Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Хайтул Іллі Володимировича*
(ПІБ)

академічної групи *122-20-1*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми Комп'ютерні науки
(назва освітньої програми)

на тему: *Розробка програмного забезпечення для управління та контролю дослідницьких експериментів в межах проекту Lab4All з використанням мови програмування JavaScript*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Спірінцев В.В.*			
розділів:				
спеціальний	*доц. Спірінцев В.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-20-1* *Хайтул Іллі Володимировича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення для управління та контролю дослідницьких експериментів в межах проекту Lab4All з використанням мови програмування JavaScript*

затверджена наказом ректора НТУ «ДП» від *23.05.2024 р.* № *469-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***15.06.2024.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***20.06.2024 р.***

Завдання видав _____ *доц. Спірінцев В.В*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Хайтул І.В*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *24.06.2024 р.*

РЕФЕРАТ

Пояснювальна записка: 105 с., 41 рис., 5 дод., 40 джерел.

Об'єкт розробки: програмне забезпечення для з’єднання та взаємодії з дослідницьким обладнання в обох університетах в межах проекту Lab4All.

Мета кваліфікаційної роботи: оптимізація та модифікація програмного забезпечення для надання учням онлайн доступу до експериментів проекту.

У вступі розглядається актуальність автоматизації наукових досліджень, особливо в рамках проекту Laboratories Across Borders. Підкреслюється важливість та актуальність використання програмного забезпечення, а також вибір мови JavaScript.

У першому розділі розглядається значення автоматизації в наукових дослідженнях, її вплив на різні галузі науки та важливість для забезпечення точності та ефективності експериментів. Також проводиться аналіз існуючих проблем і аналогів, таких як обмежений доступ до лабораторій, важливість підтримання безпеки та неефективне використання ресурсів.

У другому розділі описується процес проектування та розробки інформаційної системи. Детально розглядаються використані технології та мови програмування, такі як JavaScript, середовище Node.js та його модулі, а також їхні переваги над схожими технологіями. Крім того, в цьому розділі описується структура системи, її алгоритми функціонування, вимоги до технічних засобів та програмної сумісності.

В економічному розділі визначено обсяг трудовитрат на розробку забезпечення, проведено підрахунок вартості роботи по його створенню та обчислено час, потрібний для виконання поставленого завдання.

Практичне значення дипломної роботи полягає у створенні програмного забезпечення, яке дозволяє ефективно управляти та контролювати дослідницькі експерименти проекту Lab4All, забезпечує віддалений доступ до лабораторного обладнання, автоматизацію певних фізичних процесів, а також створення зручного інтерфейсу для взаємодії користувачів із системою.

Актуальність розробки програмного застосунку полягає в наданні віддаленого доступу та взаємодію учнів із дослідженнями в лабораторіях університетів, що особливо потрібно при скрутному воєнному часі в Україні.

Список ключових слів: LAB4ALL, ЕКСПЕРИМЕНТ, ДОСЛІДЖЕННЯ, МОДУЛІ, ДИСТАНЦІЙНЕ УПРАВЛІННЯ, АВТОМАТИЗАЦІЯ, ОПТИМІЗАЦІЯ.

ABSTRACT

Explanatory note: 105 pages, 41 fig., 5 appendices, 40 sources.

Object of development: software for connecting and interacting with research equipment in both universities within the Lab4All project.

The purpose of the qualification work: optimisation and modification of the software to provide students with online access to the project experiments.

The introduction describes the relevance of research automation, especially within the Laboratories Across Borders project. The importance and relevance of using software is emphasised, as well as the choice of the JavaScript language.

The first section includes the importance of automation in scientific research, its impact on various fields of science, and its importance for ensuring the accuracy and efficiency of experiments. It also analyses existing problems and analogues, such as limited access to laboratories, the importance of maintaining security, and inefficient use of resources.

The second section, the process of designing and developing the information system. The technologies and programming languages used, such as JavaScript, the Node.js environment and its modules, are discussed in detail, as well as their advantages over similar technologies. In addition, this section describes the structure of the system, its functioning algorithms, hardware and software compatibility requirements.

The economic section determines the amount of labour costs for the development of the software, calculates the cost of its creation and estimates the time required to complete the task.

The practical significance is the creation of software that allows for effective management and control of Lab4All research experiments, provides remote access to laboratory equipment, automation of certain physical processes, and the creation of a user-friendly interface for user interaction with the system.

The relevance of the software application development is to provide remote access and interaction of students with research in university laboratories, which is especially necessary in the difficult wartime situation in Ukraine.

List of keywords: LAB4ALL, EXPERIMENT, RESEARCH, MODULES, REMOTE CONTROL, AUTOMATION, OPTIMISATION.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ОС - операційна система;

ПЗ - програмне забезпечення;

DDoS - Distributed Denial-of-Service Attack;

Lab4All - Laboratory Across Borders;

ЕОМ - електронно-обчислювальна машина;

API - Application Programming Interface;

HTTP - Hypertext Transfer Protocol;

ЗМІСТ

ВСТУП

Сучасна наука потребує високої точності та ефективності при проведенні дослідницьких процесів. Збільшення обсягу досліджень та зростання складності експериментів вимагає застосування нових технологічних рішень для їхнього управління та контролю. Важливою частиною цього процесу є використання програмного забезпечення, яке дозволяє автоматизувати та оптимізувати роботу дослідників, забезпечуючи надійність та точність отриманих результатів.

Розроблене програмне забезпечення призначене для застосування у сфері управління та контролю дослідницьких експериментів.

Наукові експерименти — це складні процеси, які потребують ретельного планування, контролю та аналізу результатів. Від якості та точності експериментів залежить достовірність наукових даних і, як наслідок, успіх досліджень. Сучасні технології дозволяють створювати програмне забезпечення, яке полегшує управління експериментами, підвищує ефективність роботи дослідників та знижує ризики помилок.

Lab4All - це ініціатива, розпочата Національним технічним університетом «Дніпровська політехніка» та Університетом Ройтлінгена [1]. Головна мета якого – створення кібер-фізичних лабораторій та розробка індивідуальних навчальних модулів для студентів в обох університетах. «Lab4All» надає студентам можливість віддалено проводити лабораторні експерименти через інтернет, використовуючи реальне обладнання розміщене в лабораторіях обох університетів.

Проект спрямований на створення універсальної платформи для проведення наукових досліджень, яка забезпечує автоматизацію процесів експериментування, збору, аналізу та моніторингу даних в режимі реального часу. Використовуючи сучасні технології, Lab4All забезпечує зручний інтерфейс, інтеграцію з різними приладами та системами, а також високу надійність і безпеку зберігання даних.

Мову програмування JavaScript було обрано з огляду на його можливості для створення інтерактивних веб-додатків, що забезпечують зручний та багатофункціональний інтерфейс користувача [2]. Це є дуже важливим для забезпечення комфорту та ефективності в роботі дослідників. Також, JavaScript - це одна з найпоширеніших мов програмування з існуючих, і це надає нам, як розробникам, поширений доступ до великої кількості різноманітних бібліотек та фреймворків для більш простішої та розвинутої розробки.

Актуальність роботи полягає у необхідності проектування та створення надійного та ефективного інструменту для управління науковими дослідженнями, який сприятиме підвищенню ефективності та покращенню продуктивності експериментальної діяльності.

Метою моєї кваліфікаційної роботи є розробка та оптимізація системи для управління дослідницькими експериментами, яка дозволить автоматизувати ключові процеси та забезпечити точний контроль за виконанням експериментів.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- аналіз вимог до розробки управління фізичними експериментами;
- розробка алгоритмів та реалізація функціональної системи;
- тестування та оптимізація програмного забезпечення.

Основними функціональними задачами можна виділити:

- автоматизація процесу збору та обробки даних;
- надання змоги користування та взаємодії з фізичними експериментами.

В цілому, реалізоване програмне забезпечення значно спростить процес проведення дослідницьких експериментів, знизить ризик втрати або спотворення даних, а також забезпечить ефективне управління та контроль за виконанням експериментів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКИ ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

1.1.1. Значення автоматизації в наукових дослідженнях

Автоматизація наукових досліджень відіграє ключову роль у сучасному світі, дозволяючи значно підвищити ефективність, точність та безпеку проведення експериментів. Це забезпечує стандартизацію процесів, економію часу та ресурсів, покращує управління даними і підтримує інновації. У рамках проекту Lab4All автоматизація дослідницьких експериментів дозволяє створити ефективну та доступну платформу для управління науковими дослідженнями.

Автоматизація має значний вплив на різні наукові галузі. У біології та медицині, наприклад, автоматизовані системи для аналізу зображень допомагають швидко та точно діагностувати захворювання [3]. У хімії автоматизовані синтезатори дозволяють створювати нові матеріали та хімічні сполуки з високою точністю [4]. В фізиці та інженерії автоматизовані вимірювальні системи забезпечують точні результати експериментів та комп'ютерного моделювання [5]. У соціальних науках та гуманітаристиці автоматизація допомагає аналізувати великі масиви текстових даних та досліджувати соціальні мережі [6]. Також, однією з ключових її переваг є вплив на репродуктивність наукових досліджень. Автоматизовані системи дозволяють точно відтворювати умови та процедури експериментів, що є критично важливим для перевірки та підтвердження наукових результатів.

Використання автоматизований систем дозволяє зменшити вплив людського фактору, що мінімізує ризик помилок при зборі та обробці отриманих даних під час експерименту. Автоматизація також дозволяє значно заощадити час, необхідний для проведення і оптимізації експериментів, та використання їх

ресурсів. Автоматизовані системи допомагають як найбільш точніше та швидше виконувати запрограмовані дії експерименту, аніж людина, що дозволяє дослідникам витрачати час для більш теоретичної роботи і майбутнього покращення та програмування цих систем.

Важливість наявності автоматизації також можна прогледіти у високому поліпшенню управлінь даними. У більшості випадків, під час наукових досліджень генерується великий обсяг отриманої інформації. Спеціалізоване програмне забезпечення для управління даними дозволяє централізовано зберігати, організовувати, обробляти та аналізувати дані, що полегшує доступ до них і забезпечує їх збереження.

Більш доступне та легке масштабування наукових експериментів, збільшення кількості досліджуваних зразків або її повторень, при цьому без значного збільшення витраченого часу та ресурсів, також є досить важливою ознакою при наявності автоматизованого процесу. Це дає можливість проводити більш комплексні та детальні дослідження.

Також автоматизовані системи забезпечують високу відтворюваність експериментів. Використання однакових автоматизованих процесів для проведення експериментів гарантує, що всі зразки обробляються однаково, що важливо для отримання надійних і порівнянних результатів. Також, забезпечення збереження важливих результатів досліджень, їх висновків, у випадку технічних помилок. Крім того, це дає можливість доступу до даних з різних місць, що також є важливим аспектом і сприяє покращенню співпраці між роботою з різних країн світу. Машини на сьогодні здатні ефективно допомагати у прийнятті рішень під час наукових експериментів. Системи, що обладнані алгоритмами машинного навчання або штучним інтелектом, можуть без проблем, з величезною швидкістю провести аналітику великого обсягу даних, виявити закономірності та тенденції, а також, навіть, запропонувати оптимальні рішення для вирішення певних проблем та подальших досліджень.

Майбутнє автоматизації у наукових дослідженнях виглядає надзвичайно

перспективним. Завдяки стрімкого розвитку таких важливих технологій, як штучний інтелект, машинне навчання та робототехніка - автоматизовані системи стають дедалі більш ефективними та функціональними. Такі технології відкривають нові можливості для науковців, дозволяючи їм проводити більш складні, точні та масштабні дослідження. Також, більш детально розглянемо ці ключові напрямки та їх перспективи.

Розвиток штучного інтелекту та машинного навчання відіграє важливу роль у сучасних наукових рішеннях:

– «Аналіз великих даних (Big Data)»: ШІ (штучний інтелект) здатний обробляти великі обсяги даних швидко та точно, що дозволяє виявляти складні закономірності і тренди, які не піддаються традиційним методам аналізу [7]. Це особливо корисно в таких галузях, як геноміка, кліматологія та соціальні науки.

– «Автоматизоване моделювання та прогнозування»: Моделі МН (машинне навчання) використовуються для створення точних прогнозів на основі великих масивів історичних даних, що є важливим у таких галузях, як кліматологія, біологія та економіка, що надає можливість науковцям передбачати майбутні події та розробляти стратегічні плани [8].

– «Розпізнавання зображень та обробка природної мови»: ШІ дозволяє автоматизувати аналіз зображень і текстів, що значно прискорює дослідження в медичних, соціальних та гуманітарних науках. Наприклад, автоматизовані системи діагностики зображень можуть швидко і точно ідентифікувати патології на медичних знімках.

– «Збір даних у режимі реального часу»: Обладнання можуть безперервно збирати дані з навколишнього середовища, що забезпечує дослідників актуальною інформацією [9].

– «Дистанційний моніторинг та управління»: Науковці можуть контролювати експерименти і обладнання на відстані, що полегшує проведення досліджень у важкодоступних місцях.

Робототехніка продовжує вдосконалюватися, відкриваючи все нові

горизонти для успішної автоматизації наукових досліджень:

– «Автоматизація лабораторних процесів»: Роботи можуть автоматизувати рутинні лабораторні операції, такі як підготовка зразків, виконання хімічних реакцій і аналіз результатів, що значно підвищує ефективність.

– «Безперервність і надійність експериментів»: Автоматизовані системи можуть працювати без перерви, надаючи стабільність і відтворюваність експериментів. Це може бути корисним особливо для довготривалих досліджень, які потребують постійної роботи обладнання.

– «Безпека досліджень»: Використання роботів зменшує ризики для дослідників при роботі з небезпечними речовинами або в умовах високих температур і тисків, що дозволяє проводити експерименти, які були б надто небезпечними для людини.

1.1.2. Аналіз існуючих проблем та аналогів

Дуже цікаво розібрати існуючі проблеми, з якими стикаються дослідники та студенти під час проведення наукових експериментів та які, доволі часто, можна зустріти і в аналогах. І, як саме, проект Lab4All може вирішувати такі проблеми та запропонувати нові можливості.

– «Обмежений доступ до лабораторії». Звичайні лабораторії, які знаходяться окремо чи в університетах вимагають фізичної присутності студентів та майбутніх дослідників, що обмежує можливості для повноцінного навчання. Такі проблеми можуть починатися від витраченого часу з взаємодією з експериментами та обмеженого місця в лабораторії, до непередбачуваних ситуацій, наприклад, військового стану у країні, коли студентам фізично дуже складно та небезпечно бути постійно присутнім в університеті для опрацювання своїх лабораторних завдань. У свою чергу проект Lab4All надає можливість віддаленого доступу студентам до лабораторного обладнання через інтернет. Це

дозволяє студентам проводити експерименти з будь-якого місця у будь-який час, що знімає обмеження, пов'язані з фізичним доступом до лабораторій.

– «Високі витрати на обладнання та обслуговування». Лабораторне обладнання є дорогим і вимагає регулярного технічного обслуговування, що створює фінансовий тягар для навчальних закладів. Це особливо актуально для університетів з обмеженими бюджетами. Віддалені лабораторії у межах проекту Lab4All, дозволяють знизити витрати на обладнання та його обслуговування, оскільки одне і те саме обладнання може використовуватися багатьма студентами з різних університетів.

– «Неефективне використання ресурсів». Більшість фізичних лабораторій часто використовуються неефективно через обмежений графік роботи. Це призводить до простою обладнання та нераціонального використання ресурсів. За рахунок віддаленого доступу до лабораторій обладнання може використовуватися цілодобово, що підвищує його ефективність та продуктивність. Також, це дає можливість студентам використовувати лабораторні експерименти у будь яких вільних для них час.

– «Обмежені можливості для співпраці». Співпраця між університетами часто обмежується через різні стандарти, відсутність уніфікованих платформ та складності в організації спільних проектів. Це призводить до обмеженого обміну знаннями та досвідом між установами. «Lab4All» створює спільну платформу, яка дозволяє університетам обмінюватися ресурсами та досвідом. Будь який студент, не тільки співпрацюючих університетів, має змогу ознайомитися з лабораторним арсеналом одразу двох університетів, та взаємодіяти з ними, що дає більшу інформованість про заклади навчання, та деяку економію, так як, не потрібно мати теж саме обладнання в різних університетах одночасно, а зосередитися на встановленні нових для різноманітних видів дослідів фізичних явищ. Таким чином, проєкт «Lab4All» сприяє вирішенню ключових проблем сучасної лабораторної освіти, надаючи інноваційні рішення для покращення доступу, ефективності та якості навчання водночас.

– «Безпека та потенційні ризики». Деякі експерименти можуть бути потенційно небезпечними або ризикованими, що вимагає додаткових заходів безпеки та обмежень, які в свою чергу, також переносять нас на усі проблеми, згадані вище. Натомість, за допомогою онлайн лабораторії «Lab4All», проведення експериментів зменшує ризики та забезпечує безпечне середовище для студентів та дослідників.

Під час пошуку в інтернет просторі не було виявлено прямих аналогів до проекту «Lab4All», але одним із наближених до спільних цілей є проект «Go lab». Проект «Go lab» та «Lab4All» мають спільну мету – забезпечення доступу до дистанційних та онлайн лабораторій для студентів, але вони різняться за своїм підходом, масштабом і вирішенням певних проблем. Мною було проведено детальний аналіз цих двох проектів, з особливою увагою до знаходження певних проблем та недоліків, які «Lab4All» не має або може вирішити, на відміну від «Go lab». З основних виділяють наступні:

– «Інтерфейс користувача в обох проектах».

З початку може здатися, що UI/UX є не дуже важливим аспектом порівняння двох проектів, але як на мене, інтуїтивно-зрозумілий інтерфейс є дуже важливою складовою для забезпечення залученості та поглибленого використання проекту майбутніми студентами. Інтерфейс «Go lab» (рис.1.1) можна критикувати за його складність та топорність. Навігація може бути неінтуїтивною, що ускладнює роботу користувачів, особливо новачків. Користувачам часто доводиться витрачати багато часу на пошук потрібних інструментів та функцій, що може спричинити фрустрацію. Наявність численних меню та підменю ускладнює навігацію та може знизити ефективність роботи.

У свою чергу «Lab4All» (рис.1.2) був розроблений з урахуванням сучасних принципів UX/UI дизайну. Інтерфейс інтуїтивно зрозумілий, що значно зменшує кількість часу та проблеми, які можуть з'являтися під час першої зустрічі з сайтом проекту. Це дозволяє швидше адаптуватися та ефективніше використовувати платформу.

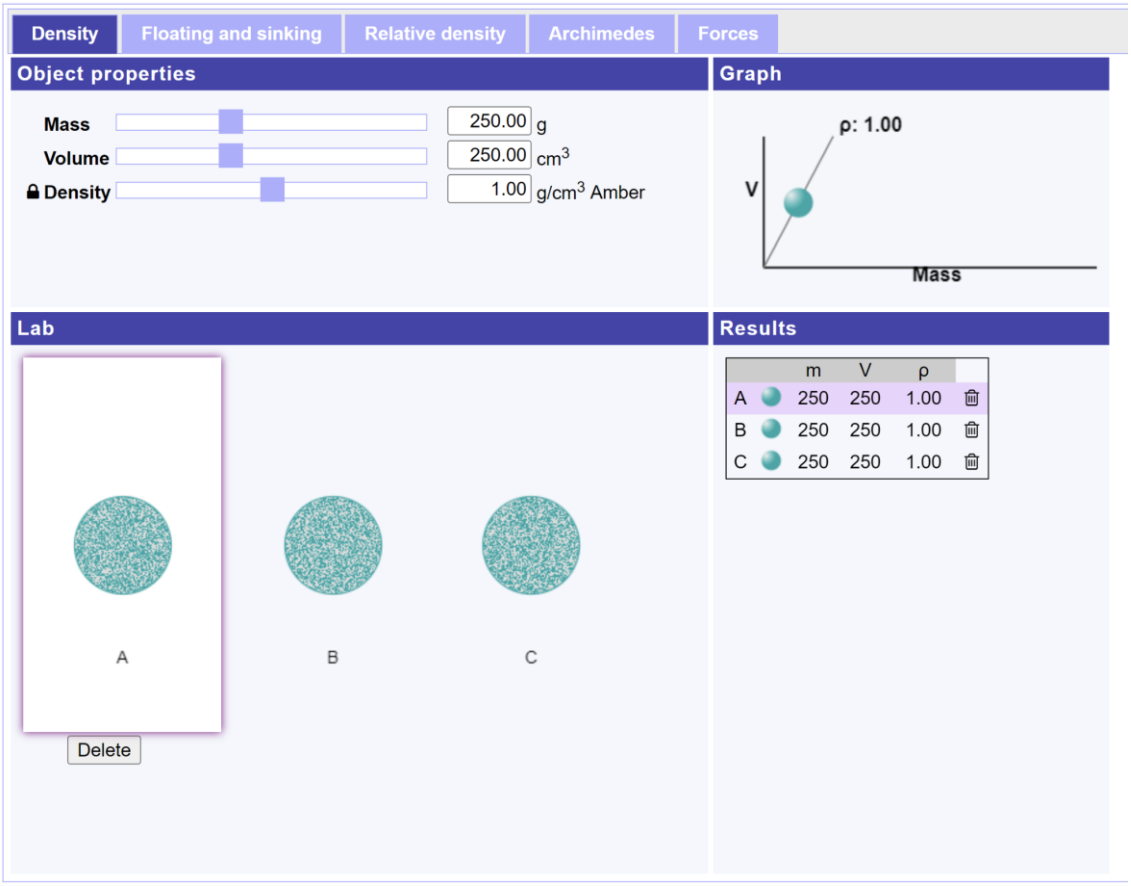

Рис.1.1. Приклад інтерфейсу одного з експериментів проекту «Go lab»

