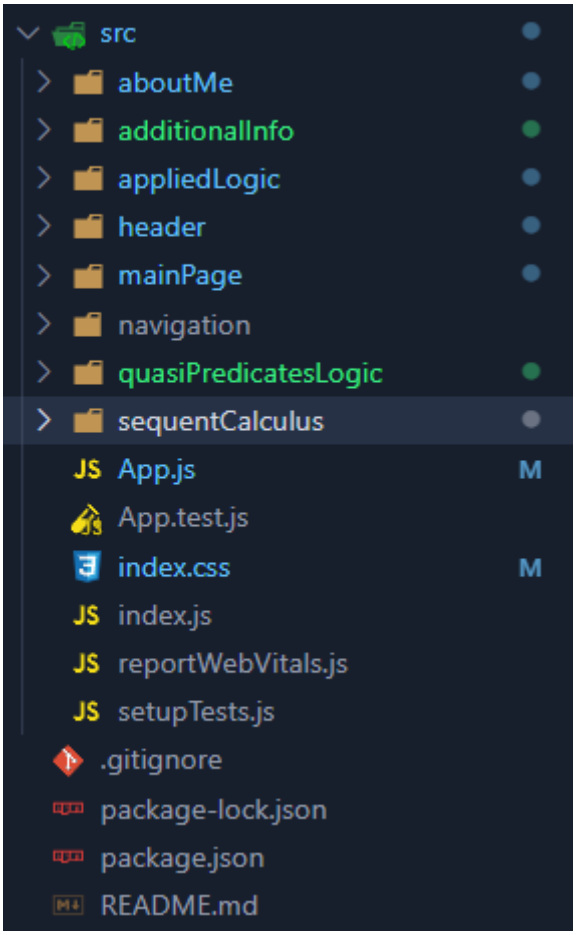

Рис 8. Повна структура проекту

Була розроблена основна шапка підручника, яка залишатиметься незмінною впродовж усього користування підручника:

Рис 9. Директорія «Header»

У самому коді як шапки, так і посилань на інші розділи, загалом було прийнято рішення використовувати Link та NavLink:

```
<div className="menu">
  <ul>
    <li>
      <Link to="/" className={splitLocation[1] === "" ? "newAdd"
      : splitLocation[1] === "HowToUse" ? "newAdd"
      : splitLocation[1] === "MainPageFirst" ? "newAdd"
      : ""}>Головне меню</Link>
    </li>
    <li>
      <Link to="/AppliedLogicInfo" className={splitLocation[1] === "AppliedLogicInfo" ? "newAdd"
      : splitLocation[1] === "AppliedLogicUses" ? "newAdd"
      : splitLocation[1] === "AppliedLogicPractice" ? "newAdd"
      : ""}>Прикладна логіка</Link>
    </li>
    <li>
      <Link to="/QuasiPredicatesLogicInfo" className={splitLocation[1] === "QuasiPredicatesLogicInfo" ? "newAdd"
      : splitLocation[1] === "QuasiPredicatesLogicUses" ? "newAdd"
      : splitLocation[1] === "QuasiPredicatesLogicPractice" ? "newAdd"
      : ""}>Логіки квазіарних предикатів</Link>
    </li>
    <li>
      <Link to="/SequentCalculusInfo" className={splitLocation[1] === "SequentCalculusInfo" ? "newAdd"
      : splitLocation[1] === "SequentCalculusUses" ? "newAdd"
      : splitLocation[1] === "SequentCalculusPractice" ? "newAdd"
      : ""}>Нетрадиційні та модальні логіки</Link>
    </li>
    <li>
      <Link to="/AboutMe" className={splitLocation[1] === "AboutMe" ? "newAdd"
      : splitLocation[1] === "MyContacts" ? "newAdd"
      : splitLocation[1] === "MyWorks" ? "newAdd"
      : ""}>Інформація про автора</Link>
    </li>
```

Рис 10. Програмний код шапки

Для навігації по підрозділах були розроблені функції навігації, які відсилали нас на конкретний підрозділ підручника.

```
const handleClickScroll1 = (el) => {
    const element = document.getElementById('1');
    if (element) {
      element.scrollIntoView({ behavior: 'smooth' });
    }
  };
```

Рис 11. Одна з функцій навігації

Повний код підручника можна продивитися в додатку А.