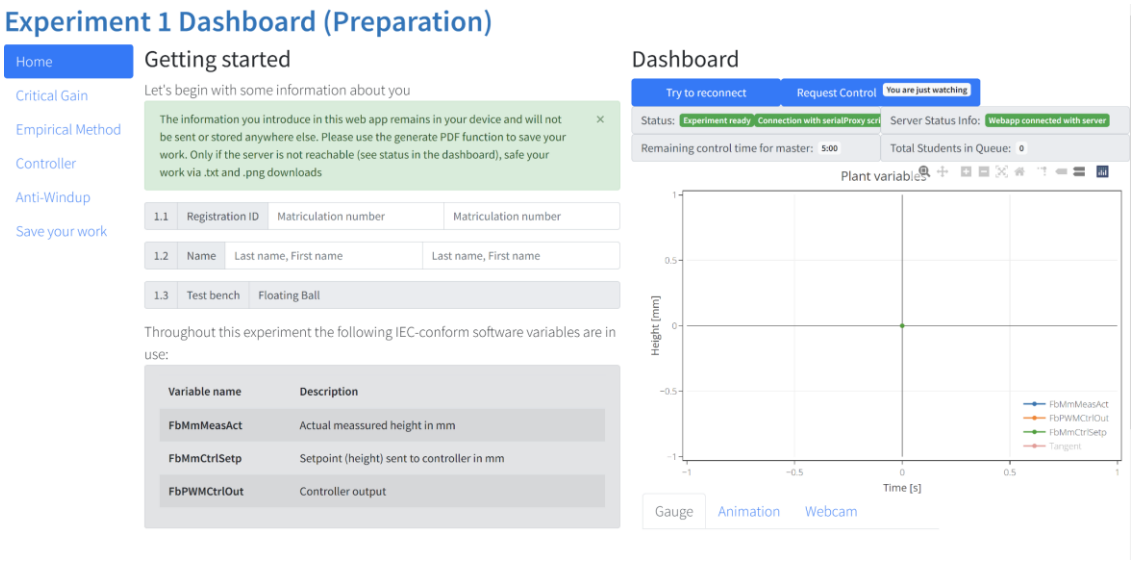

Рис.1.2. Приклад вигляду оригінального інтерфейсу проекту Lab4All

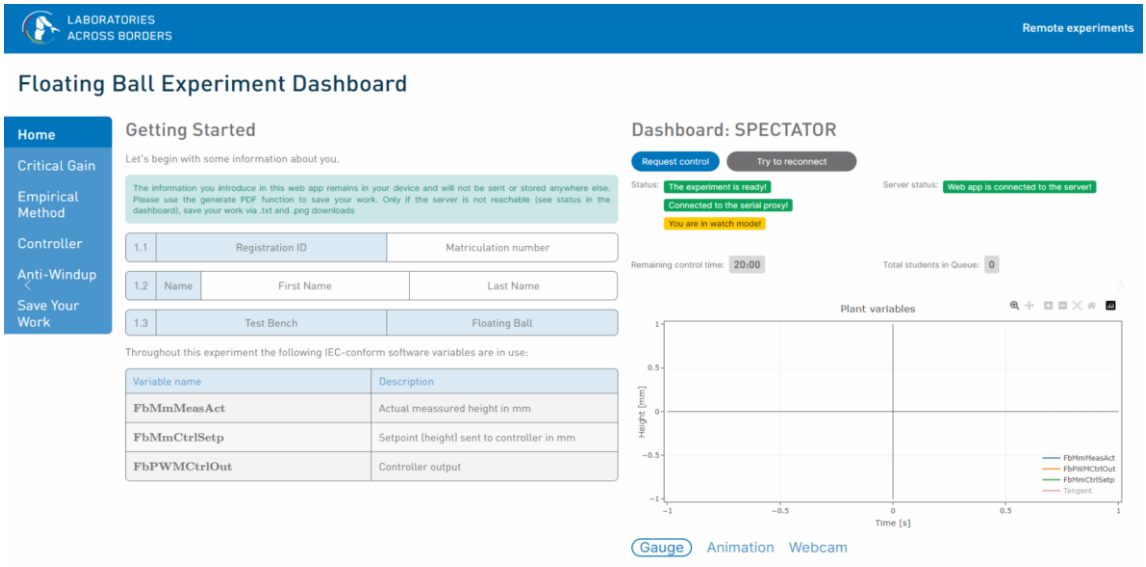

Рис.1.3. Приклад вигляду модернізованого інтерфейсу проекту «Lab4All»

Також, адаптивність інтерфейсу дозволяє користувачам працювати з платформою з будь-якого пристрою, що значно підвищує мобільність та гнучкість їхньої роботи.

Гнучкість налаштувань дозволяє дослідникам адаптувати інструменти до специфічних потреб своїх експериментів, значно підвищуючи їх ефективність та інноваційність досліджень. Якщо цього немає, це може стати досить серйозною проблемою під час навчання, так як буде не вистачати різноманітності у випадках тестування фізичних експериментів, для отримання нових та унікальних результатів досліджень.

Також, однією з важливих відмінностей є те, що у проекті «Go lab» я не знайшов теоретичних описів або документацій до кожного із експериментів, в якому було б описано принципи та інструкція з використання та взаємодії з дослідницьким обладнанням. На відміну від «Lab4All», в якій можна знайти детальний опис взаємодії з віддаленою лабораторією.

Отже, виходячи з наявної інформації, можна виділити, що «Lab4All» має певні переваги завдяки відсутності проблем, з якими стикається «Go lab» та пропонуванню нових адаптованих рішень, що робить його більш ефективним та зручним середовищем для студентів.

1.2. Призначення розробки та галузь застосування

У межах кваліфікаційної роботи розглядається розроблене програмне забезпечення, для управлінням експериментів для проекту Lab4All - «Laboratories Across Borders». Це система, яка забезпечує роботу над автоматизацією у проведенні дослідів з використанням фізичних експериментів.

Необхідність розробки програмного забезпечення виникла через потребу в ефективному управлінні та контролі дослідницьких експериментів, що знаходяться в двох університетах, українському та німецькому. Даний проект спрямований на створення онлайн лабораторій з відкритим доступом для студентів різних університетів, що дозволяє виконувати лабораторні експерименти дистанційно через інтернет, використовуючи реальне обладнання. Також, програмне забезпечення призначене для забезпечення безперебійного та зручного управління експериментами, збору даних та аналізу результатів. Це дозволить ефективно використовувати обмежені ресурси лабораторного обладнання, надаючи віддалений доступ для студентів та дослідників.

Деякі причини впровадження проекту:

– «Забезпечення віддаленого доступу до лабораторного обладнання». Студенти мають змогу отримувати доступ до реального лабораторного обладнання на відстані, онлайн, без необхідності фізичної присутності.

– «Ефективне використання обмеженої кількості обладнання». Розподіл доступу до обладнання між більшою кількістю користувачів, підвищуючи ефективність його використання.

– «Необхідність автоматизації процесів».

– «Збір, зберігання та аналіз даних, отриманих під час експериментів». Дозволяє отримувати та зберігати отримані дані, робити обґрунтовані висновки та приймати науково обґрунтовані рішення.

– «Створення зручного інтерфейсу для взаємодії студентів із системою». Це забезпечує інтуїтивно зрозумілу взаємодію з експериментами , що полегшує

процес досліду та більш ефективне навчання.

– «Економічна ефективність». Замість необхідності придбання додаткового дорогого обладнання в кожні зацікавлені навчальні заклади, проект дозволяє максимально використовувати наявні ресурси обох університетів.

Усі ці причини та фактори роблять проект Lab4All дуже ефективним та простим засобом для впровадження більш зручного навчання для студентів вищих закладів освіти різних країн світу.

Загальна термінологія вміщує в собі терміни з галузей інформаційних технологій та наукових досліджень:

– «Кібер-фізична або онлайн лабораторія» - лабораторне середовище, в якому реальне фізичне обладнання поєднується з віртуальними елементами керування.

– «Автоматизація» - це процес використання технологій для виконання завдань з мінімальним втручанням людини, з метою підвищення ефективності, точності та продуктивності.

Також, важливо виділити галузі, де проект Lab4All може бути корисним та активно використовуватися:

– Вищі навчальні заклади світу. Проект може бути цікавим для впровадження його у навчальний план, для зручного та ефективного проведення фізичних дослідів та набування практичного досвіду. Університети та коледжі, що пропонують програми навчання у сферах мехатроніки, робототехніки, електротехніки, машинобудування та суміжних дисциплін.

– Стартапи та інноваційні компанії. Новостворені компанії, що розробляють інноваційні рішення у сферах робототехніки, автоматизації, фізичних систем та суміжних галузях. Також можливість доступу до фізичного обладнання для проведення експериментів та тестування прототипів без необхідності значних інвестицій на ранніх стадіях.

– Незалежні дослідники та ентузіасти. Окремі студенти, дослідники або ентузіасти, що працюють над проектами спільних галузях. Можливість

проводити фізичні експерименти та тестувати ідеї без необхідності придбання власного дорогого обладнання.

– Інженерні проекти. У сфері інженерії Lab4All використовує онлайн лабораторій, що забезпечують можливість проведення експериментів з використанням роботів, електродвигунів та іншого обладнання. Це дозволяє інженерам тестувати нові технології та матеріали, а також розвивати і впроваджувати інновації в різних інженерних дисциплінах.

Таким чином, Lab4All є універсальним інструментом, що може бути ефективно використаний в різних галузях для покращення управління та проведення дослідницьких експериментів в освітніх цілях.

1.3. Підстава для розробки

Підставами для розробки (виконання кваліфікаційної роботи) є:

– освітня програма спеціальності 122 «Комп'ютерні науки»;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» №469-с від 23.05.2024 р.;

– завдання на кваліфікаційну роботу на тему «Розробка програмного забезпечення для управління та контролю дослідницьких експериментів в межах проекту Lab4All з використанням мови програмування JavaScript».

1.4. Постановка завдання

Мета: оптимізувати та дорозробити програмний код, використовуючи мову програмування JavaScript, який надасть змогу взаємодії майбутнього користувача з реальним фізичним обладнанням для виконання певних досліджень у межах лабораторій Lab4All.

Призначення: забезпечити віддалений доступ до реального лабораторного

обладнання та проведення експериментів через Інтернет.

Техніко-економічну сутність завдання: зменшити витрат часу та ресурсів, покращити загальний рівень та зробити більш зручним навчальний процес для студентів вищих навчальний закладів. Оптимізувати доступ до дорогого обладнання, надаючи можливість проводити експерименти віддалено з будь-якої частини світу. Також, це сприятиме підвищенню безпеки, оскільки користувачі не будуть безпосередньо контактувати з потенційно небезпечним обладнанням. Крім того, це забезпечує збір та аналіз даних, отриманих під час експериментів, що є критичним для наукових досліджень та освітніх цілей.

Обґрунтування доцільності рішення полягає в підвищенні ефективності наукових досліджень, скороченні часу на обробку даних та зменшенні помилок, що виникають через людський фактор. Впровадження системи Lab4All сприятиме більш раціональному використанню дослідницьких ресурсів та підвищенню якості наукових результатів.

Програмне забезпечення повинно задовольняти основні вимоги користування та його програмування:

– Безперебійна робота.

– Аналіз та виявлення дефектів з подальшим їх позбавленням.

– Зрозуміло написаний програмний код, що дасть змогу майбутнім розробникам зручно продовжити працювати та покращувати роботу коду у проекті.

– Оптимізація одного великого файлу з кодом, шляхом розбиття його на малі за певними спільними частинами.

– Наявність документації, з детально розписаної інформацією про роботу коду та його компонентів.

Розроблена система планується використовуватися в конкретних об’єктах управління, переважно до яких належать навчальні заклади. Заклади, такі як школи, коледжі та університети, що проводять лабораторні заняття та активно вимагають взаємодії з лабораторним обладнанням, особливо потребують такої

системи.

Структура програмного забезпечення складається з певних об’єктів інформаційної системи:

– «Бекенд частина».

Працюючий програмний код, який має змогу виконувати певні задані дії, для використання різних експериментів в лабораторіях до отримання та збору точних даних досліджень, фіксуванню змісту активних користувачів та загального часу виконання експериментів.

– «Серверна частина».

Найголовніша частина програмного коду - налаштування зв’язку з фізичним обладнанням, яке розташоване в лабораторіях для управління та контролю експериментів, за допомогою технології WebSocket [11].

Завдяки використанню WebSocket, програмний код може безперервно відправляти команди на обладнання та отримувати зворотні сигнали про його стан. Наприклад, якщо в лабораторії проводиться експеримент із використанням сенсорів та інших вимірювальних приладів, програмне забезпечення може оперативно отримувати дані про поточні вимірювання, аналізувати їх та відображати результати в реальному часі [12]. Одночасно, за необхідності, програма може надсилати команди для зміни параметрів експерименту або налаштування обладнання.

− «Веб частина».

Підключення бекенд частини до фронт частини, для майбутнього надання можливості взаємодії користувачу з фізичними експериментами для отримання деяких результатів певних досліджень. Бекенд відповідає за обробку запитів, управління експериментами та обробку даних, тоді як фронтенд забезпечує зручний і зрозумілий інтерфейс для користувача.

Вихідна інформація генерується або створюється в результаті отримання та обробки вхідних даних системою або програмним забезпеченням.

Вихідною інформацією системи є дані отримані під час виконання певних

експериментів, які можуть бути представлені у вигляді графіків, таблиць, звітів або інших доступних форматів. Звіти зазвичай містять текстові описи, висновки та рекомендації, що базуються на отриманих даних. Згодом така вихідна інформація буде використовуватися для аналізу результатів експериментів. Надається можливість включати порівняння даних з різних експериментів, пошук закономірностей і тенденцій, виявлення відхилень та певних аномалій.

Аналіз результатів допомагає краще зрозуміти процеси, що відбуваються, та зробити обґрунтовані висновки. Крім того, ця інформація може бути використана для інших наукових досліджень, що сприятиме розвитку відповідної галузі знань та технологій.

Вхідною інформацією для системи будуть параметри експериментів, задані користувачами, а також дані, отримані від сенсорів та інших пристроїв, під'єднаних до лабораторного обладнання. Дані повинні збиратися в реальному часі та передаватися в систему для обробки та аналізу. Система повинна забезпечувати контроль та валідацію вхідних даних, а також можливість їх коригування у разі виявлення помилок або некоректних значень.

Також під час взаємодії з експериментом можуть скластися певні умови, при яких буде порушено роботу експерименту. Причинами збоїв може бути виникнення критичних помилок, збоїв у роботі обладнання або програмного забезпечення, а також у разі втрати з'єднання з Інтернетом або недоступності сервера, на якому розміщена система. Або, на жаль, на сьогодні, досить частими причинами є відключеннями світла в Україні, що також призведе до майбутніх збоїв та припиненню роботи експериментів. Невірно введені вхідні дані можуть слугувати припиненню роботи дослідницького обладнання. Також, користувач має змогу у будь-який час припинити процес виконання досліду, шляхом натискання спеціальної кнопки в інтерфейсі проекту.

У процесі роботи з системою для управління та контролю фізичних дослідницьких експериментів функції розподіляються між користувачами та технічними засобами та забезпеченням.

До функцій користувача можна віднести:

– Налаштування та підготовка експериментального обладнання для проведення експериментів.

– Запуск та контроль процесу експерименту через клієнтські додатки або інтерфейс управління системою.

– Аналіз отриманих даних та результатів дослідження експерименту.

До функцій технічний засобів можна віднести:

– Збір експериментальних даних від підключеного обладнання в режимі реального часу.

– Передача, зберігання та обробка отриманих експериментальних даних.

– Надання доступу до експериментальних даних та результатів обробки через клієнтські додатки та веб-інтерфейси.

– Забезпечення надійності зберігання даних у відповідні документи.

Отже, можемо зробити висновок, що персонал переважно відповідає за налаштування та безпосереднє проведення експериментів, аналіз результатів та прийняття рішень, тоді як технічні засоби забезпечують автоматизацію процесів збору, обробки, зберігання та передачі експериментальних даних, а також надають необхідні інструменти та інтерфейси для взаємодії із системою.

1.5. Вимоги до інформаційної системи

1.5.1. Вимоги до функціональних характеристик

Користувач має можливість взаємодіяти з лабораторним обладнанням через веб-інтерфейс, як показано на рис.1.4. Для цього створено спеціальні поля вводу, які забезпечують взаємодію між технічним засобом та користувачем. Кожне з цих полів вводу повинно бути заповнене правильно, оскільки введені дані будуть оброблені та збережені для подальшого формування звіту про експеримент. Форми для введення даних включають різноманітні відкриті

відповіді, що дозволяє користувачеві вводити інформацію у вільному форматі. Це може бути введення особистої інформації, такої як ім'я та група, а також коментарів до звіту.

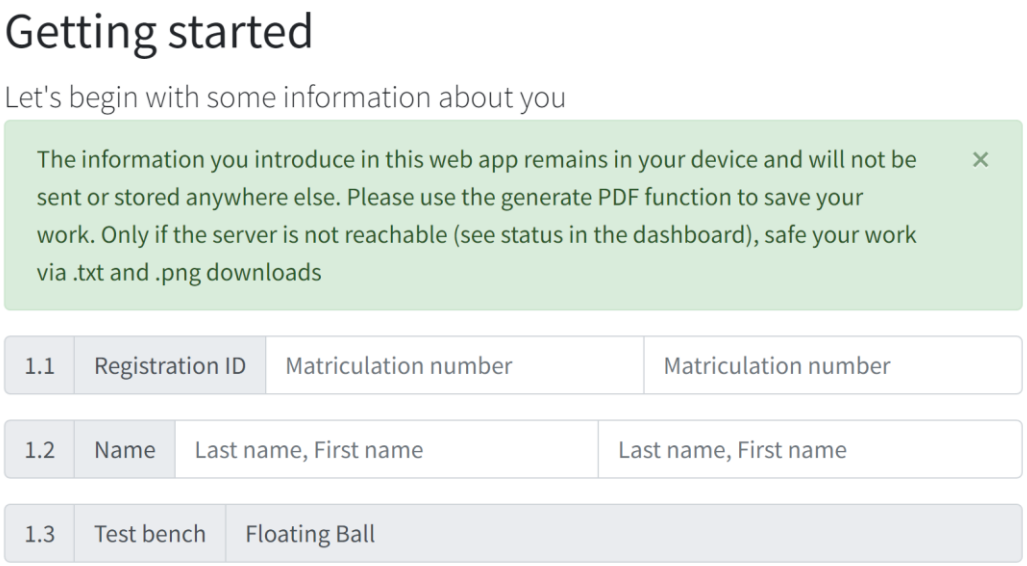

Рис.1.4. Приклад вигляду відкритих полів вводу даних

Крім цього, користувач має можливість детально вводити точні значення для кожного аспекту експерименту. Такий підхід гарантує, що всі необхідні параметри будуть враховані, а результати експерименту будуть максимально точними (рис.1.5).

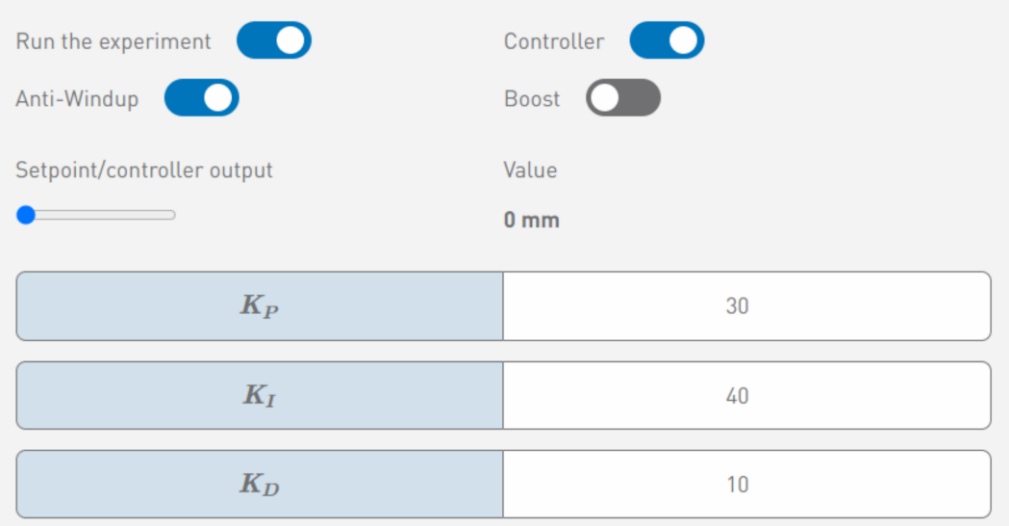

Рис.1.5. Приклад вигляду точних значень для проведення експерименту

Система також надає можливість створення графіків, які наочно демонструють процес дослідження (рис.1.6). Користувач може візуально оцінити результати на різних етапах експерименту, виявити будь-які відхилення даних, та внести необхідні корективи в процесі проведення експерименту.

Рис.1.6. Приклад вигляду роботи графіку на основі введених даних

Крім того, система автоматично формує звіти з отриманими результатами у форматі .pdf, що є зручним для подальшого аналізу та зберігання. Звіти у форматі .pdf забезпечують збереження всіх важливих даних і дозволяють легко ділитися результатами з колегами або використовувати їх у подальших дослідженнях (рис.1.7-1.8).

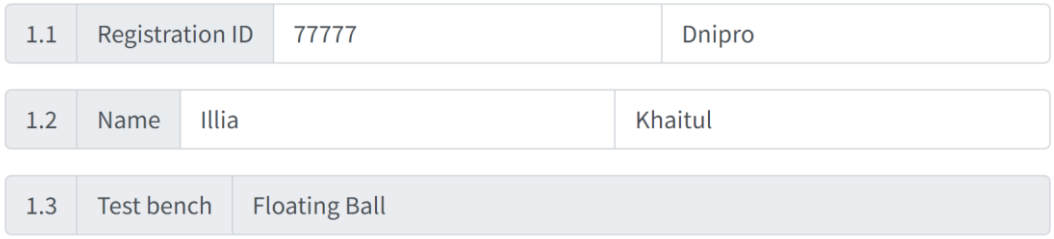

Рис.1.7. Приклад введених даних для формування звіту

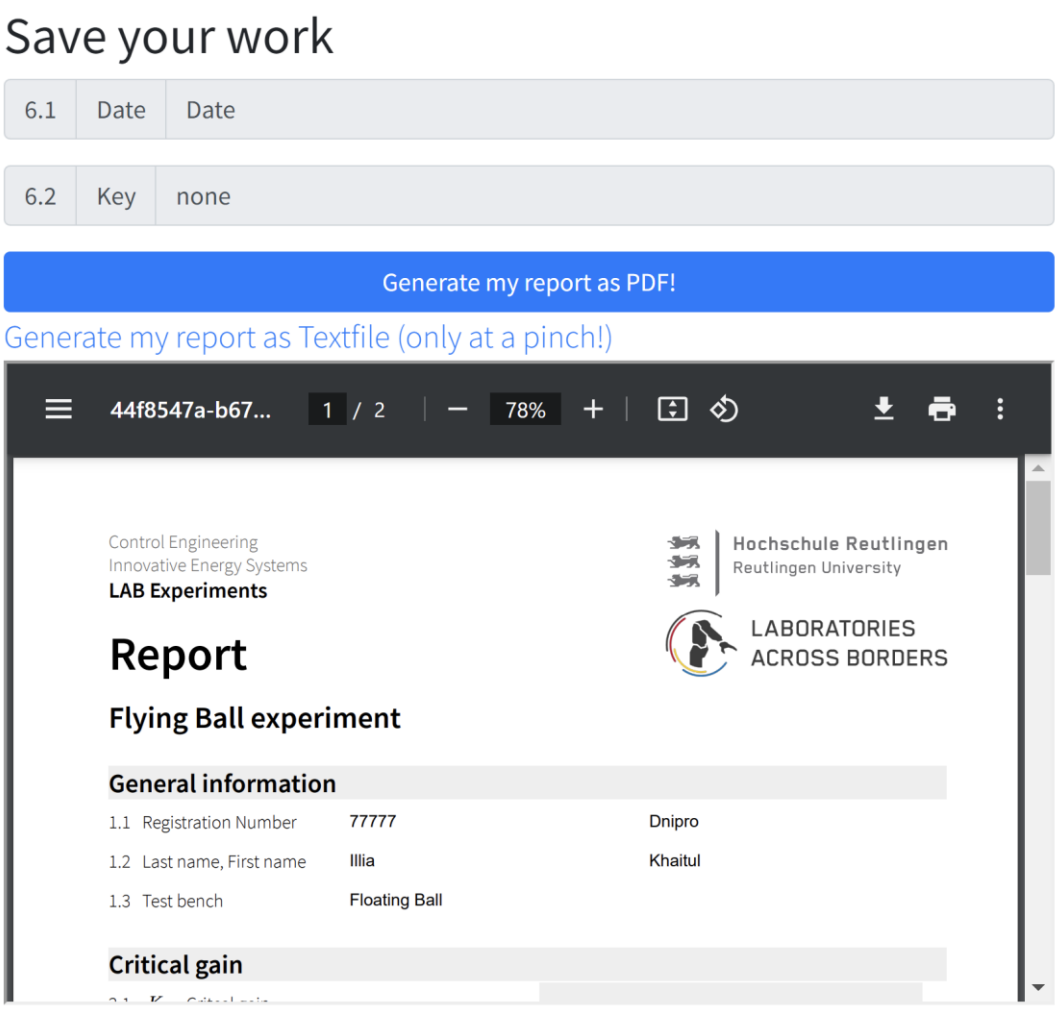

Рис.1.8. Приклад формування звіту

1.5.2. Вимоги до інформаційної безпеки

Інформаційна безпеки у своєму понятті означає забезпечення конфіденційності, цілісності та збереженні у захищеному середовищі оброблений даних та даних користувачів.

Головним принципом безпеки лабораторних обладнань Lab4All є те, що доступом для взаємодії є певний код, який може бути отриманий від адміністраторів проекту, що робить доступ більш перевіреним та захищеним, хто і коли буде намагатися авторизуватися у проекті.

Також, до вже реалізованих критеріїв безпеки проекту можна віднести:

– Вищезгаданий тризначний код для доступу до експериментів, який генерується кожні 30 секунд.

– Поп-ап інформаційне повідомлення (рис.1.9), яке зазначає певні дії, які краще не робити, аби зберегти цілісність результатів при запуску експерименту:

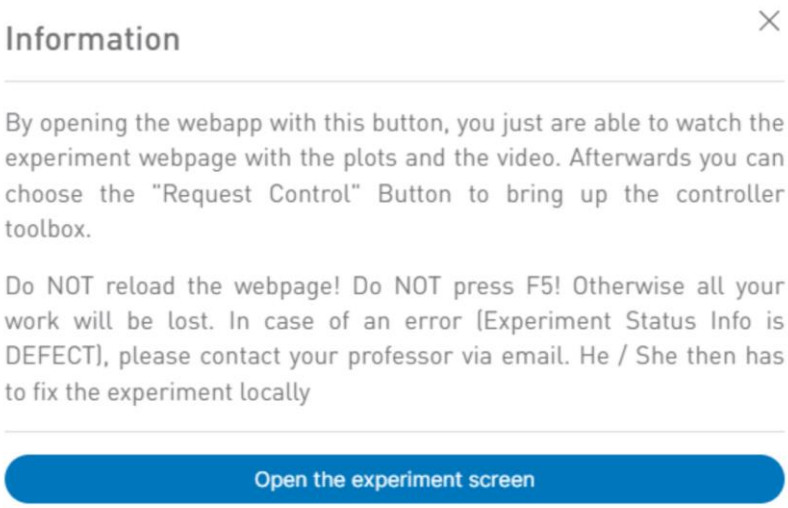

Рис.1.9. Приклад вигляду поп-ап повідомлення

– Регулярний моніторинг стану проекту, постійне покращення коду для забезпечення цілісності та інформаційної безпеки.

Такі критерії є досить важливим аспектом у формуванні більшої захищеності та безпеки використання та взаємодії з лабораторними експериментами проекту Lab4All.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для успішного та безперебійного користувальницького використання та

функціонування програмного забезпечення Lab4All не потрібно мати дуже розвинене та потужне технічне обладнання так як процес взаємодії відбувається онлайн та серверним видом, шляхом підключення через інтернет до серверної частини лабораторій в університетах та активного їх використання.

У свою чергу адміністративне обладнання в університетах потребує більш потужних технічних засобів для надання ефективного та безпомилкового доступу до дослідницьких експериментів.

Також, можна роздивитися більш детально певні технічні вимоги кожної із сторін.

Технічні вимоги для користувачів лабораторного обладнання у рамках проекту:

– Веб-браузер з підтримкою HTML5, CSS3 та JavaScript. Google Chrome, Mozilla Firefox, Microsoft Edge, Safari (для macOS).

– Стабільне інтернет-з'єднання. Високошвидкісне з'єднання для безперебійного доступу до лабораторій.

– Вимоги до обладнання. Комп'ютер з достатньою обчислювальною потужністю та пам'яттю.

– Комп’ютерна периферія для взаємодії з комп’ютером. Клавіатура, комп’ютерна мишка та монітор.

Технічні вимоги для серверної частини лабораторного обладнання у рамках проекту:

– Потужні сервери з достатніми обчислювальними ресурсами та місткістю пам'яті для обробки даних експериментів та підтримки великої кількості користувачів.

– Високошвидкісне підключення до Інтернету з високою пропускною здатністю для забезпечення стабільної передачі даних.

– Система управління базами даних для зберігання та обробки експериментальних даних, налаштувань користувачів та іншої інформації.

– Платформа для розгортання веб-додатку.

– Засоби моніторингу та аналізу продуктивності системи, а також інструменти для резервного копіювання та відновлення даних.

– Надійні заходи безпеки, такі як брандмауер, система виявлення вторгнень (IDS/IPS), шифрування даних та захист від DDoS-атак.

– Кваліфікований персонал для адміністрування серверів, баз даних, веб-додатку та забезпечення безперервної роботи системи.

Наведені технічні вимоги забезпечать користувачам проекту Lab4All зручний та безперебійний доступ до веб-додатку, а адміністраторам серверної частини - надійну та безпечну інфраструктуру для розміщення та обслуговування веб-додатку та його даних.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для постійно та успішної роботи експериментів проекту необхідно, щоб програмне забезпечення комп'ютера, на якому буде працювати веб орієнтована підсистема, відповідало наступним критеріям інформаційної структури сумісності:

– «Коректна взаємодія з браузерами». Для правильного відображення графічної частина та успішного запуску програмної частини.

– ОС-Windows, будь-якої версії, бажано починаючи з 10-ої.

– «Протоколи зв’язку». Підтримка стандартних протоколів, таких як HTTPS, для забезпечення безпечної передачі даних.

– «Захист даних». Шифрування даних користувачів та використання SSL-сертифікатів для захисту інформації.

Та певним вимогам з боку програмної сумісності:

– «Сумісність з ОС». Підтримка основних операційних систем (Windows, macOS, Linux).

– «Серверна частина». Node.js використовується для серверної частини, забезпечуючи виконання JavaScript на сервері та взаємодію з базами даних та

іншими сервісами.

– «Документація». Всі вихідні коди повинні бути детально документовані для забезпечення зрозумілості та легкості підтримки. Це включає коментарі в коді, а також окрему документацію для кожного модуля та компонента системи.

– «Контроль версій». Використання системи контролю версій (наприклад, Git) для відстеження змін у вихідних кодах.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Розроблене програмне забезпечення, у рамках проекту Lab4All призначене для навчальних цілей та виконання наукових досліджень. Функціональне призначення інформаційної системи додатку складається з певних пунктів:

– «Пряма інтеграція з експериментами»: Розроблена система дає можливість безперебійної взаємодії з доступними експериментами та їх успішного використання у певних навчальних цілях.

– «Визначення параметрів експерименту»: Детальне редагування певних фізичних значень, які повністю впливають на підсумковий результат досліджуваного експерименту.

– «Ведення бази даних експериментів»: Дозволяється створювати, редагувати та зберігати записи та отримані дані про всі проведені експерименти.

– «Генерація звітів»: Додана можливість автоматично генерувати звіти, включаючи графіки, таблиці та текстові висновки на основі проведених досліджень та експериментів.

– «Контроль доступу»: Реалізовано функції контролю доступу до експериментів та його функцій, завдяки впровадженню тризначного оновлюємого коду, до якого мають доступ тільки адміністратори системи.

– «Моніторинг та контроль»: Також розроблювальна система дає можливість відстежувати процес експерименту, контролювати умови його проведення та вносити необхідні корективи в режимі реального часу.

До експлуатаційного призначення системи можна віднести декілька пунктів, які надають певні переваги для замовника:

– «Оптимізація»: Це дозволяє зменшити час на планування та проведення експериментів, за рахунок автоматизації більшості процесів, що також, безумовно, призводить до зменшення ймовірності помилок та оптимізації

використання ресурсів.

– «Покращення якості»: Забезпечення більш точних вимірювань та збору даних. Також система зберігає повну історію експериментів, включаючи всі параметри та налаштування, що надає можливість згодом точно відтворити умови минулих дослідження та порівняти їх з більш новими.

Отже, можна зробити певний висновок, експлуатаційне призначення полягає в підвищенні продуктивності та вдосконалені наукових досліджень, гарантуванні точності та повторюваності експериментів, а також у досить серйозному збереженні ресурсів, часу або фінансів та зусиль завдяки автоматизації та вдосконаленню робочих процесів.

2.2. Опис застосованих математичних методів

Через специфіку галузі розробки, яка не вимагає застосування математичних методів, при процесі рефакторингу та розробки програмного забезпечення для управління та контролю дослідницьких експериментів математичні методи не застосовувалися.

2.3. Опис використаних технологій та мов програмування

Для виконання розробки та рефакторингу програмного забезпечення, було використано наступні технології та мови програмування:

– JavaScript. Основна мова програмування, що використовується при розробці [13].

– Node.js. Середовище виконання JavaScript, що активно використовувалося для створення веб-сокетного сервера та обробки певних задач на стороні сервера [14].

– «WebSockets». Протокол двосторонньої комунікації, між каналами - клієнтом і сервером.

Модулі Node.js [15].

– «HTTPS». Використовується для створення безпечного «HTTPS-сервера», який забезпечує захищене з'єднання між клієнтом і сервером [18].

– «FS». Використовується для взаємодії та роботи з файловою системою, забезпечуючи можливість читання, запису, видалення та інші операції з файлами [16].

– «WS». Використовується для роботи з веб-сокетами [17].

2.3.1. Детальний опис мови програмування

Мова програмування JavaScript

JavaScript - це мова програмування, яка спочатку була розроблена для створення інтерактивних веб-сторінок. Вона була створена Бренданом Айком у 1995 році для компанії Netscape з метою додавання інтерактивності до веб-сторінок. Спочатку JavaScript був призначений для написання логіки на стороні клієнта, але пізніше, з появою середовища Node.js, його стали використовувати і для створення серверних додатків [19].

Ключовою особливістю JavaScript є його здатність працювати як на стороні клієнта, так і на стороні сервера. На клієнтській стороні JavaScript використовується для динамічного оновлення контенту веб-сторінок, обробки подій користувача, тощо. На серверній стороні, завдяки середовищу Node.js, JavaScript дозволяє розробникам писати певний код, який може обробляти великі обсяги запитів, забезпечуючи швидку та ефективну роботу серверів [20].

Також, JavaScript має вбудовані механізми для роботи з асинхронними операціями. Асинхронне програмування дозволяє виконувати кілька завдань одночасно без блокування головного процесу виконання програми. Певні операції можуть бути виконані у фоновому режимі, а результат цих операцій може бути оброблений пізніше, коли він стане доступним. Наприклад, завантажувати дані з сервера, не блокуючи інтерфейс користувача, що

забезпечує більш плавний та швидкий досвід взаємодії з додатком. Це особливо важливо для веб-додатків, які потребують швидкої реакції на дії користувача та обробки даних у реальному часі [21].

Крім того, наразі JavaScript має величезну і активну спільноту розробників, а також безліч відкритих бібліотек та фреймворків для різноманітних цілей, що надає можливість комунікування з досвідченими людьми на різних тематичних платформах для вирішення персональних питань або спільних проблем. Також, це надає доступ до готових рішень та підказок, що прискорює розробку та полегшує використання даної мови програмування.

Використання JavaScript як на клієнтській, так і на серверній частинах дозволяє розробникам використовувати єдину мову програмування для всього додатку. Це сприяє спрощенню процесу розробки та дозволяє легше підтримувати код. Для розробки Lab4All це означає, що всі компоненти системи, від клієнтського інтерфейсу до серверної логіки, можуть бути реалізовані на JavaScript.

2.3.2. Детальний опис технологій

Середовище Node.js

Node.js - це середовище виконання JavaScript, яке дозволяє використовувати та успішно запускати код JavaScript на сервері. Воно зроблене на високопродуктивному двигуні V8 від Google, який компілює JavaScript в машинний код, що забезпечує швидке виконання заданих процесів. Node.js був створений Райаном Далем у 2009 році і з того часу став одним із найпопулярніших інструментів для серверної розробки завдяки своїй ефективності та гнучкості [22].

Також, середовище надає розробникам можливість використовувати JavaScript для написання серверних додатків, таких як веб-сервери, мережеві утиліти, файлові системи, «middleware» та інше. Воно має вбудовану бібліотеку

модулів, яка забезпечує схвальну функціональність для роботи з файловою системою, мережевими сокетами та потоками великої кількості даних.

Node.js використовує подієво-орієнтовану архітектуру, що дозволяє обробляти велику кількість одночасних з'єднань при цьому з мінімальними витратами ресурсів [23]. Ці фактори надають можливість йому бути ідеальним для розробки масштабованих мережевих додатків, таких як веб-сервери, API та інші серверні додатки, що потребують високої продуктивності [35].

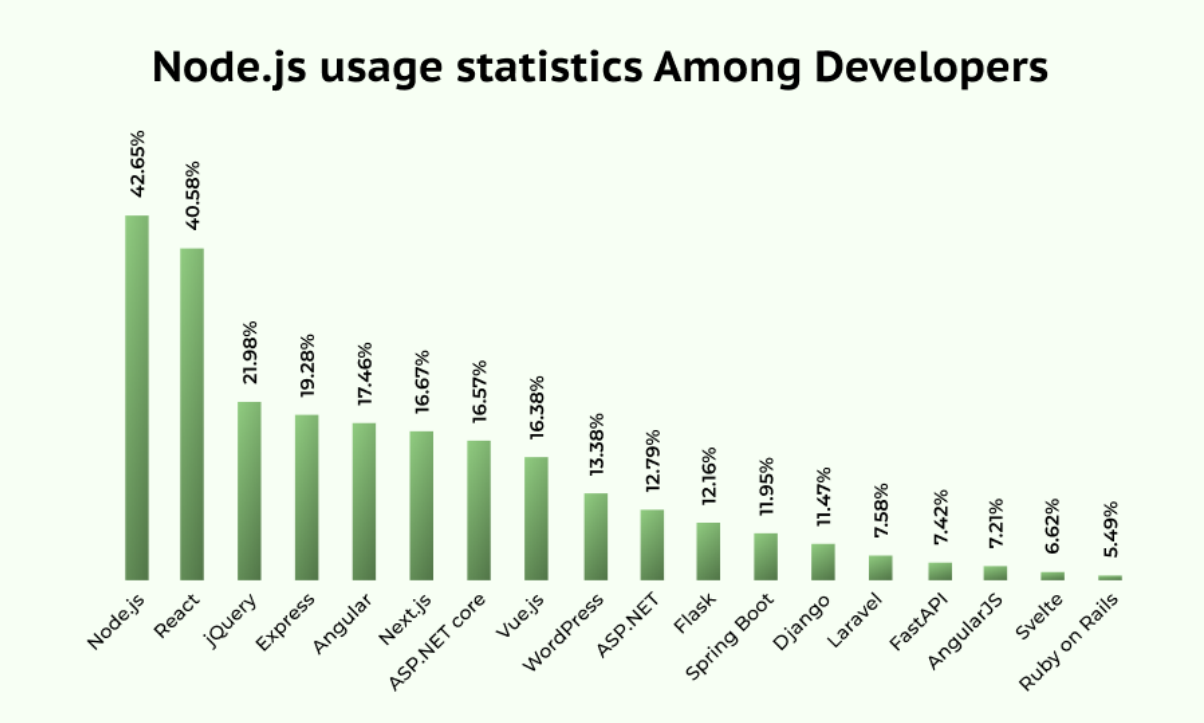

Рис.2.1. Статистичний приклад популярності використання Node.js

Важливим фактором використання Node.js є впровадження високої продуктивність завдяки використанню неблокуючої «Вводу/Виводу моделі», яка дозволяє виконувати операції вводу/виводу асинхронно, незважаючи один одному. Це означає, що сервер може обробляти інші запити під час очікування завершення операцій вводу/виводу, що значно підвищує продуктивність.

Архітектура Node.js дозволяє створювати додатки, які можуть реагувати на події у реальному часі. Вона забезпечує можливість миттєво реагувати на зміну стану експериментів та взаємодію з користувачами.

Також, важливо зазначити, що Node.js під час розробки у проекті Lab4All надає ефективне управління WebSocket-з’єднаннями. Node.js ефективно керує кількома WebSocket-з'єднаннями, забезпечуючи зв'язокок у реальному часі між внутрішніми експериментами та зовнішніми клієнтами. Скрипт обробляє кілька експериментів і клієнтів, використовуючи асинхронні можливості Node.js для підтримання стабільної роботи без блокування процесів. Node.js також керує таймерами та управлінням чергами, забезпечуючи правильне обслуговування клієнтів у реальному часі.

Він містить в собі дуже розвинену екосистему та наявність численних бібліотек, які значно допомагають у розробці. Також, можна привести як приклад, під час розробки даного проекту, були використані такі бібліотеки, як «ws» для роботи з WebSocket, «fs» для роботи з файловою системою та «https» для створення захищених з'єднань. Система дозволяє швидко інтегрувати необхідні функціональні можливості без потреби писати їх з нуля.

Протокол зв’язку WebSocket

WebSocket є протоколом зв'язку, який надає направлений зв'язок між сервером і клієнтом через «TCP-з'єднання» [24].

«TCP» (Transmission Control Protocol) є одним із головний протоколів, які використовують для передавання даних у межах Інтернету. Це спосіб, за допомогою якого два комп'ютери можуть спілкуватися один з одним через Інтернет [25]. Прикладом до роботи системи є телефонний дзвінок: коли ви телефонуєте другові, спочатку встановлюється з'єднання, потім ви обмінюєтеся інформацією, тобто просто розмовляєте, і в кінці з'єднання завершується.

WebSocket був стандартизований у 2011 році як частина HTML5 і забезпечує значні переваги в порівнянні з традиційними HTTP-запитами [26]. Він створює постійне з'єднання між сервером і клієнтом, дозволяючи їм обмінюватися даними в будь-який момент. Це відрізняється від HTTP, де клієнт завжди повинен ініціювати запит до сервера. Завдяки постійному з'єднанню WebSocket зменшує затримку, оскільки не потрібно кожного разу встановлювати

нове з'єднання для передачі даних.

Важливо вказати, що у межах розробки ПЗ важливо забезпечити спілкування між експериментами та клієнтами у реальному часі. WebSocket дозволяє серверу негайно відправляти оновлення клієнтам, тому він як найкраще підходить для цієї задачі. Постійне з'єднання зменшує час та інші витрати на встановлення нових з'єднань і дозволяє швидше обмінюватися повідомленнями, що критично важливо для нашого проекту, який потребує швидкої реакції на події. WebSocket значно зменшує кількість мережевого трафіку завдяки ефективному використанню заголовків у порівнянні з HTTP. Він дозволяє обробляти більше з'єднань одночасно і зменшує навантаження на сервер.

Дуже важливо забезпечувати найкращу синхронізацію даних між сервером і клієнтами для задач, де потрібно, щоб всі клієнти мали актуальні дані в реальному часі. WebSocket також дозволяє зручно керувати сесіями клієнтів, підтримуючи постійне з'єднання та легко ідентифікуючи кожного клієнта, що адміністрування та моніторинг активних сесій.

Також, під час розробки активно використовуються асинхронні можливості WebSocket для управління кількома з'єднаннями одночасно, забезпечуючи високу продуктивність та швидкий відгук на події. Завдяки WebSocket сервер може ефективно обробляти численні підключення, зменшуючи навантаження на мережу та серверні ресурси.