Плани на подальшу роботу

На даному етапі маємо наступні плани на подальшу розробку та покращення електронного підручника:

- Розробити спеціальний функціонал користувача: на даному етапі підручником може користуватися будь-хто, тому планується додати функціонал реєстрації та авторизації для студентів навчальних закладів, які мають право користуватися цим підручником.
- Додати візуалізацію до кожного розділу підручника: зараз користувач може отримати лише теоретичну частину та наявні приклади окремих розділів. Тому планується додати аудіо- та відеоматеріали до розділів підручника.
- Надалі поглиблювати інформацію з вже створених тем та передбачити додавання нових: підручник має достатньо інформації для вивчення потрібного матеріалу, але в майбутньому навчальний матеріал буде оновлюватись та доповнюватись, тому підручник має також оновлювати свій матеріал.

Висновки

Під час розробки електронного підручника були проаналізовано та опановано сучасні інформаційні технології. Для програмування використовувалась мова JavaScript, а також бібліотека React. Дизайн сторінок був створений з використанням HTML для розмітки та CSS для стилізації. Були додані функціональні та анімаційні елементи до сторінок та вкладок сайту.

На початковому етапі був розроблений план проекту та дизайн. Головними принципами були привабливий зовнішній вигляд та зручність використання. На головній сторінці була розміщена вкладка з поясненнями щодо використання сайту. Також була включена інформація про автора в електронному підручнику. Для забезпечення зручності користувачів, була реалізована можливість переходу на будь-яку вкладку підручника з будь-якої частини сайту без затримок завантаження сторінок.

На наступному етапі були вибрані оптимальні технології для вирішення як поточних, так і майбутніх проблем, як у функціональній, так і у візуальній частині. При цьому були використані:

- Редагувальне середовище VS Code для зручної розробки;
- Мова програмування JavaScript для реалізації логіки;
- Бібліотека React для створення компонентів;
- Мова розмітки HTML для структуризації сторінок;
- Мова стилів CSS для оформлення вигляду сторінок.

На останньому етапі було проведене тестування з метою знаходження та усунення проблем та багів. Було проведено тестування функціональності шляхом перевірки роботи всіх вкладок, кнопок та інтерактивних елементів. Використовувалися різні сценарії. Було виявлено та вирішено декілька наявних проблем та помилок – несправні посилання,

несправна робота на деяких пристроях та некоректна поведінка функціональності. Після вирішення проблем було проведено ще декілька тестувань для остаточної перевірки функціональності.

Після успішного завершення всіх трьох етапів можна зробити висновок, що поставлену роботу успішно виконано - електронний підручник створено. Основні технології були проаналізовані, вивчені та успішно впроваджені. Були виявлені та вирішені проблеми, які виникали на етапах планування та розробки, а також сформульовані майбутні цілі для подальшого вдосконалення підручника.

Таким чином в процесі виконання випускної кваліфікаційної роботи бакалавра отримано наступні результати:

– створено електронний підручник з курсу «Прикладна логіка» із розвинутими методами надання інформації користувачам;

– проаналізовано та опановано засоби для розробки браузерних застосунків, вироблено практичні навички щодо використання бібліотек та фреймворків;

– отримано знання щодо програмно-орієнтованих логік квазіарних предикатів та їх секвенційних числень, про модальні й темпоральні логіки та про їх застосування в інформатиці й програмуванні.