Модулі Node.js

Модулі Node.js є основною частиною екосистеми Node.js і відіграють дуже важливу роль у розширенні функціональності та спрощенні розробки додатків, і також є одним із ключових причин поширенності Node.js. Вони дозволяють розробникам організовувати свій код у відповідні блоки коду, які можуть бути легко використані в різних проектах. При цьому вони є окремими одиницями коду, які можна підключати до програми за потребою.Від початку своєї появи, Node.js підтримував модульну структуру для організації коду, що дозволило розробникам ділити свій код на менші, більш контрольовані частини.

Модулі дозволяють програмістам ділити код. Замість того щоб писати весь код в одному файлі, можна розділити його на декілька модулів. Кожен модуль відповідає за певну функціональність, що полегшує читання, розуміння та тестування коду. Вони також дозволяють повторно використовувати код у різних проектах без необхідності його копіювати.

Модуль «ws» використовується для створення та управління WebSocket-з'єднаннями, що забезпечує двоканальний зв'язок між сервером і клієнтами, що є критично важливим для нашого проекту Lab4All. Він надає можливість створити WebSocket-сервер, який вміє обробляти багато з'єднань одночасно. Сервер може приймати підключення від клієнтів, отримувати повідомлення від них і надсилати певні відповіді. Це досягається через простий та інтуїтивно зрозумілий API, який дозволяє швидко налаштувати сервер та обробляти події, такі як підключення, повідомлення і відключення клієнтів [36]. Модуль забезпечує можливість роботи з бінарними даними, що дозволяє передавати не тільки текстові, але й будь-які інші типи даних між клієнтом та системою. WebSocket забезпечує постійне з'єднання, що зменшує витрати ресурсів на створення нових зв'язків і дозволяє швидше обмінюватися повідомленнями.

Модуль «fs» використовується для роботи з файловою системою, що дозволяє читати та записувати файли, корисно для зберігання налаштувань, даних і сертифікатів. Це один з найважливіших модулів для роботи з файловою системою у Node.js [38]. Модуль надає можливість використовувати різні функції для роботи з файлами. Можливість читати файли, використовуючи функції "fs.readFile" або "fs.readFileSync", що дозволяє завантажувати вміст файлу у змінні. Він є особливо корисним для зберігання і читання налаштувань даних.

Для запису даних у файл використовуються функції "fs.writeFile" та "fs.writeFileSync", вони дозволяють створювати нові файли або змінювати вже існуючі. Модуль fs також надає функції для роботи з каталогами, створювати нові каталоги, видаляти каталоги або читати його вміст, що надає розробникам

можливість легко управляти структурою файлів і каталогів у додатку. Модуль «fs» також може бути використаний для зберігання конфігураційних файлів, їх параметрів, логів і інших даних, які потрібні для роботи повноцінної та стабільної роботи програмного забезпечення [37]. Підсумувавши, можна зробити висновок, що модуль «fs» дозволяє ефективно управляти файлами і каталогами, забезпечуючи високу продуктивність і надійність при роботі з файловою системою.

Модуль «https» використовується для впровадження безпечних HTTPS-серверів, що дозволяє захистити дані від перехоплення та забезпечує стабільність з'єднання між сервером та клієнтською базою. Використання модуля «https» у проекті забезпечує кілька досить важливих переваг. По-перше, серйозне шифрування даних гарантує, що інформація, що передається між клієнтами і серверами, захищена від перехоплення або модифікації зловмисниками та вірусами. По-друге, використання сертифікатів дозволяє клієнтам переконатися в достовірності та валідності сервера, з яким вони спілкуються. Це особливо важливо для додатків, які обробляють приватну та конфіденційну інформацію, таку як особисті дані користувачів або фінансовий звіт компанії. У підсумку, модуль Node.js «https» є потужним інструментом для створення та впровадження безпечності та захищеності у веб-додатках, забезпечуючи захист даних і автентифікацію сервера. Він є незамінним для додатків, що вимагають високого рівня безпеки та конфіденційності.

Використання модулів під час роботи над проектом дозволяє розділити логіку ПО на окремі частини, покращуючи організацію та зрозумілість написаного коду. Крім того, модулі дозволяють зменшити ризик або зовсім уникнути дублювання коду, повторно використовуючи функції та константи з різних модулів. Також модулі забезпечують легку інтеграцію стандартних та додаткових функціональностей Node.js, таких як робота з файловою системою, веб-сокетами та HTTP-серверами. Нарешті, вони покращують підтримку та розширюваність додатку, оскільки модулі можна замінювати або оновлювати

окремо. Загалом, модулі Node.js забезпечують модульність, повторне використання коду, зменшення залежностей та доступ до величезної екосистеми пакетів, відіграючи ключову роль у забезпеченні необхідної функціональності, організації та підтримки додатку, що працює з веб-сокетами та керує експериментами.

2.4. Опис структури системи та алгоритмів її функціонування

2.4.1. Опис логічної структури системи

Lab4All являє собою динамічний та інтерактивний веб-застосунок. Проект складається з функціональної веб-сторінки та головного коду системи, який стоїть за веб-сторінкою та всім функціоналом. Система складається з кількох основних компонентів, які взаємодіють між собою для забезпечення повного функціоналу.

Логічна структура системи складається з певних компонентів сайту:

- Головна сторінка.
- Авторизація.
- Теоретична інформація.
- Контролер.
- Графік експерименту.
- Збереження результатів у .pdf документ.

Головна сторінка веб-сайту є початковою точкою для користувачів та учнів проекту. Вона надає вступну інформацію про експеримент, його мету та цілі. Тут також розміщені посилання на інші розділи сайту, такі як «Теоретична інформація» про використання експерименту, його фізичних формул, математичних структур, тощо, «Контролер», детальне налаштування важливих параметрів певного експерименту, і «Збереження підсумків дослідів та роботи з експериментом». Основні функції головної сторінки включають привітання

користувачів, можливість авторизації через трьохзначний код, який можна отримати від адміністрації проекту, надання короткого огляду експерименту, забезпечення швидкого доступу до інших розділів сайту та можливість вводу своєї персональної інформації, ім’я та студентського ідентифікаційного коду.

Автоматизація через код. Для доступу до експерименту необхідно пройти авторизацію, ввівши спеціальний користувацький код (рис.2.2-2.3). Для уникнення зломів, код оновлюється кожні 30 секунд. Після цього надається повний доступ до всіх інструментів проекту.

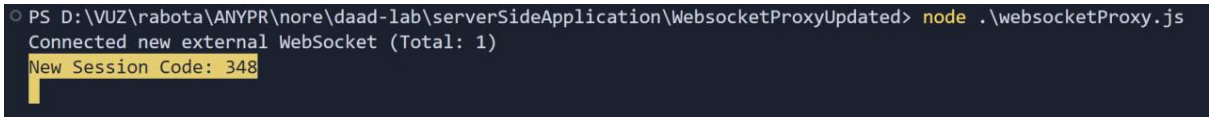

Рис.2.2. Отримання коду автоматизації через консоль

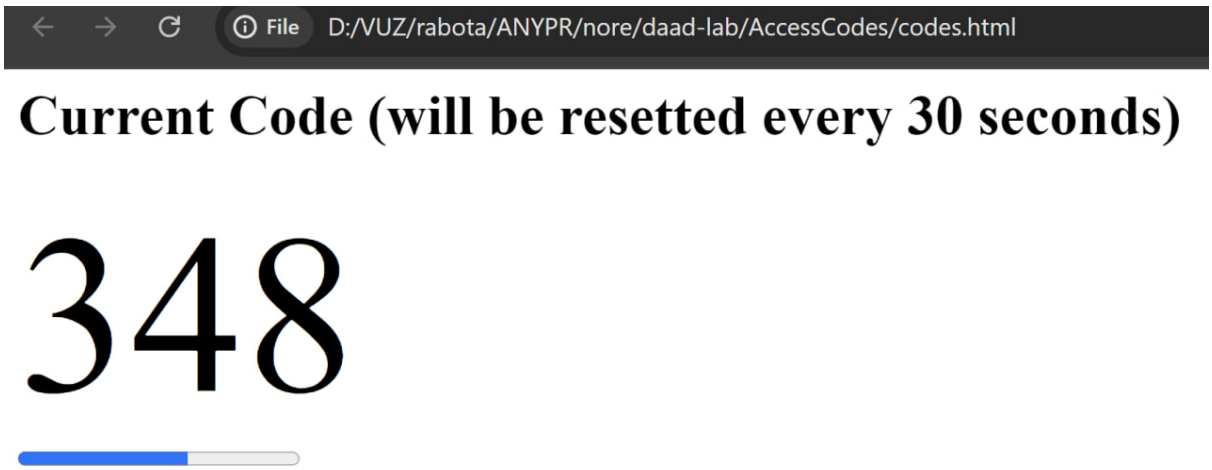

Рис.2.3. Отримання коду автоматизації через керувальний веб сайт

Теоретична сторінка є ключовим ресурсом для користувачів, надаючи їм всю необхідну інформацію для розуміння і проведення експериментів. Тут детально розписані пояснення теоретичних концепцій, методологій та принципів, що лежать в основі досліджень. Користувачі мають змогу отримати глибше уявлення про досліджувані явища, що дуже важливо для точного і надійного виконання експериментів. Сторінка тісно взаємопов’язана з

контролером, і користувачі можуть ознайомитися з теоретичною інформацією експериментів перед тим, як перейти до налаштування параметрів і запуску експериментів у контролері.

Контролер є центральним компонентом системи, де користувачі можуть детально налаштовувати фізичні значення експерименту. Також, тут представлено три основні типи контролерів: пропорційний «P-контролер», інтегральний «I-контролер» та диференціальний «D-контролер».

– «P-контролер» швидко реагує на відхилення від заданої траєкторії, пропорційно змінюючи вихідний сигнал, що дозволяє миттєво реагувати на зміни, але занадто великий коефіцієнт посилення може призвести до нестабільності в роботі системи.

– «I-контролер» працює за принципом накопичення (інтеграції) відхилення за часом, що дозволяє зменшити постійне відхилення.

– «D-контролер» оцінює швидкість зміни відхилення, що дозволяє швидко реагувати на зміни, зменшувати коливання і допомагає стабілізувати систему при раптових змінах.

На сайті можна змінювати параметри цих контролерів і спостерігати, як це впливає на результати експерименту. Це чудовий спосіб на практиці зрозуміти, як різні типи контролерів працюють у реальних умовах і які особливості вони мають. Також, контролер тісно взаємодіє з графіком експерименту, який відображає дані в режимі реального часу під час проведення експериментів.

Графік експерименту працює у тісній зв’язці з контролером, надаючи візуалізацію отриманих даних у реальному часі. Графік допомагає користувачам аналізувати хід експерименту та зробити необхідні коригування або висновки з отриманих результатів.

Збереження результатів у документи забезпечує можливість створення звітів на основі даних, зібраних під час експериментів та інформації персональної авторизації. Цей компонент взаємодіє як з контролером, так і з графіком експерименту, отримуючи від них необхідні дані для формування

повноцінних звітів. Збережені документи можуть бути використані для подальшого аналізу, звітування або навчання.

Таким чином, всі основні компоненти системи тісно взаємопов’язані між собою: головна сторінка забезпечує навігацію, авторизація надає доступ, теоретична сторінка містить необхідну інформацію, контролер керує експериментами, графік експерименту візуалізує дані, а збереження результатів дозволяє документувати результати дослідів.

Проект також складається і з бекенд частини, над якою, велася моя головна оптимізація, рефаткоринг та доповнення розробки. Система управляє кількома експериментами та забезпечує належну обробку станів WebSocket, управління сесіями та взаємодію з клієнтами.

Основні компоненти:

– Внутрішній WebSocket сервер «wssInt»: Обробляє внутрішні з'єднання експериментів.

– Зовнішній WebSocket сервер «wssExt»: Обробляє зовнішні з'єднання експериментів.

Експерименти:

– Експеримент з літаючою кулею «expFb»: Управляє взаємодією з клієнтами, управлінням сесіями та контролем для експерименту «FlyingBall».

– Експеримент з ліфтом «expE»: Обробляє запити на управління, управління сесіями та оновлення статусу для експерименту «Elevator».

Внутрішній WebSocket сервер «wssInt».

Сервер призначений для обробки з'єднань від експериментів, які працюють у внутрішній мережі. Він працює на порту «8081» і забезпечує зв'язок між внутрішніми експериментами та сервером. Основні завдання цього сервера включають керування з'єднаннями, обробку вхідних повідомлень і моніторинг стану експериментів. Коли до системи підключається новий експеримент, якщо вже існує активне з'єднання з цим експериментом, попереднє з'єднання автоматично закривається. Всі отримані від експериментів повідомлення

обробляються і можуть бути переслані на зовнішній WebSocket сервер або використані для керування експериментами іншими способами. Крім цього, внутрішній сервер постійно відстежує стан експериментів і відповідно оновлює їхні статуси, щоб забезпечити правильну роботу всієї цілісної системи.

Розглянемо більш детально створення та роботу внутрішнього серверу на прикладі програмного коду:

```
const wssInt = new WebSocket.Server({server: serverInt});
wssInt.on('connection', function (ws, req) {
   // Обробка з'єднання для експерименту з ліфтом
   experimentE = ws;
   console.log('Встановлено з\'єднання з експериментом E');
   // Відстеження повідомлень від експерименту
   ws.on('message', function (msg) {
      // Перевірка стану експерименту
      checkStatusE(msg);
      // Відправка даних клієнтам
      SendProxyDataE();
   });
   ws.on('close', function () {
      console.log("Відключено WebSocket з експериментом E");
   });
});
```

Також, невелике пояснення наданого коду:

Цей код створює внутрішній WebSocket сервер «wssInt», який обробляє з'єднання з експериментами. Коли встановлюється з'єднання, воно зберігається у змінній "experimentE", яка надає активне з'єднання з експериментом «Elevator».

Після чого в консолі відображається повідомлення про встановлення нового з'єднання. Функція "checkStatusE" обробляє повідомлення, що надійшло, для перевірки і оновлення стану експерименту. Наприклад, ця функція може

визначати, чи експеримент працює нормально, або, наприклад, виникли якісь помилки. Після перевірки стану експерименту, функція "SendProxyDataE" відправляє оновлені дані всім підключеним клієнтам, які відстежують цей експеримент, наприклад, дані про поточний стан експерименту або виміряні значення експерименту.

Зовнішній WebSocket сервер «wssExt».

Зовнішній WebSocket сервер відповідає за керування підключеннями від зовнішніх клієнтів. До основних функцій цього сервера включають прийом підключень від клієнтів, які бажають взаємодіяти з експериментами, обробку їхніх повідомлень та перевірку доступу за допомогою сесійних кодів авторизації.

Повідомлення від клієнтів можуть бути перенаправлені до експериментів або використані для керування станом клієнтів, наприклад, додавання їх до черги очікування. Сервер також ретельно контролює доступ клієнтів до експериментів, використовуючи для цього сесійні коди, які кожні 30 секунд оновлюються для підтримки безпеки та коректної роботи системи.

Розглянемо більш детально, на прикладі програмного коду:

```
const wssExt = new WebSocket.Server({server: serverExt});
wssExt.on('connection', function(ws, req) {
   connections++;
   console.log("Підключено новий зовнішній WebSocket (Всього: " + connections
+ ")");
   if(req.url == "/clientExpFb") {
      clientsExpFb.push(ws);
      ws.state = OPEN;
      ws.on('message', function(msg){
         var msgObj = JSON.parse(msg);
         switch(this.state) {
            case OPEN:
               // Обробка запиту на контроль експерименту
```

```
            if((msgObj.SessionCode == sessionCode) ||
isUidValid(msgObj.SessionCode)) {
                if(clientExpFbMaster != null) {
                    clientsMasterQueueExpFb.push(ws);
                    this.state = QUEUE;
                    this.send(`{"CONTROL":"queue_entered"}`);
                } else {
                    this.state = CONTROL;
                    clientExpFbMaster = this;
                    this.send(`{"CONTROL":"true"}`);
                    TimerFb();
                }
            } else {
                this.send(`{"CONTROL":"false"}`);
            }
            break;
        case CONTROL:
            // Пересилання повідомлення до експерименту
            experimentFb.send(msg);
            break;
    }
});
ws.on('close', function() {
    var idx = clientsExpFb.indexOf(this);
    if (clientExpFbMaster == this) {
        QuitMasterFb();
    }
    clientsExpFb.splice(idx, 1);
    connections--;
```

```
            console.log("Відключено WebSocket Fb (Всього: " + connections + ")");
        });
    }
});
```

Також, надам детальне пояснення до приведеного коду:

Код створює новий зовнішній «WebSocket сервер», використовуючи об'єкт "serverExt" як базовий «HTTP сервер». Коли зовнішній «WebSocket»сервер отримує нове з'єднання, змінна "connections" збільшується на 1, щоб відобразити нове з'єднання. В консолі виводиться повідомлення з загальним числом з'єднань.

Якщо URL запиту відповідає "clientExpFb", тобто відбувається з’єднання з експериментом «FlyingBall», нове з'єднання додається до масиву "clientsExpFb", який містить усіх клієнтів, підключених до експерименту. З'єднанню присвоюється початковий стан «OPEN». Коли клієнт відправляє повідомлення через «WebSocket», повідомлення розбирається з формату JSON і залежно від поточного стану з'єднання обробляється відповідним чином. Якщо стан з'єднання «OPEN», клієнт намагається отримати контроль над експериментом. Якщо «SessionCode» в повідомленні збігається з поточним кодом сесії, то відбувається перевірка, чи вже є активний майстер "clientExpFbMaster". Якщо майстер є, новий клієнт додається до черги "clientsMasterQueueExpFb", і його стан змінюється на «QUEUE», про що клієнту надсилається повідомлення. Якщо ж майстра немає, цей клієнт стає майстром "clientExpFbMaster", його стан змінюється на «CONTROL», і клієнту надсилається відповідне повідомлення, і надається доступ до експерименту. Також запускається таймер «TimerFb()», який відображає час використання експерименту.

2.4.2. Опис алгоритмічної структури системи

Для повноцінної взаємодії користувача з сайтов експериментів Lab4All, треба розуміти логічний алгоритм дій, виконання якого приведе користувача від

початку роботу, авторизації та налаштування фізичних параметрів експерименту до його повноцінного запуску та отримання певних результатів.

Отже, коли користувач переходить на офіційний сайт Lab4All для використання доступних фізичних експериментів, він спочатку потрапляє на головну сторінку цього веб-додатку.

Першим кроком для користувача є зустрів з поп-ап сторінкою (рис.1.9), після цього починається повне ознайомлення з інтерфейсом сайту, де можна побачити пункт авторизації з введенням коду, головний графік з осями та таблиця із заповненням персональних даних.

Після першого заходу на сайт, користувач повинен увійти в систему, за допомогою трьохзначного коду, який можна отримати від адміністратора проекту. Після авторизації, користувач отримує повний доступ до функціоналу та можливостей веб-застосунку.

Зліва можна побачити список доступних сторінок сайту. За порядком, можна почати зі сторінки «Critical Gain». На сторінці «Critical Gain» користувач має змогу ознайомитися з теоретичними відомостями та інструкціями по використанню експериментів веб-додатку.

Після чого користувач може перейти на наступну сторінку «Empirical Method» де також можна побачити теоретичну інформації та використані формули для повноцінного проведення експерименту.

Далі йде сторінка «Controller», де користувач також отримує запас інформації на рахунок працювання експериментів, і, також, тепер він може ввести необхідні фізичні параметри для налаштування експерименту, а саме параметри: Пропорційне посилення, Інтегральне посилення та Похідний приріст.

Наступною йде сторінка «Anti-Windup», яка служить для тестування та примітки для запобігання збоїв при використанні дослідницького майна.

Під графіком можна побачити основні параметри встановлення значень для тестування, запуску експерименту та надання можливості включення або виключення введених параметрів, із сторінок «Controller» та «Anti-Windup».

Також тут є пункт швидкості польоту кульки експерименту.

Після налаштування потрібних параметрів, користувач має змогу запустити експеримент, натиснувши вимикач "Run Experiment" та побачити певні отримані результати, які будуть відображені на графіку.

Також, останнім кроком користувача є занотування своїх результатів у сторінці «Save your work». Де можна побачити сталу дату проведення експерименту, ідентифікаційний ключ та кнопку "Generate my report as PDF!", що перекладається як "Згенерувати мій звіт у форматі PDF!". Після натискання на кнопку генерації, буде сформовано PDF документ, з усією заповненою та отриманою інформацією під час проведення експерименту.

Також, для точнішого розуміння структури алгоритму, мною було створено алгоритмічну діаграму веб-сторінки проекту «Lab4All» (рис.2.4) та її головних компонентів:

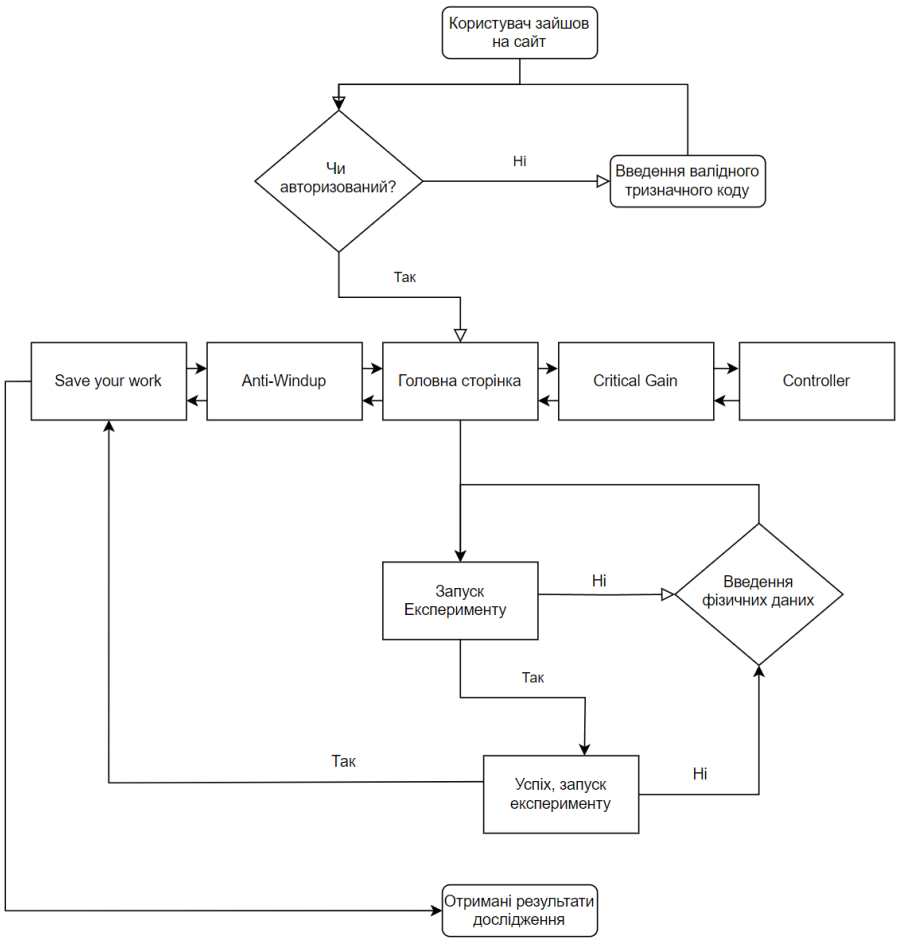

Рис.2.4. Алгоритм взаємодії користувача з веб-застосунком

Структуру даних, описану у бекенд частині проекту, важко назвати традиційною базою даних. Замість цього це набір об'єктів у пам'яті, які зберігають поточний стан експериментів, клієнтів та черг. Ці об'єкти містять дані про підключення клієнтів, стан експериментів та управління чергами, але не використовують системи управління базами даних (СУБД) для зберігання чи обробки даних. У традиційній базі даних дані зберігаються в структурованому форматі (таблиці, схеми), а доступ до них здійснюється через SQL або інші мови запитів. Дані нашого проекту зберігаються безпосередньо в пам'яті програми, і управління ними здійснюється через код JavaScript, тому це більше схоже на інфраструктуру управління станом програми, а не на повноцінну базу даних.

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Програмне забезпечення отримує вхідні дані від сенсорів висоти та керуючого сигналу, що подається на вентилятор для регулювання його швидкості. У свою чергу, вихідні дані, включаючи поточну висоту кулі та відхилення від заданої висоти, обробляються і відображаються на графіку або зберігаються в логах для подальшого аналізу, або наживо, якщо є така можливість. Для ефективного функціонування програми необхідно правильно організувати вхідні та вихідні дані, а також забезпечити їх попередню підготовку та правильне кодування.

До вхідних даних можна віднести:

– Процес введення коду авторизації.

– Персональні дані користувача.

– Напруга. Значення, що визначає швидкість вентилятора.

– Задані параметри контролеру (K_P, K_I, K_D). Налаштовуються для досягнення стабільності у процесі дослідження.

– Введення коментарів та висновків. Під кожним етапом налаштування експерименту для подальшого їх використання у підсумковому звіті.

До вихідних даних можна віднести:

– Поточний стан кульки у вакуумі експерименту.

– Отримані дані дослідження, що записані у звіті.

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби

Веб-орієнтований додаток був успішно протестований, за допомогою використання персональної ЕОМ. Її характеристика та периферія:

– Процесор: AMD Ryzen 7 5800H з 8 ядрами і 16 потоками, базова частота 3.2 ГГц з можливістю розгону до 4.4 ГГц.

– Графічна карта: NVIDIA GeForce RTX 3050 Ti.

– Оперативна пам'ять: 16 ГБ DDR4.

– Сховище: 1 ТБ M.2 NVMe PCIe 3.0 SSD.

– Дисплей: 16.0-дюймовий 4K OLED з співвідношенням сторін 16:10.

– Батарея: 63 Вт·год 3-елементна літій-іонна батарея.

– Клавіатура.

– Тачпад.

– Комп’ютерна миша.

Розроблений програмний додаток здатний функціонувати і на різних апаратних системах і альтернативних конфігураціях, навіть якщо характеристики трохи гірше за представлені вище.

2.6.2. Використані програмні засоби

Так як проект розроблявся на мові JavaScript, то і під час розробки та працювання над застосунком були використані наступна програма - Visual Studio Code (рис.2.5).

«Visual Studio Code» (VS Code) — це редактор коду від компанії Microsoft, який став дуже популярним серед розробників завдяки своїй потужності та гнучкості. Він підтримує безліч мов програмування, надаючи функції підсвічування синтаксису та автозавершення коду, що значно спрощує процес написання коду. Крім того, система дозволяє зручно керувати репозиторіями, робити коміти та переглядати історію змін прямо з редактора. Завдяки відкритому API, розробники можуть створювати власні розширення, тож VS Code також має величезний ринок розширень, де можна знайти додаткові інструменти, підтримку нових мов програмування, дебагери та багато іншого.

Окрім цього, «VS Code» пропонує вбудовані інструменти для налагодження коду, що дозволяють ставити брейкпоінти, переглядати змінні та контролювати виконання програм. Вбудований термінал дозволяє запускати команди безпосередньо з редактора. Редактор доступний на платформах Windows, macOS та Linux.

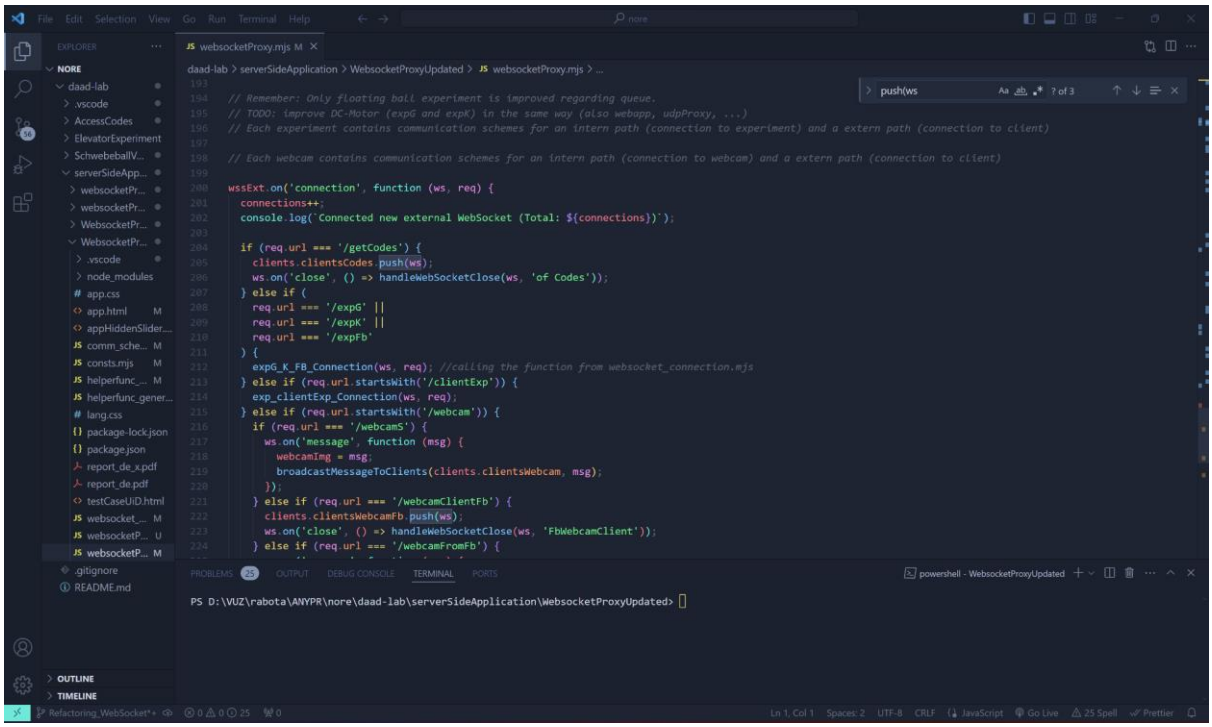

Рис.2.5. Вигляд програмного коду Lab4All у «VS Code»

2.6.3. Виклик та завантаження програми

Взаємодія із веб-сайтом може бути реалізована і за допомогою, не дуже потужної ЕОМ, комп'ютера, ноутбука, чи навіть, мобільного смартфона. Також, знадобиться гарне інтернет-з'єднання для уникнення неочікуваних помилок та збоїв. Але на боці університетів, завантаження програми вимагає більш ресурсо витратних методів та обладнання, потужних серверів, для утримання певної кількості експериментів.

Якщо є потрібність запустити проект локально, то нижче наведено алгоритм дії по його запуску:

– Потрібно в файлі «codes.html», що генерує коди авторизації, змінити адрес хосту запуску проекту на «ws://localhost:8080/» (рис.2.6-2.7). Цей рядок коду створює нове з'єднання «WebSocket» з сервером на локальному хості на порті 8080.

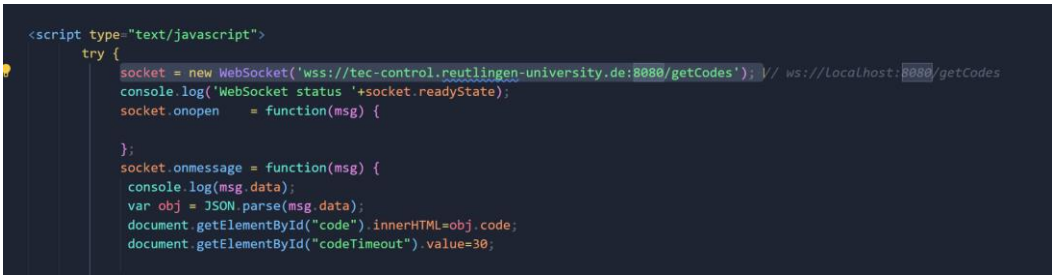

Рис.2.6. Попередній вигляд створення нового з'єднання

Рис.2.7. Створення нового з'єднання на локальному хості

– Після чого потрібно зберегти файл.

– Відкрити головний файл логіки проекту «websocketProxy.mjs». Та запустити його та дочекатися генерації Нових Сесійних кодів (рис.2.8).

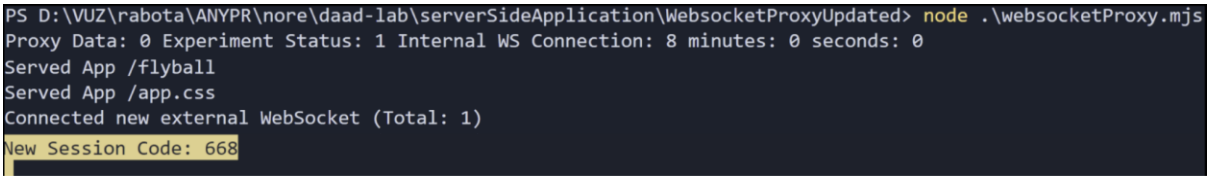

Рис.2.8. Запуск головного файлу проекту

– Далі потрібно, перейти на локальний сервер, на якому і розташований наш проект - «localhost:8080» та потрібний експеримент - «/flyball» (рис.2.9):

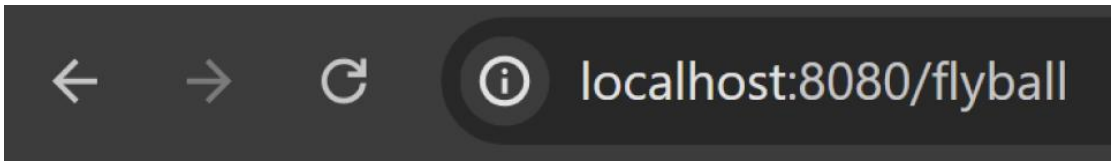

Рис.2.9. Перехід на локальный сервер експерименту

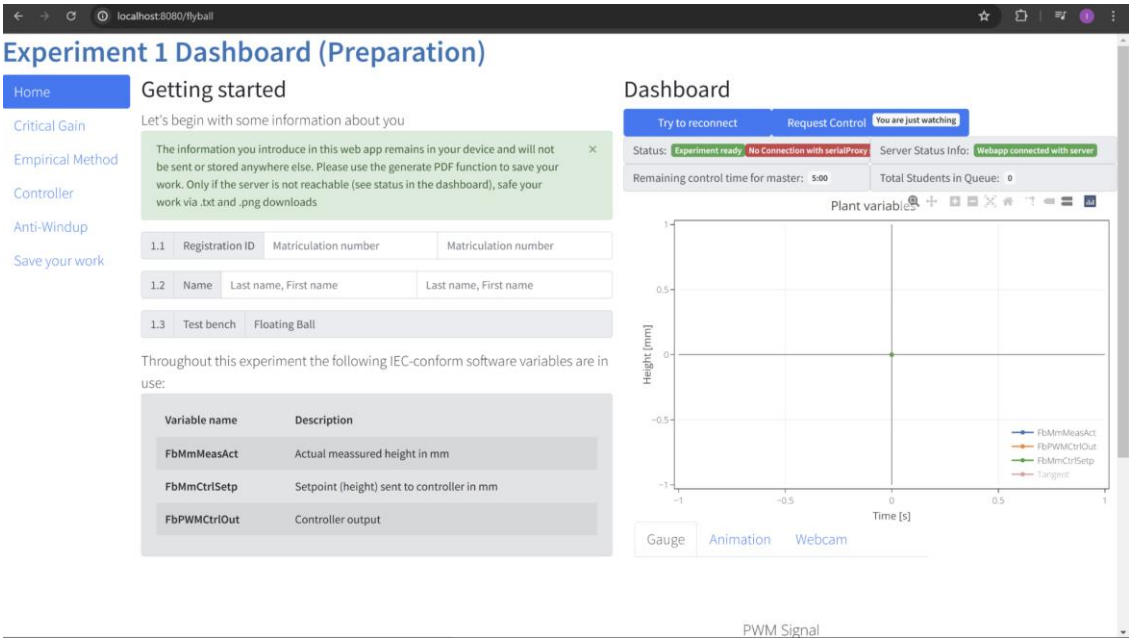

Рис.2.10. Вигляд експерименту на локальному серверу, з ще не оновленою дизайн частиною

– Для авторизації, нам потрібно запросити управління, натиснувши на кнопку «Request Control» (рис.2.11) та введенням тризначного коду (рис.2.12) (у нашому випадку - 668), який ми отримали у консолі, під час запуску основного логічного файлу експерименту.

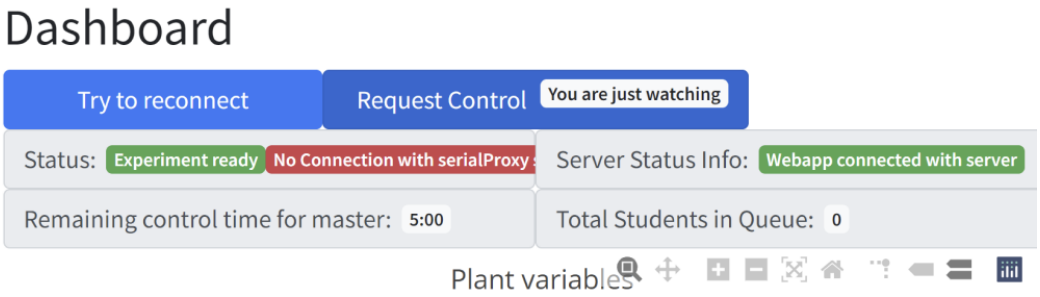

Рис.2.11. Приклад кнопки «Request Control»

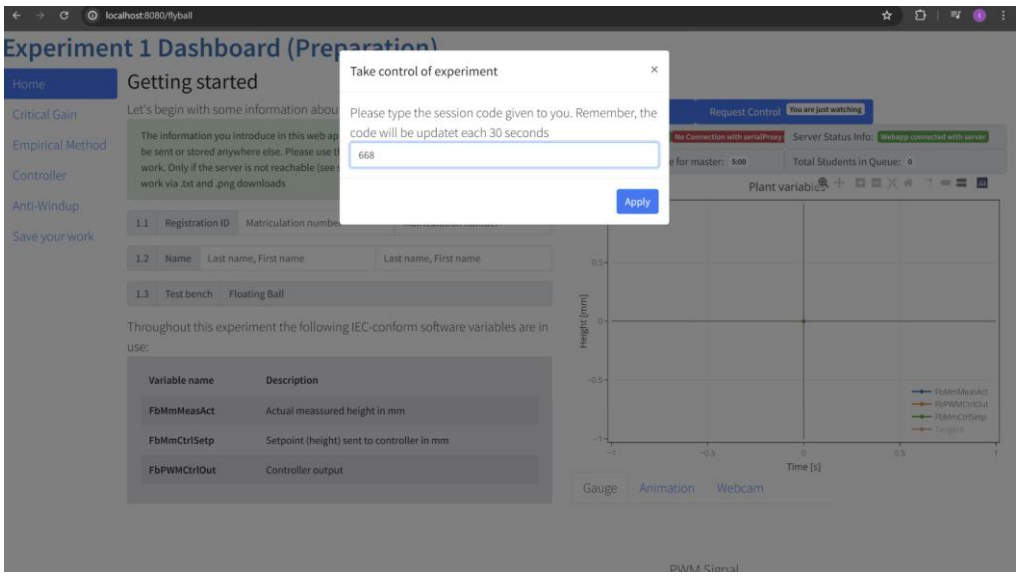

Рис.2.12. Приклад вводу коду авторизації

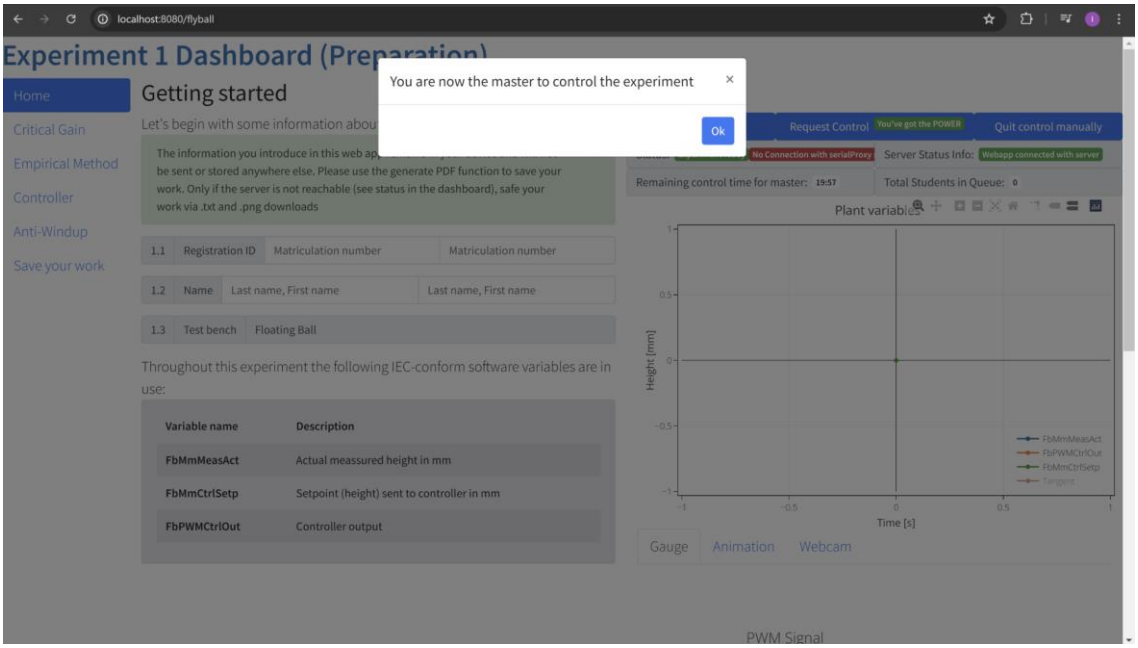

Рис.2.13. Приклад успішного отримання контролю та повного функціоналу веб-сторінки експерименту

2.6.4. Опис інтерфейсу користувача

Веб-сторінку експерименту можна знайти за посиланням «https://tec-control.reutlingen-university.de:8080/flyball». В описі інтерфейсу користувача буде використовуватися оновлений дизайн представленої веб-сторінки проекту.

Перше, що бачить користувач, це поп-ап повідомлення (рис.2.14). в якому можна побачити кілька порад по використанню експерименту.

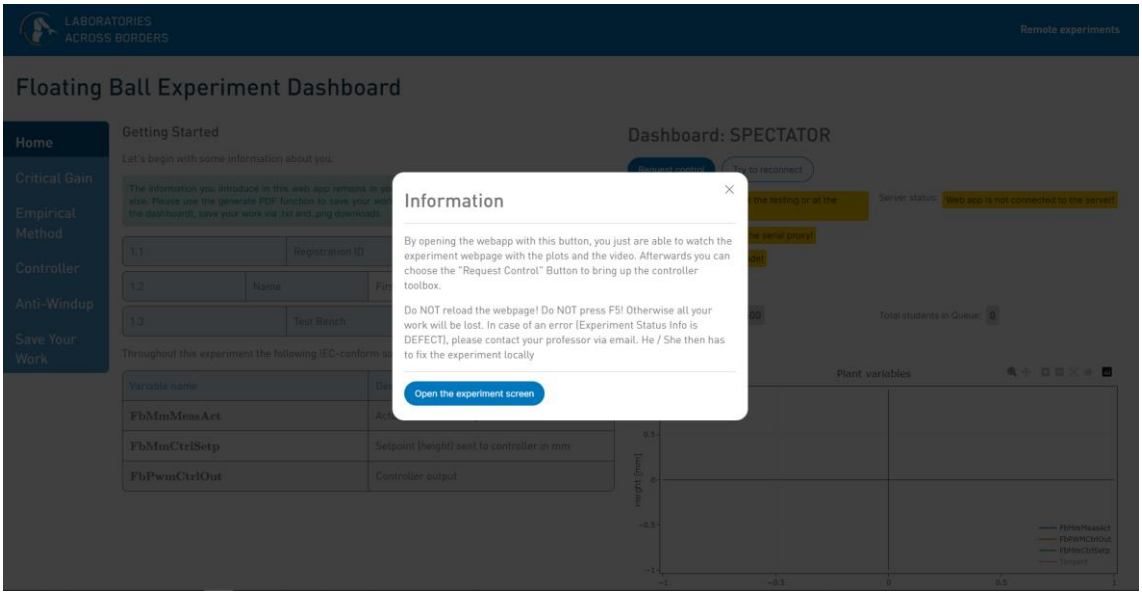

Рис.2.14. Приклад вводу коду авторизації

Після натискання кнопки Відкриття, ми знаходимося на головній веб-сторінці дослідницького експерименту. Перед нами відкривається доступ до всіх важливих сторінок сайту.