Перелік джерел

1. Посібник JavaScript. **Режим доступу:** https://uk.javascript.info/
2. Посібник React. **Режим доступу**: https://uk.reactjs.org/tutorial/tutorial.html
3. Посібник Redux. **Режим доступу:** https://react-redux.js.org/introduction/getting-started
4. М.С. Нікітченко, С.С. Шкільняк. Математична логіка та теорія алгоритмів – К.: ВПЦ "Київський ун-т", 2008. – 528 с.
5. М.С. Нікітченко, С С. Шкільняк. Прикладна логіка – К.: ВПЦ "Київський ун-т", 2013. – 278 с.
6. М.С. Нікітченко, О.С. Шкільняк, С.С. Шкільняк. Чисті першопорядкові логіки квазіарних предикатів // Пробл. програмування. – 2016. – № 2–3. – С. 73–86.
7. О.С. Шкільняк. Транзиційні модальні логіки немонотонних квазіарних предикатів // Компьютерная математика. – 2014. – Вып. 2. – С. 99–110.
8. О.С. Шкільняк, С.С. Шкільняк. Першопорядкові секвенційні числення логік квазіарних предикатів з розширеними реномінаціями та рівністю // Пробл. програмування. – 2022. – № 3–4.
9. С.С. Шкільняк. Математична логіка. Приклади й задачі. – К.: ВПЦ "Київський ун-т", 2022. – 304 с.
10. Mykola S. Nikitchenko and Stepan S. Shkilniak. Algebras and logics of partial quasiary predicates // Algebra and Discrete Mathematics, Vol. 23 (2017). No 2, pp. 263–278.
11. M. Nikitchenko, O. Shkilniak, S. Shkilniak. Sequent calculus for a program-oriented predicate logic over complex-named data // 10th Int. Conf. on Advanced Computer Information Technologies (ACIT), Deggendorf, Germany, 2020, pp. 497–500.

Додатки

Додаток А.

Посилання на код електронного підручника:

https://github.com/NikitaOrliak/React-Project

Додаток Б.

Дизайн основних сторінок.