Роздивимося детальніше кожен із списку.

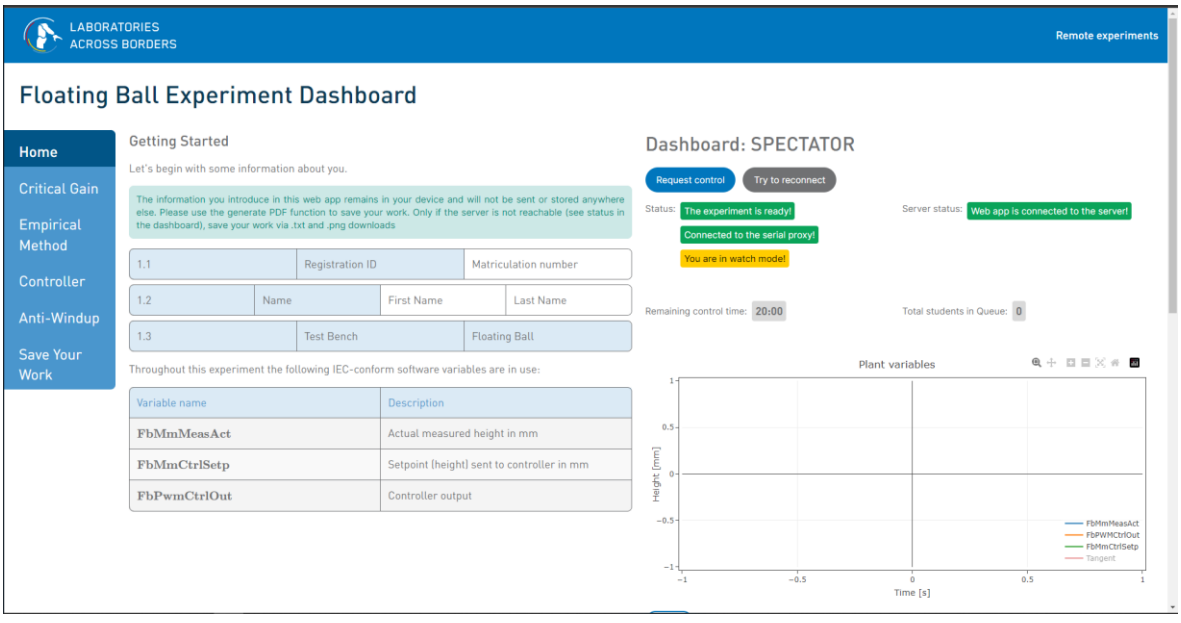

Рис.2.15. Сторінка «Home» з оновленим дизайном

– «Home». Зліва можна побачити панель вводу персональних даних користувача - Ім’я та Прізвище, Студентський ІД-номер та назва використовуваного експерименту, у даному випадку - «Floating Ball».

Праворуч можна побачити головний інтерфейс експерименту, панель авторизації від звичайного клієнта (глядача) - до керівника дослідженням (майстра). Також, важливі індикатори правильного функціонування експерименту, на Рис.2.15 можна побачити напис зеленого кольору «Connected to the serial proxy!», що вказує на правильну роботу системи.

Також, можливі ситуації певних збоїв системи, на рівні серверів чи навіть фізичного обладнання. У такому випадку, можна побачити наступне повідомлення (рис.2.16):

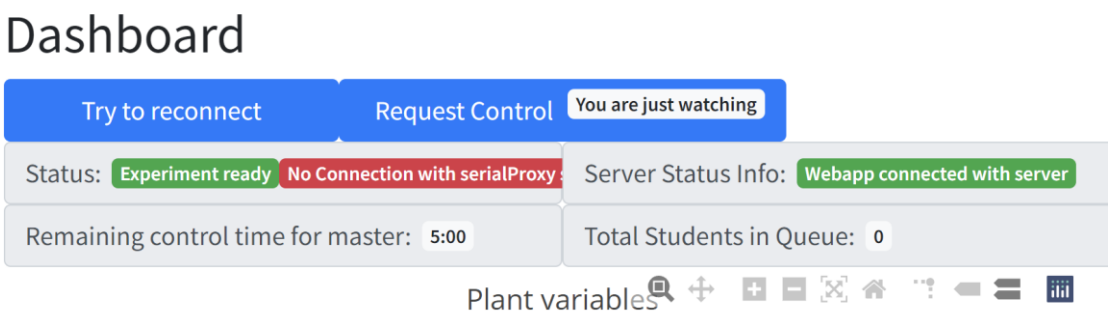

Рис.2.16. Приклад вигляду збою на інформаційній панелі експерименту

Також, нижче можна побачити таймер використання експерименту, який запуститься як тільки користувач увійде в систему. Загальна кількість студентів у черзі на отримання контролю експерименту (рис.2.17). І, нарешті, головний елемент системи - графік експерименту. Під графіком знаходиться одна з найважливіших панелей параметрів для запуску експерименту (рис.2.18):

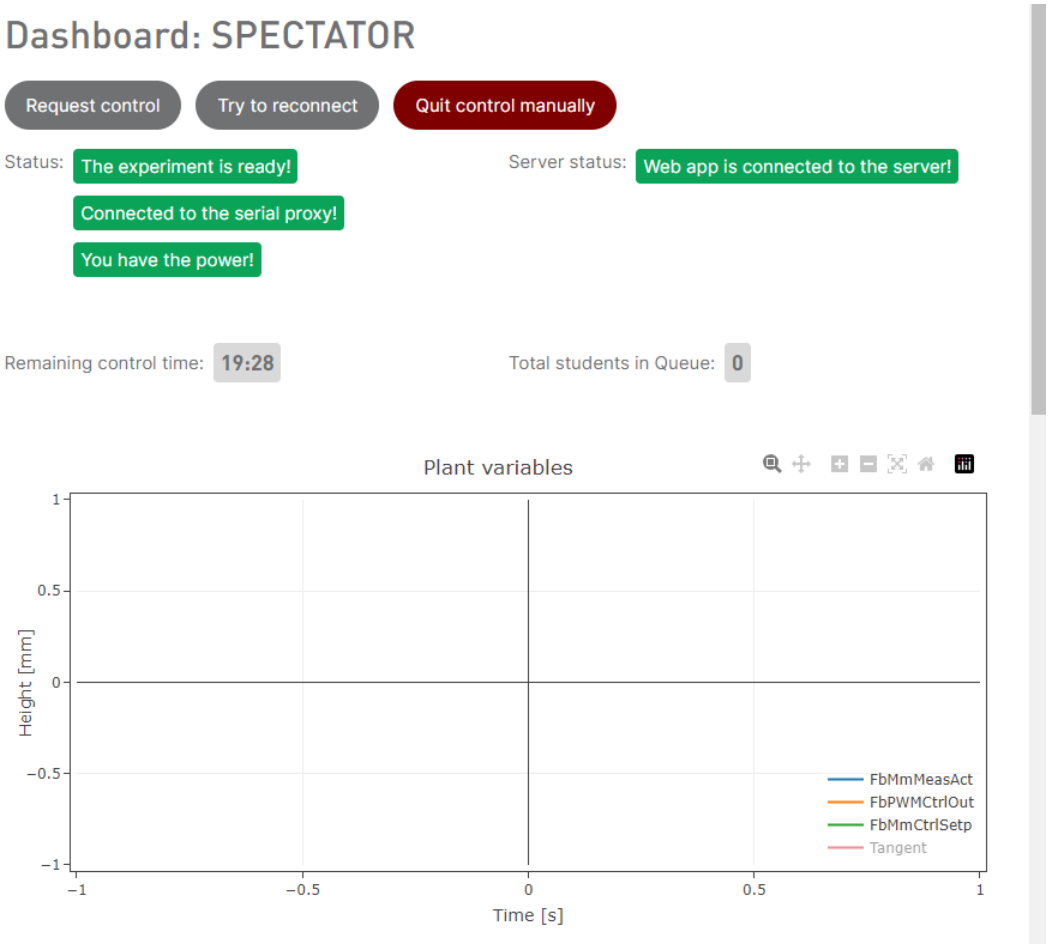

Рис.2.17. Вигляд графіку експерименту

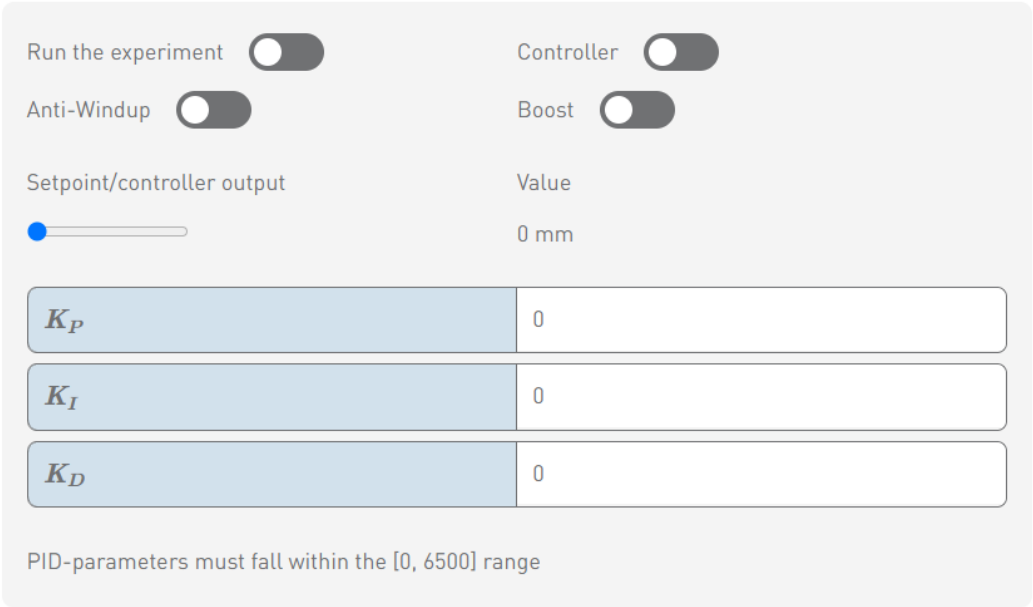

Рис.2.18. Вигляд параметрів експерименту

Тут можна побачити параметри впливу різних фізичних значень, напруги на кульку, з якою швидкістю буде її політ, значення параметрів контролеру (K_P, K_I, K_D), додавання або обмеження впливу сторінок «Controller» та «Anti-Windup» на процес експерименту. Також, кнопка запуску експерименту "Run the experiment", що слугує для початку дослідження.

Трохи нижче знаходиться діаграма манометру експерименту (рис.2.19), що підтримує кульку на певній висоті. Сигнал дозволяє регулювати швидкість мотора шляхом зміни ширини імпульсів при постійній частоті, змінює середню напругу, що подається на мотор, і впливає на силу повітряного потоку, і відповідно, на поточну висоту кульки.

Рис.2.19. Вигляд графіку манометру

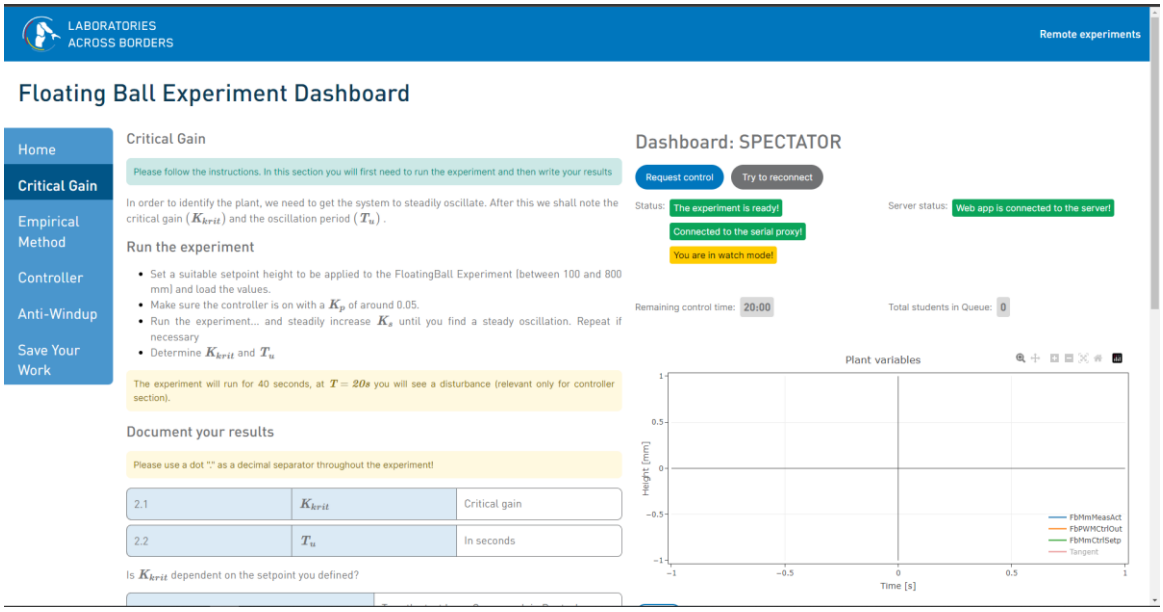

Рис.2.20. Вигляд сторінки «Critical Gain»

– «Critical Gain». На цій сторінці (рис.2.20), здебільшого представлена теоретична інформація про фізичні та практичні принципи використання експерименту. Описується процес визначення критичного коефіцієнта підсилення в системі керування.

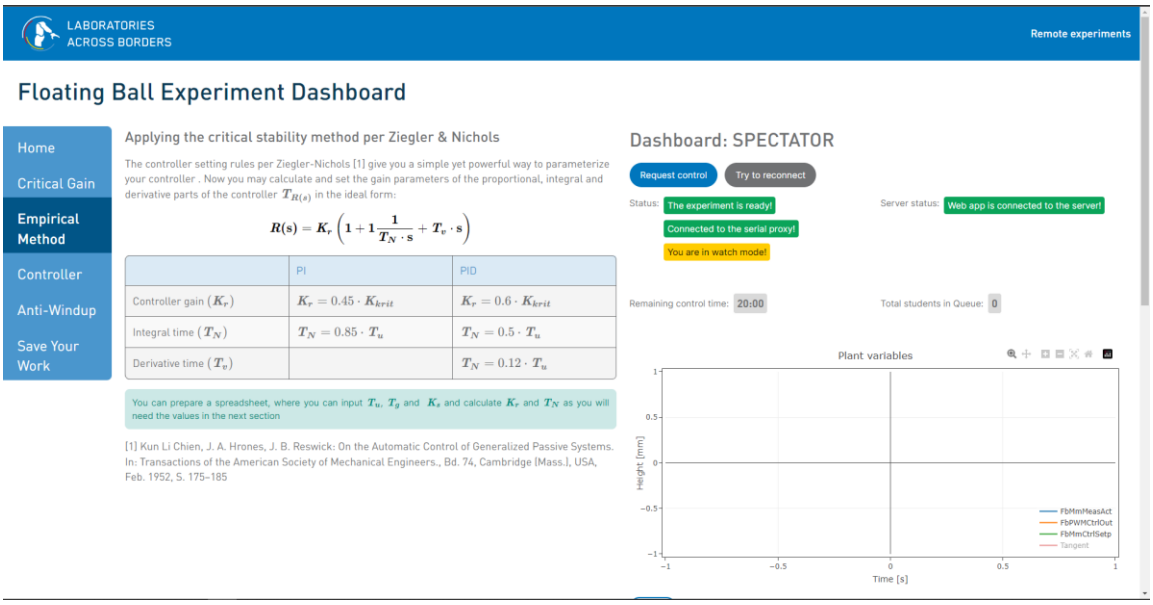

Рис.2.21. Сторінки «Empirical Method»

– «Empirical Method». На цій сторінці (рис.2.21), описується правила налаштування контролера, що дає простий, але потужний спосіб параметризації контролера. Також, приводиться фізична формула для розрахунку та посилення пропорційної, інтегральної та похідної частин контролера.

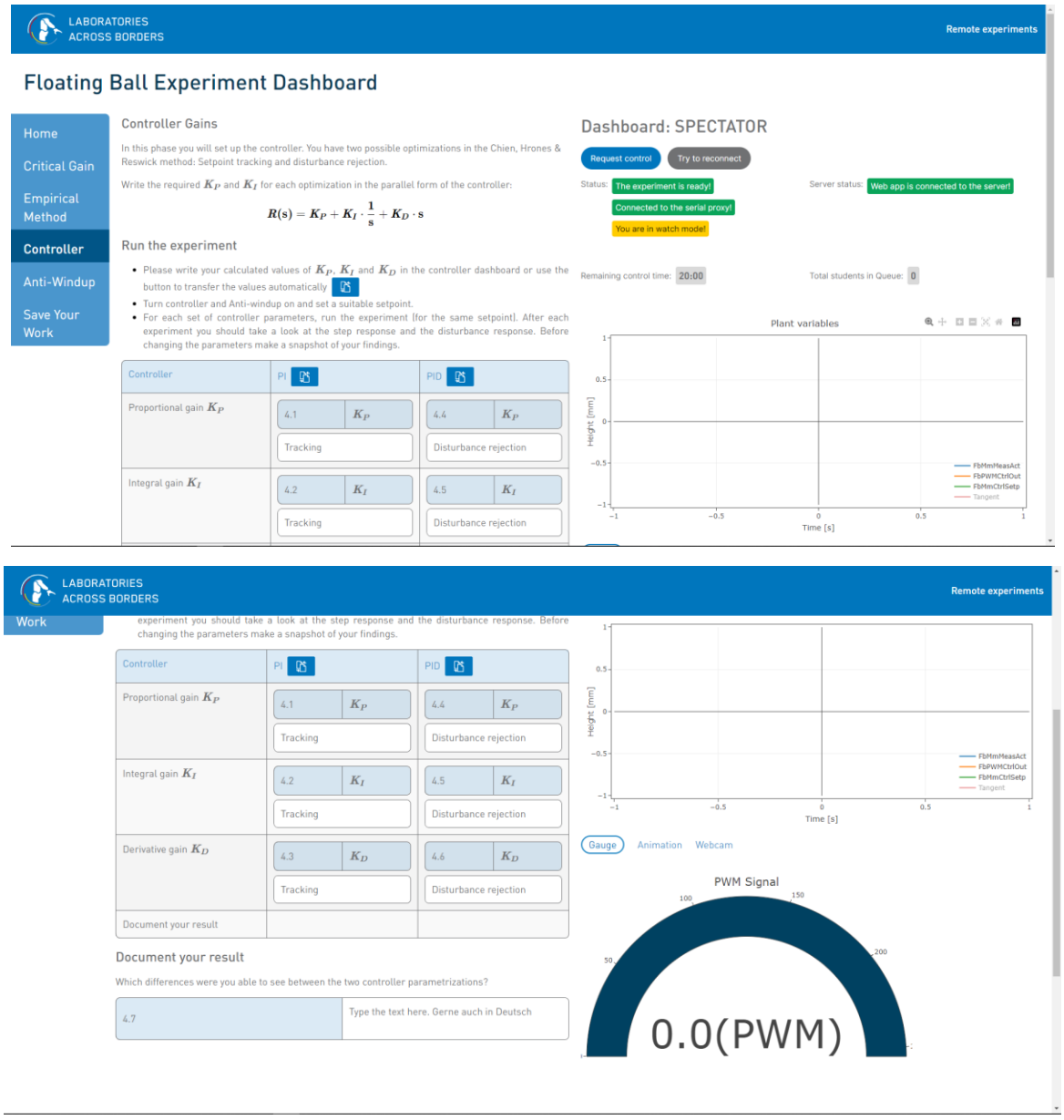

Рис.2.22. Приклад вигляду сторінки «Controller»

– «Controller». На цій сторінці (рис.2.22), зображені певні теоретичні відомості для налаштування експерименту. Також, представлений детальний алгоритм по запуску та взаємодії користувача з експериментом, та значення параметрів контролеру (K_P, K_I, K_D). K_P - значення пропорційного посилення, K_I - значення інтегрального посилення та K_D - похідний приріст.

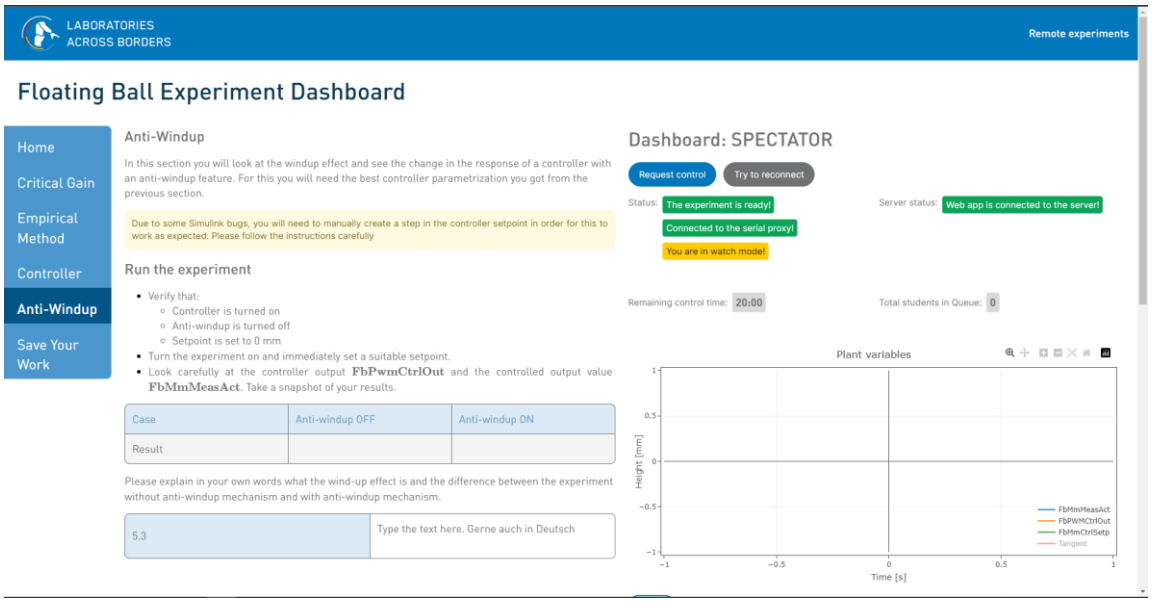

Рис.2.23. - приклад вигляду сторінки «Anti-Windup»

– «Anti-Windup». У цьому розділі веб-додатку (рис.2.23), можна подивитися на ефект закручування та побачити зміну реакції контролера з функцією запобігання закручуванням. Цей розділ повністю пов’язаний з попереднім, тому для цього знадобиться найкраща параметризація контролера.