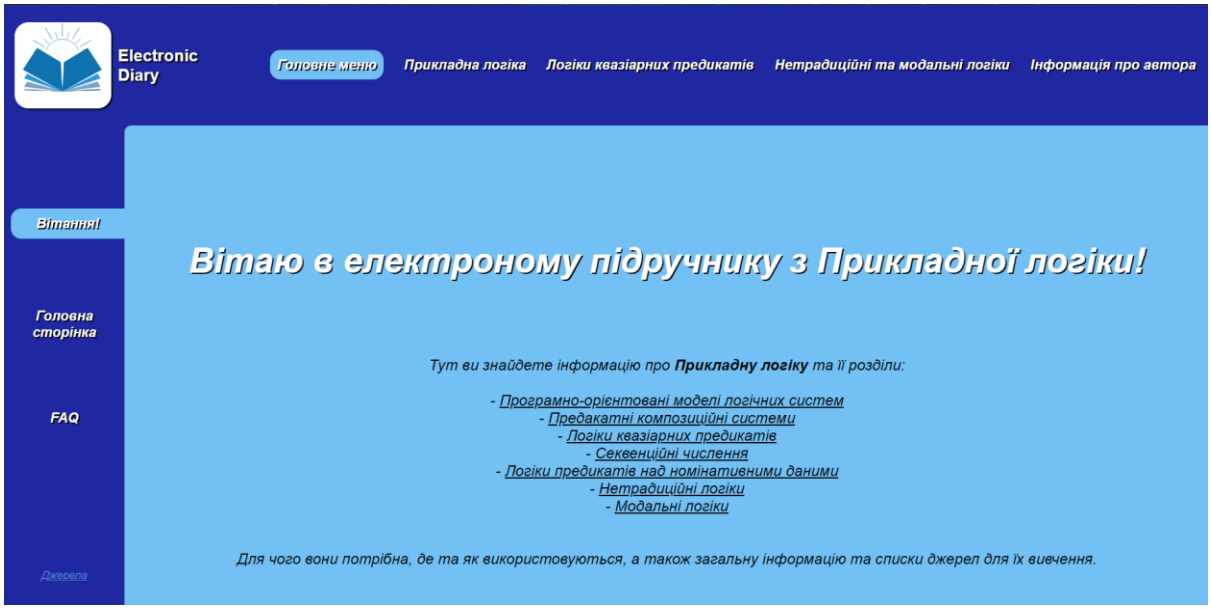

Рис 12. Сторінка привітання

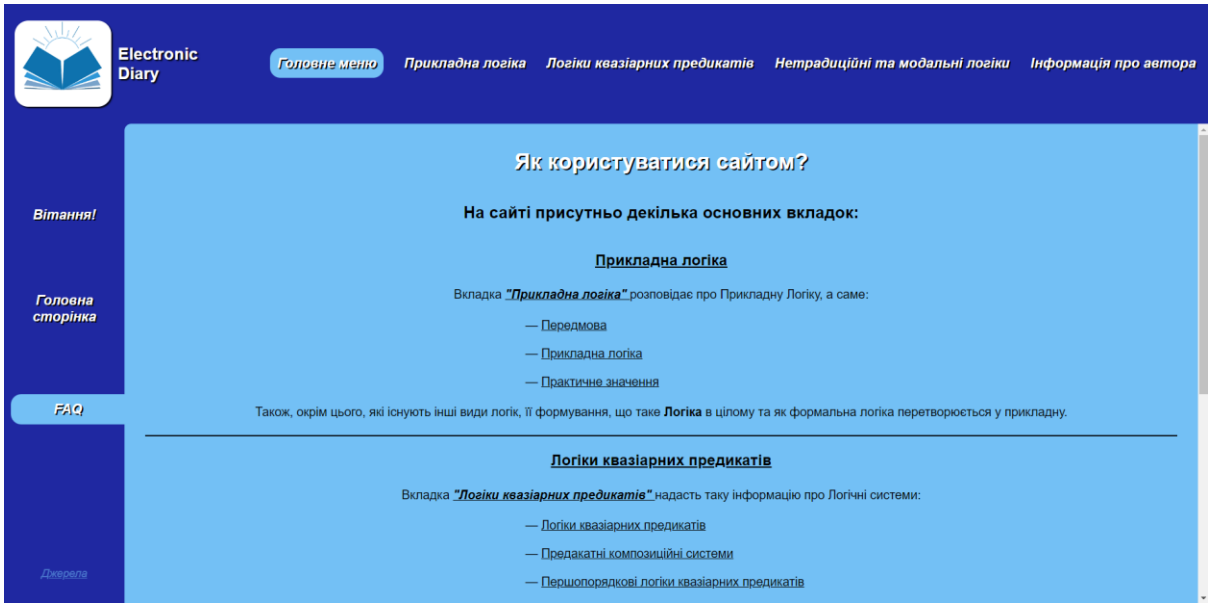

Рис 13. Сторінка FAQ

Рис 14. Сторінка «Передмова»

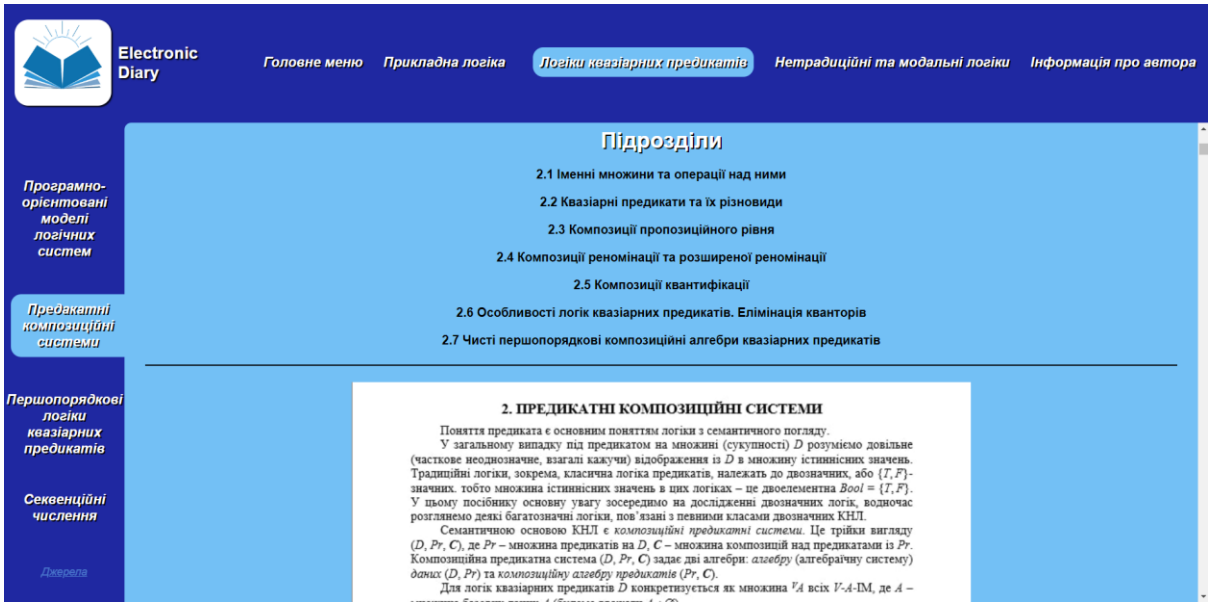

Рис 15. Реалізація однієї з навігацій