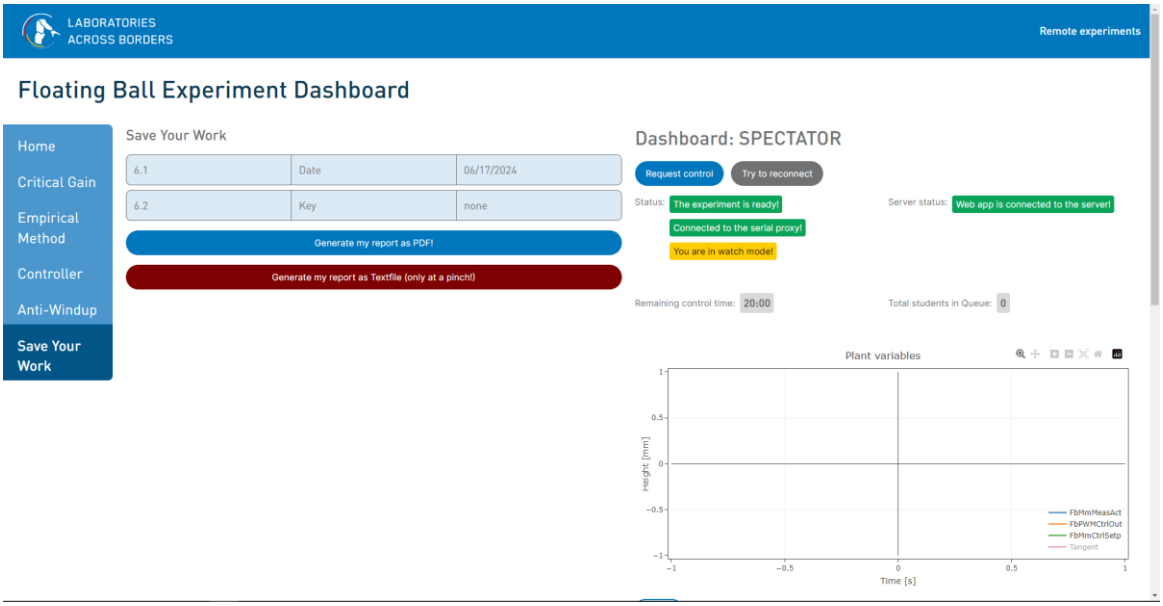

Рис.2.24. Вигляд сторінки «Save Your Work»

– «Save Your Work». У цьому розділі (рис.2.24) є можливість зберегти загальні введені дані користувача, від персональних до отриманих після експерименту шляхом генерації у документ формату .pdf або у файл формату .txt.

– Проведемо авторизацію на сайті проекту.

Для цього запросимо контроль, натисканням кнопки «Request Control».

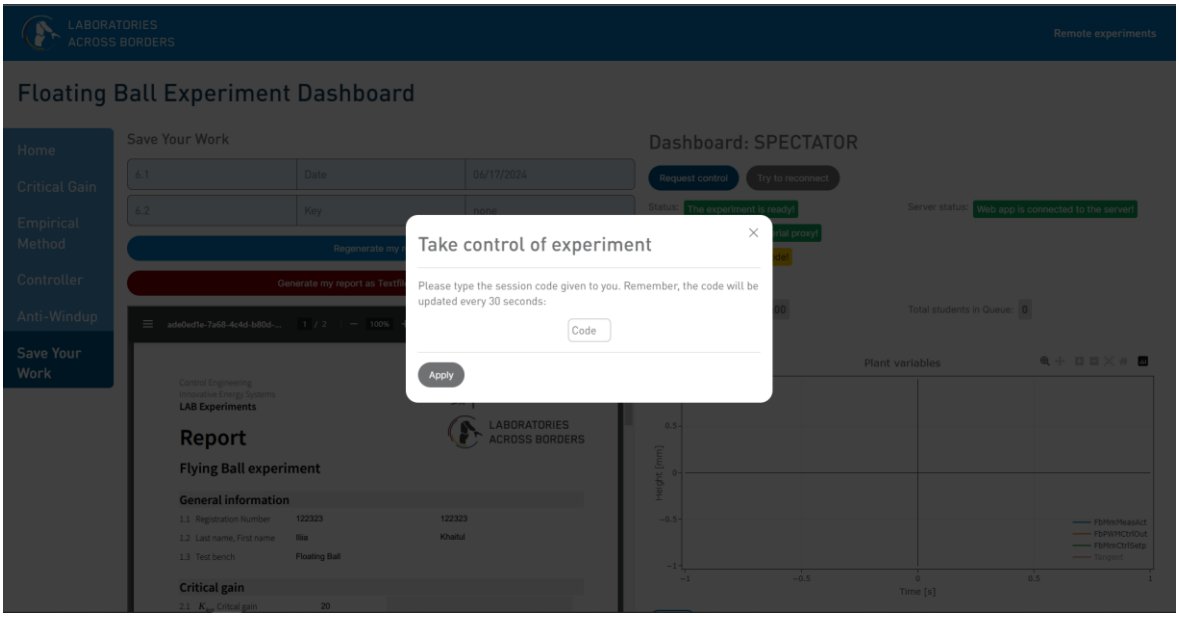

Рис.2.25. Отримання контролю та введення тризначного коду

– Скористаємося файлом адміністратора, де автоматично генеруються тризначні коди авторизації (рис.2.26-2.27):

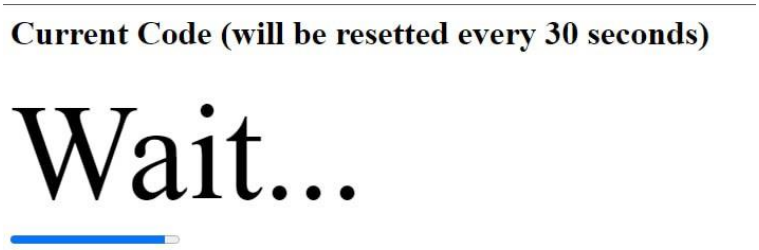

Рис.2.26. Приклад генерації тризначного коду

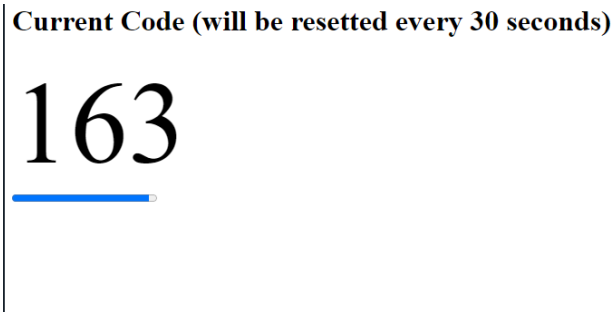

Рис.2.27. Приклад згенерованого тризначного коду

– Введемо код на сторінці отримання контролю, якщо все пройшло успішно, то отримаємо наступне повідомлення (рис.2.28). Також, спробуємо ввести фізичні дані та запустити експеримент (рис.2.29-2.30).

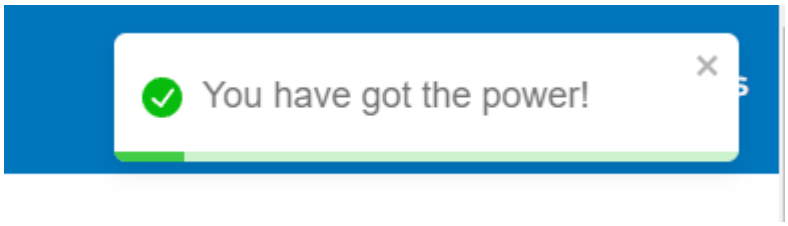

Рис.2.28. Отримання повідомлення авторизації

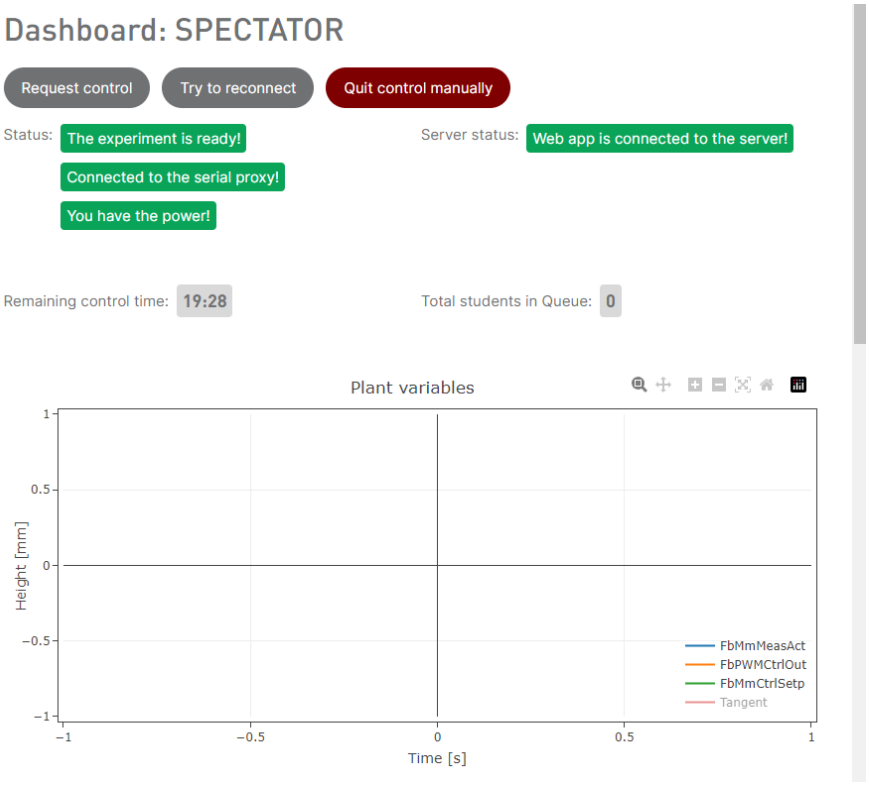

Рис.2.29. Зміна вигляду інтерфейсу після отримання контролю

Рис.2.30. Приклад зміни вигляду роботи експерименту

– Після чого, збережемо отримані дані у документ .pdf (рис.2.31) та файл .txt (рис.2.32) на сторінці Save Your Work:

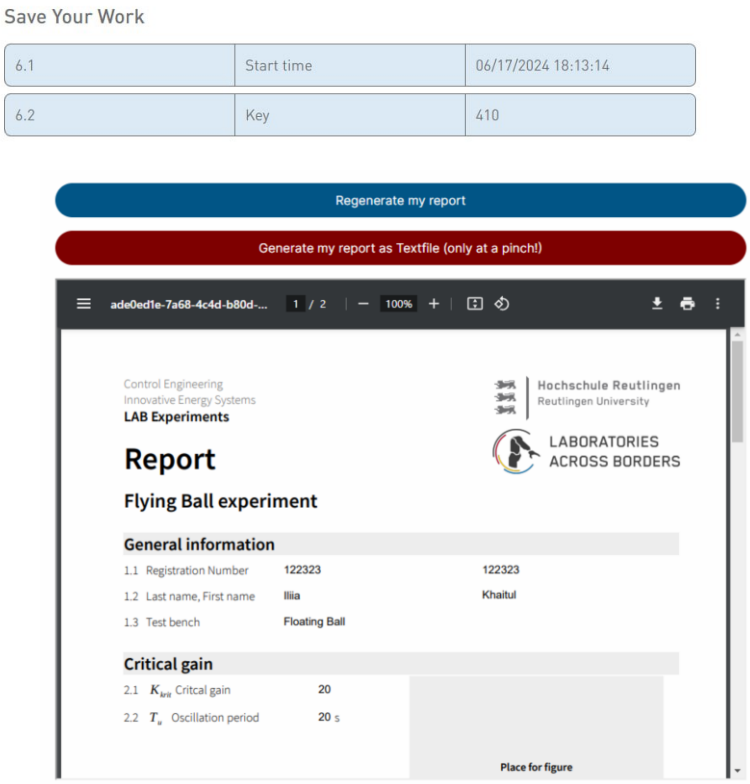

Рис.2.31. Генерація звіту у документ .pdf

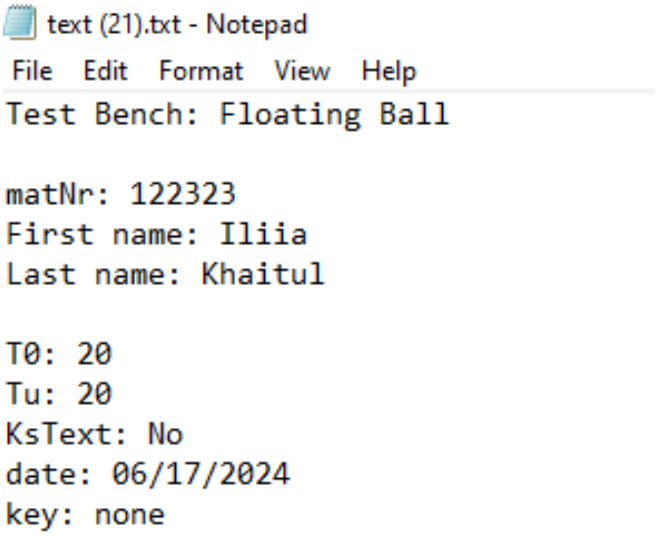

Рис.2.32. - Генерації звіту у файл .txt

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Визначення трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів (q) - 1 548;

2. коефіцієнт складності програми (C) - 1,4;

3. коефіцієнт кореляції програми в ході її розробки (p) - 0,2;

4. годинна заробітна плата програміста (Cnp) - 144,05 грн;

Аналізуючи статистичні дані, взяті з української платформи «DOU» [https://jobs.dou.ua/salaries/?period=2023-12&position=Junior%20SE], з грудня 2023 року, середня щомісячна зарплата програміста на посаді Junior JavaScript Software Developer становить 626$ [30].

Згідно інформації Національного банку України, наразі, валютний курс доллара до гривні - (40.5 грн), що прирівнює місячну виплату до 25,353 грн [31]. При загальному графіку роботи, з 09:00 до 17:00, маємо 8 годин на день [34].

Отже, стандартно кількість робочих днів на місяць дорівнює приблизно 22 дні. 22 дні помножимо на 8 годин в день - отримаємо 176 годин на місяць.

Тобто, щоб дізнатися погодинну заробітну плату, ділимо місячну зарплату на кількість годин у місяці, 25 353 / 176 ≈ 144,05 грн.

5. коефіцієнт збільшості витрат праці внаслідок недостатнього опису задачі (B) - 1,1;

6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності (k) - 0,8;

7. вартість машино-годин ЕОМ - 0,76 грн / год;

Під час написання кваліфікаційної роботи витрати обмежувалися лише на інтернет та електроенергію при роботі за ноутбуком. Період розробки у межах кваліфікаційної роботи становить 9 місяців, з 01.06.2023 до 29.02.2024. Згідно з

офіційними затвердженими даними України, вартість 1 кВт/год вартує 2,64 грн [https://index.minfin.com.ua/ua/tariff/electric/2023-06-01/] [32] .

У ході роботи активно використовувався модель ноутбуку «Asus VivoBook Pro 16X OLED», яка споживає у середньому 63Вт/год. Тобто, вартість електроенергії за годину роботи становила 0,063 * 2,64 = 0,166 грн [27]. Під час виконання кваліфікаційного проекту, перебуваючи у гуртожитку, мною, щомісячно сплачується 10 євро [33].

Згідно інформації Національного банку України, наразі, валютний курс євро до гривні - (43.8 грн), що прирівнює місячну виплату до 438 грн. З середнього урахування, що в одному місяці 30 днів, що є 720 годин, вартість використання студентського інтернету за 1 годину становить 438 / 720 = 0,6 грн.

Тобто, поточна вартість машино-годин ЕОМ буде дорівнювати 0,166 + 0,6 = 0,76 грн / год.

Нормування праці при розробці програмного забезпечення значно ускладнюється через творчий характер роботи програміста. Відтак, для оцінки трудомісткості створення програмного забезпечення можна використовувати систему моделей, які забезпечують різний рівень точності.

Трудомісткість розробки програмного забезпечення можна обчислити за допомогою формули (3.1):

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \,, \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p) , \quad (3.2)$$

де q - передбачуване число операторів (1 548);

C - коефіцієнт складності програми (1,4);

p - коефіцієнт кореляції програми в ході її розробки (0,2);

$$Q = 1548 \cdot 1{,}4 \cdot (1 + 0{,}2) = 2600{,}6 \text{ людино-годин.}$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста, за допомогою формули:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k} , \quad (3.3)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі менше двох років, буде дорівнювати - 0,8. За рахунок підстановки значень, отримаємо:

$$t_u = (2\,600{,}6 \cdot 1{,}1) / (75 \cdot 0{,}8) = 47{,}6 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задач, із використанням формули:

$$t_a = \frac{Q}{(20..25)\cdot k}\text{, людино-годин,} \qquad (3.4)$$

де Q - умовне число операторів програми (2 600,6);

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі менше двох років, буде дорівнювати - 0,8. За рахунок підстановки значень, отримаємо:

$$t_a = 2600{,}6 / (20 \cdot 0{,}8) = 162{,}5 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі, з використанням формули:

$$t_n = \frac{Q}{(20..25)\cdot k}\text{, людино-годин.} \qquad (3.5)$$

де Q - умовне число операторів програми (2 600,6);

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі менше двох років, буде дорівнювати - 0,8. За рахунок підстановки значень, отримаємо:

$$t_n = 2600{,}6 / (25 \cdot 0{,}8) = 130{,}03 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання, з використанням формули:

$$t_n = \frac{Q}{(4..5)\cdot k}\text{, людино-годин.} \quad (3.6)$$

Підставивши значення отримаємо:

$$t_{отл} = 2600{,}6 / (4 \cdot 0{,}8) = 812{,}7 \text{ людино-годин,}$$

– за умови комплексного налагодження завдання, з використанням формули:

$$t^k_{отл} = 1{,}5 \cdot t_{отл}\text{, людино-годин.} \quad (3.7)$$

Підставивши значення отримаємо:

$$t^k_{\text{отл}} = 1{,}5 \cdot 812{,}7 = 1219{,}05 \text{ людино-годин.}$$

Витрати праці на підготовку документації, з використанням формули:

$$t_д = t_{др} + t_{до}\text{ , людино-годин,} \quad (3.8)$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису та знаходиться за формулою:

$$t_{др} = \frac{Q}{(15..20)\cdot k}\text{ , людино-годин,} \quad (3.9)$$

Підставляючи відповідні значення, отримаємо:

$$t_{др} = 2600{,}6 / (16 \cdot 0{,}8) = 203{,}1 \text{ людино-годин,}$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації та знаходиться за формулою:

$$t_{до} = 0{,}75 \cdot t_{др} \text{ , людино-годин.} \quad (3.10)$$

Підставляючи відповідні значення, отримаємо:

$$t_{до} = 0{,}75 \cdot 203{,}1 = 152{,}3 \text{ людино-годин,}$$

Тож, тепер можемо отримати значення $t_д$ за формулою:

$$t_д = 203{,}1 + 152{,}3 = 355{,}4 \text{ людино-годин.}$$

Отже, після того як отримали всі значення змінних, повертаючись до формули (3.1), трудомісткість розробки програмного забезпечення буде дорівнювати:

$$t = 50 + 47.6 + 162{,}5 + 130{,}03 + 812{,}7 + 355{,}4 = 1558{,}23 \text{ людино-годин;}$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ зображені за формулою (3.11):

$$К_{ПО} = З_{ЗП} + З_{Мч} \text{ , грн.} \quad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}\text{ , грн,} \qquad (3.12)$$

де t - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година.

Використовуючи значення середньої погодинної зарплати програміста на посаді Junior JavaScript Software Developer, що становить 144,05 грн, отримаємо:

$$З_{ЗП} = 1558{,}23 \cdot 144{,}05 = 224\,463{,}03 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ знаходиться за формулою:

$$З_{мв} = t_{отл} \cdot C_{мч}\text{ , грн,} \qquad (3.13)$$

де: $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, (1219,05 год);

$C_{мч}$ - вартість машино-години ЕОМ, (0,76 грн / год;);

Визначаємо вартість машинного часу, завдяки підставки певних значень до формули:

$$З_{мв} = 1219{,}05 \cdot 0{,}76 = 926{,}4 \text{ грн.}$$

Отже, повертаючись до формули (3.2) можемо знайти значення витрат на створення ПЗ:

$$К_{ПО} = 224\,463{,}03 + 926{,}4 = 207\,942{,}55 \text{ грн.}$$

Очікуваний період створення ПЗ знаходиться використовуючи наступну формулу:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс}, \quad (3.14)$$

де B_k - число виконавців;

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p = 176 годин);

t - трудомісткість розробки програмного забезпечення.

Отже, очікуваний період створення ПЗ становить:

$$T = 1558{,}23 / 1 \cdot 176 = 8{,}85 \text{ міс};$$

Висновок: Під час проведення детальної аналітики вихідних даних кваліфікаційної роботи, можна зробити певний підсумок, для реалізації програмного забезпечення для управління та контролю дослідницьких експериментів в межах проекту Lab4All потрібно задіяти 1558,23 людино-годин, остаточний бюджет витрат складає 207 942,55 гривень, очікуваний термін якого становить 8,85 місяців.

ВИСНОВКИ

Відповідно до завдання та мети, успішно було проведено оптимізацію, рефакторинг та дорозробку програмного забезпечення для взаємодії та використання дослідницького обладнання в рамках проекту Laboratories Across Borders.

Додаток є дуже актуальним вирішенням досить важливих проблем, як у напрямку дистанційного навчання, так і у напрямку впровадження автоматизації та оптимізації роботи, тим паче, у такі складні часи, які зараз переживає Україна.

Мета Lab4All полягає в забезпеченні учнів постійного та безперебійного доступу до дослідницького обладнання для виконання певних фізичних навчальних експериментів. Можливості оптимізації експериментів під певні потрібні вимоги, за рахунок надання вводу різноманітних фізичних змінних та величин, що прямо впливають на виконання та отримані дані дослідження.

Проект надає користувачі певні можливості:

– Онлайн доступ до дослідницького обладнання, яке знаходиться в двох університетах, українському та німецькому.Теоретичні фізичні відомості про використання експериментів.

– Можливість власного введення певних фізичних величин, та аналіз їх впливу на результат виконаного дослідження.Отримання результатів експерименту, як у наочному (у вигляді змінення графіку), так і текстовому форматі (у вигляді звіту).

– Надання можливості збереження отриманих результатів дослідження, за рахунок збереження їх у звіти, документу .pdf та .txt.

Розробка додатку була виконана, за допомогою використання мови програмування JavaScript, його середовища Node.js, та його модулів та бібліотек, таких як: WebSocket, HTTPS та FS.

Згідно з розрахунками економічного розділу, для реалізації проекту знадобиться задіяти 1558,23 людино-годин, бюджет остаточних витрат

складатиме 207 942,55 гривень, і очікуваний термін розробки становитиме 8,85 місяців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Головний сайт проекту Laboratories Across Borders. / URL: https://www.lab-project.eu//. Дата звернення: 10.05.2024.
2. What is JavaScript? / URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. Дата звернення: 11.05.2024.
3. Робототехніка та автоматизація: минуле, сьогодення та майбутнє. / URL: https://www.education.ua/blog/48114/. Дата звернення: 11.05.2024.
4. RoboChem лідирує в автоматизації хімічних досліджень на основі ШІ. / URL: https://bit.ly/3XznLTV. Дата звернення: 11.05.2024.
5. Комп'ютеризовані вимірювальні системи і технології. / URL: https://okv.suitt.edu.ua/wp-content/uploads/2022/09/152_kompiuteryzovani_vym_s_my_i_tekhnolohiii.pdf. Дата звернення: 11.05.2024.
6. Методологія аналізу українського сегменту соціальних мереж та месенджерів. / URL: https://detector.media/infospace/article/194698/2021-12-11-metodologiya-analizu-ukrainskogo-segmentu-sotsialnykh-merezh-ta-mesendzheriv/. Дата звернення: 12.05.2024.
7. ЩО ТАКЕ ВЕЛИКІ ДАНІ (BIG DATA)? / URL: https://university.sigma.software/what-is-big-data/. Дата звернення: 14.05.2024.
8. Машинне навчання: як штучний інтелект вчиться і розвивається / URL: https://bizmag.com.ua/mashynne-navchannya/. Дата звернення: 14.05.2024.
9. МАШИННЕ НАВЧАННЯ ТА АНАЛІЗ ВЕЛИКИХ ДАНИХ: ЩО ЦЕ, ЯК ПРАЦЮЄ І ДЛЯ ЧОГО ПОТРІБНО БІЗНЕСУ / URL: https://aiconference.com.ua/uk/news/mashinnoe-obuchenie-i-analiz-bolshih-dannih-chto-eto-kak-rabotaet-i-dlya-chego-nugno-biznesu-98145. Дата звернення: 14.05.2024.
10. Go Lab / URL: https://www.golabz.eu/. Дата звернення: 17.05.2024.

11. Writing WebSocket client applications / URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_client_applications. Дата звернення: 18.05.2024.

12. WebSocket / URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSocket. Дата звернення: 18.05.2024.

13. JavaScript/ URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript. Дата звернення: 19.05.2024.

14. Node.js Introduction/ URL: https://www.w3schools.com/nodejs/nodejs_intro.asp. Дата звернення: 19.05.2024.

15. Node.js Modules / URL: https://www.w3schools.com/nodejs/nodejs_modules.asp. Дата звернення: 19.05.2024.

16. Node.js v22.3.0 documentation / URL: https://nodejs.org/api/fs.html. Дата звернення: 24.05.2024.

17. ws: a Node.js WebSocket library / URL: https://www.npmjs.com/package/ws. Дата звернення: 24.05.2024.

18. Node.js HTTPS Module / URL: https://www.w3schools.com/nodejs/ref_https.asp. Дата звернення: 24.05.2024.

19. Сучасний підручник з JavaScript / URL: https://uk.javascript.info/. Дата звернення: 26.05.2024.

20. What is Client-side scripting with JavaScript? / URL: https://codedamn.com/news/javascript/client-side-scripting. Дата звернення: 26.05.2024.

21. Mastering Asynchronous JavaScript: Promises, async/await, Error Handling, and More / URL: https://dev.to/kelvinguchu/mastering-asynchronous-javascript-promises-asyncawait-error-handling-and-more-41ph. Дата звернення: 26.05.2024.

22. The V8 JavaScript Engine / URL: https://nodejs.org/en/learn/getting-started/the-v8-javascript-engine#the-v8-javascript-engine. Дата звернення: 02.06.2024.

23. Node.js Demystified: Your Complete Guide to Mastering Server-Side JavaScript / URL: https://clouddevs.com/node/server-side-javascript-guide/. Дата звернення: 02.06.2024.

24. How Do Websockets Work? / URL: https://sookocheff.com/post/networking/how-do-websockets-work/. Дата звернення: 04.06.2024.

25. TCP: How the Transmission Control Protocol works - IONOS / URL: https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:the-internet/xcae6f4a7ff015e7d:transporting-packets/a/transmission-control-protocol--tcp. Дата звернення: 04.06.2024.

26. The Road to WebSockets / URL: https://websocket.org/guides/road-to-websockets/. Дата звернення: 04.06.2024.

27. ASUS Vivobook 16X OLED / URL: https://www.asus.com/de/laptops/for-home/vivobook/asus-vivobook-16x-oled-m3604/. Дата звернення: 15.06.2024.

28. VS Code — Documentation for Visual Studio Code / URL: https://code.visualstudio.com/Docs. Дата звернення: 16.06.2024.

29. Visual Studio Code Extension API / URL: https://code.visualstudio.com/api. Дата звернення: 16.06.2024.

30. DOU - СТАТИСТИКА ЗАРПЛАТ/ URL: https://jobs.dou.ua/salaries/?period=2023-12&position=Junior%20SE. Дата звернення: 18.06.2024.

31. Офіційний курс гривні щодо іноземних валют/ URL: https://bank.gov.ua/ua/markets/exchangerates. Дата звернення: 18.06.2024.

32. МІНФІН - Тарифи на електроенергію з 1.06.2023 по 31.05.2024/ URL: https://index.minfin.com.ua/ua/tariff/electric/2023-06-01/. Дата звернення: 18.06.2024.

33. Selfnet - Willkommen bei Selfnet/ URL: https://www.selfnet.de/support/faq.html#membershipfee/. Дата звернення: 18.06.2024.

34. Як працюватимемо та відпочиватимемо у квітні 2024 року/ URL: https://kadrovik.isu.net.ua/news/571207-yak-pratsyuvatymemo-ta-vidpochyvatymemo-u-kvitni-2024-roku. Дата звернення: 18.06.2024.

35. Latest Insights into Node.js: Analyzing Usage Statistics and Future Trends Beyond 2024 / URL: https://www.agileinfoways.com/blogs/nodejs-statistics/. Дата звернення: 20.06.2024.

36. WebSockets and Node.js - testing WS and SockJS by building a web app / URL: https://ably.com/blog/web-app-websockets-nodejs. Дата звернення: 20.06.2024.

37. Understanding the Node.js File System Module (FS) / URL: https://kinsta.com/knowledgebase/nodejs-fs/. Дата звернення: 20.06.2024.

38. The fs Module in Node.js: A Short Guide to File System Interaction / URL:https://blog.risingstack.com/fs-module-in-node-js/. Дата звернення: 20.06.2024.

39. A Detailed look into the Node.js HTTP module / URL : https://mirzaleka.medium.com/a-detailed-look-into-the-node-js-http-module-680eb5e4548a. Дата звернення: 21.06.2024.

40. JavaScript modules / URL : https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules. Дата звернення: 21.06.2024.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

websocketProxy.mjs

```
// Importing constants
import {
  SocketState,
  CommunicationType,
  CommunicationExperiment,
  ConnectionStatus,
  settings,
  experiments,
  clients,
  masterClients,
  allowedUid,
  fbExperimentConstants,
  eExperimentConstants,
  secret,
  sessionCode,
  ConstantsFB,
  webcamImg,
} from './consts.mjs';
// Importing specific helper functions
import {
  SelfTstExpE,
  checkStatusE,
  SendProxyDataE,
  SendQueuePositionE,
  /* SendIntConStateToClientsFb, */
  SendClientStateToExpFb,
  checkStatusFb,
  SendExpStateToClientsFb,
  SendQueueNumberFb,
  SendQueuePositionFb,
  TimerFb,
  ResetExpDataFb,
  RemoveClientfromQueueFb,
  QuitMasterFb,
  setMasterE,
```

```
  kickMasterE,
  RemoveClientfromQueueE,
} from './helperfunc_fb_e.mjs';
// Importing general helper function to check UID validity
import { isUidValid } from './helperfunc_general.mjs';
// Dependencies
import WebSocket from 'ws';
import fs from 'fs';
import https from 'https';
import http from 'http';
import colors from 'colors';
// Creating an HTTP server for internal connections
const serverInt = http.createServer();
// Read files
const fileContents = {
  appHTML: fs.readFileSync('appHiddenSlider.html'),
  appCSS: fs.readFileSync('app.css'),
  appLangCSS: fs.readFileSync('lang.css'),
  appPDF: fs.readFileSync('report_de.pdf'),
  appFlyBall: fs.readFileSync('../../SchwebeballVersuch/app.html'),
  appFlyBallPDF: fs.readFileSync('../../SchwebeballVersuch/report_en.pdf'),
  appElevator: fs.readFileSync('../../ElevatorExperiment/app.html'),
  appPdfElevator: fs.readFileSync(
    '../../ElevatorExperiment/report_en_elevator.pdf'
  ),
  testEmeneo: fs.readFileSync('testCaseUiD.html'), // DELETE FOR PRODUCTION
};
// Creating and connection to the server
const serverExt = http.createServer(function (req, res) {
  console.log('Served App ' + req.url);
  if (req.url == '/') {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.end(fileContents.appHTML);
  } else if (req.url == '/flyball') {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.end(fs.readFileSync('../../SchwebeballVersuch/app.html'));
  } else if (req.url == '/app.css') {
    res.writeHead(200, { 'Content-Type': 'text/css' });
    res.end(fileContents.appCSS);
  } else if (req.url == '/report_de.pdf') {
    res.writeHead(200, { 'Content-Type': 'application/pdf' });
```

```
    res.end(fileContents.appPDF);
  } else if (req.url == '/lang.css') {
    res.writeHead(200, { 'Content-Type': 'text/css' });
    res.end(fileContents.appLangCSS);
  } else if (req.url == '/flyball.pdf') {
    res.writeHead(200, { 'Content-Type': 'application/pdf' });
    res.end(fileContents.appFlyBallPDF);
  } else if (req.url == '/elevator') {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.end(fileContents.appElevator);
  } else if (req.url == '/report_en_elevator.pdf') {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.end(fileContents.appPdfElevator);
  } else {
    res.writeHead(404);
    res.end();
  }
});
// Creating WebSocket servers for internal and external communication
let wssInt = new WebSocket.Server({ server: serverInt });
let wssExt = new WebSocket.Server({ server: serverExt });
// Initialize valuables
let connections = 0;
var ExperimentStatusFb = CommunicationType.NORMAL;
function SendIntConStateToClientsFb() {
  // Scheme for sending data to clients: creating obj --> stringify obj --> send it
  let obj_int_fb = {
    FbIntWSConnection: [ConstantsFB.IntWSConStateFb],
  };
  // Convert the object to a JSON string
  let obj_int_fb_str = JSON.stringify(obj_int_fb);
  // Send the connection state to each Fb client
  clients.clientsExpFb.forEach(function each(client) {
    if (client.readyState === WebSocket.OPEN) {
      client.send(obj_int_fb_str);
    }
  });

  // Log the connection status
  if (ConstantsFB.IntWSConStateFb == ConnectionStatus.NO_CONNECTION) {
    console.log(
```

```
      'No Connection with serialProxy script--> send it to each client'.red
    );
  } else {
    console.log(
      'New Connection with serialProxy script--> send it to each client'.green
    );
  }
}
// Call the function to send the connection state to clients
SendIntConStateToClientsFb();
wssInt.on('connection', function (ws, req) {
  // START Experiment E (Elevator, intern connection)
  // If already connected, the websocket will close.
  try {
    experiments.experimentE.close();
  } catch (e) {
    console.log('Handling: No Websocket to close');
  }
  // New socket is registered
  experiments.experimentE = ws;
  // Log the establishment of an internal Experiment E WebSocket connection
  console.log(
    'Established internal experiment e websocket connection from: ${req.socket.remoteAddress}'
      .green
  );
  // Connection to RaspberryPi successful
  eExperimentConstants.IntWSConStateE = ConnectionStatus.CONNECTION;
  let experimentOnline = 1;
  // Perform self-test on startup to determine Experiment E status
  SelfTstExpE();
  // Event handler for WebSocket message reception
  ws.on('message', function (msg) {
    // When a message is received from experiment, this should be sent to peer.
    // Give the experiment state to the session management and act for case "Defect"
    checkStatusE(msg); // Update Experiment E Status
    SendProxyDataE(); // Broadcast public Experiment Data to external clients
    SendQueuePositionE(); // Send queue position to external clients
    // Parse the received message
    const obj = JSON.parse(msg);
    let experiment = obj.tokenID;
    // Extract relevant data from the message
```

```
    let { tof, pwm, StatusExp } = JSON.parse(msg);
    let t = parseFloat(t);
    // Prepare experiment data message for external clients
    let msgExtClientE = JSON.stringify({
      tof: tof,
      pwm: pwm,
      StatusExp: StatusExp,
      t: t,
    });
    // Pass through experiment data to Master Client
    if (connections > 0 && masterClients.clientExpEMaster != null) {
      if (
        masterClients.clientExpEMaster.readyState === WebSocket.OPEN &&
        masterClients.clientExpEMaster.experiment == obj.tokenID
      ) {
        masterClients.clientExpEMaster.send(msgExtClientE);
      }
    }
  });
  // Event handler for WebSocket connection close
  ws.on('close', function () {
    // Set Experiment E status to offline when WebSocket connection is closed
    let experimentOnline = 0;
    console.log('Disconnected WebSocket of experiment E');
  });
}); // END Experiment E (intern connection)
and a extern path (connection to client)
// Establishes WebSocket connection for external clients
wssExt.on('connection', function (ws, req) {
  connections++;
  console.log('Connected new external WebSocket (Total: ' + connections + ')');
  // Decides what to do depending on requested URL
  if (req.url == '/getCodes') {
    clients.clientsCodes.push(ws);
    // On closed websocket communication --> decrease connection variable
    ws.on('close', function () {
      console.log(`Disconnected WebSocket of Codes`.yellow);
      connections--;
    });
  }
  else if (req.url == '/expG') {
```

```
  // START Experiment G (intern connection)
  // If already connected, the websocket will close.
  try {
    experiments.experimentG.close();
  } catch (e) {
    console.log('Handling: No Websocket to close');
  }
  // New socket is registered
  experiments.experimentG = ws;
  console.log(
    `Registered new Experiment G service from: ${req.socket.remoteAddress}`
      .green
  );
  // When new message arrives, it will sent it to all registered Websockets
  ws.on('message', function (msg) {
    // Check on credentials
    const obj = JSON.parse(msg);
    if (clients.clientsExpG.length > 0) {
      clients.clientsExpG.forEach(function each(client) {
        if (client.readyState === WebSocket.OPEN) {
          client.send(msg);
        }
      });
    }
  });
  // On closed websocket communication should be cut
  ws.on('close', function () {
    console.log(`Disconnected WebSocket of experiment G`.yellow);
    connections--;
  });
} // END Experiment G (intern connection)
else if (req.url == '/expK') {
  // START Experiment K (intern connection)
  // If already connected, the websocket will close.
  try {
    experiments.experimentK.close();
  } catch (e) {
    console.log('Handling: No Websocket to close');
  }
  // New socket is registered
  experiments.experimentK = ws;
```

```
  console.log(
    `Registered new Experiment K service from: ${req.socket.remoteAddress}`
      .green
  );
  // When new message arrives, it will sent it to all registered Websockets
  ws.on('message', function (msg) {
    if (clients.clientsExpK.length > 0) {
      clients.clientsExpK.forEach(function each(client) {
        if (client.readyState === WebSocket.OPEN) {
          client.send(msg);
        }
      });
    }
  });
  // On closed websocket communication should be cut
  ws.on('close', function () {
    console.log(`Disconnected WebSocket of experiment K`.yellow);
    connections--;
  });
} // END Experiment K (intern connection)
else if (req.url == '/expFb') {
  // START Experiment FB (floating ball, intern connection)
  // If already connected, the websocket will close.
  try {
    experiments.experimentFb.close();
  } catch (e) {
    console.log('Handling: No Websocket to close');
  }
  // New socket is registered
  experiments.experimentFb = ws;
  console.log(
    `Registered new Experiment Fb service from: ${req.socket.remoteAddress}`
      .green
  );
  // Connection to serialProxy on RaspberryPi successful --> send it to clients
  ConstantsFB.IntWSConStateFb = ConnectionStatus.CONNECTION;
  SendIntConStateToClientsFb();

  // Wether there is currently a master or not --> send info to serialProxy script
  if (masterClients.clientExpFbMaster != null) {
    ConstantsFB.ClientStatusFb = CommunicationExperiment.MASTER;
```

```
  } else {
    ConstantsFB.ClientStatusFb = CommunicationExperiment.NO_MASTER;
  }
  SendClientStateToExpFb();
  // When new message arrives from intern connection (serialProxy), message will be send to all registered fb-clients
  ws.on('message', function (msg) {
    // Give the experiment state to the session management and act for case "Defect"
    checkStatusFb(msg);
    if (clients.clientsExpFb.length > 0) {
      clients.clientsExpFb.forEach(function each(client) {
        if (client.readyState === WebSocket.OPEN) {
          client.send(msg);
        }
      });
    }
  });
  // On closed websocket communication should be cut
  ws.on('close', function () {
    console.log(`Disconnected WebSocket of experiment Fb`.yellow);
    connections--;
    // Connection to serialProxy on RaspberryPi failed / broken --> send it to Fb-clients
    ConstantsFB.IntWSConStateFb = ConnectionStatus.NO_CONNECTION;
    SendIntConStateToClientsFb();
  });
} // END Experiment Fb (intern connection)
else if (req.url == '/clientExpG') {
  // START Experiment G (extern connection to clients)
  clients.clientsExpG.push(ws); // Push this to be a client in the experiment
  ws.state = SocketState.OPEN; // Sets the initial state
  ws.on('message', function (msg) {
    // On message arrival from client
    switch (this.state) {
      case SocketState.OPEN:
        console.log('Attempting to take control');
        let msgObj = JSON.parse(msg);
        try {
          if (msgObj.SessionCode == sessionCode) {
            // Change state to CONTROL
            this.state = SocketState.CONTROL;
            // Take control of Experiment G
            console.log(`Taken control ExpG`.green);
```

```
          this.send(`{"CONTROL":"true"}`);
          // Check if there is a master client for ExpG
          if (masterClients.clientsExpGMaster != null) {
            // Change state of the master client to OPEN
            masterClients.clientsExpGMaster.state = SocketState.OPEN;
            // Notify master client to release control
            masterClients.clientsExpGMaster.send(`{"CONTROL":"false"}`);
          }
          masterClients.clientsExpGMaster = this;
        } else {
          this.send(`{"CONTROL":"false"}`);
        }
      } catch (e) {
        console.log(e);
      }
      break;
    case SocketState.CONTROL:
      try {
        // Relay message to Experiment G
        if (experiments.experimentG.readyState === WebSocket.OPEN) {
          experiments.experimentG.send(msg);
          console.log(`Relayed message for Exp G`.magenta);
        }
      } catch (e) {
        console.log(
          `Socket attempted to send signals to closed experiment`.red
        );
      }
      break;
  }
 });
 ws.on('close', function () {
  // Connection closed
  let idx = clients.clientsExpG.indexOf(this);
  clients.clientsExpG.splice(idx, 1); // Removes client from list
  connections--;
  console.log('Disconnected WebSocket G (Now: ' + connections + ')');
 });
} // END Experiment G (extern connection to clients)
else if (req.url == '/clientExpK') {
 // START Experiment K (extern connection to clients)
```

```
clients.clientsExpK.push(ws); // Push this to be a client in the experiment
ws.state = SocketState.OPEN; // Sets the initial state
ws.on('message', function (msg) {
  // On message arrival from client
  switch (this.state) {
    case SocketState.OPEN:
      console.log('Attempting to take control');
      let msgObj = JSON.parse(msg);
      try {
        if (msgObj.SessionCode == sessionCode) {
          this.state = SocketState.CONTROL;
          // Take control of Experiment K
          console.log(`Taken control ExpK`.green);
          this.send(`{"CONTROL":"true"}`);
          if (masterClients.clientsExpKMaster != null) {
            masterClients.clientsExpKMaster.state = SocketState.OPEN;
            masterClients.clientsExpKMaster.send(`{"CONTROL":"false"}`);
          }
          masterClients.clientsExpKMaster = this;
        } else {
          this.send(`{"CONTROL":"false"}`);
        }
      } catch (e) {
        console.log(e);
      }
      break;
    case SocketState.CONTROL:
      try {
        if (experiments.experimentK.readyState === WebSocket.OPEN) {
          experiments.experimentK.send(msg);
          console.log(`Relayed message for Exp K`.cyan);
        }
      } catch (e) {
        console.log(
          `Socket attempted to send signals to closed experiment`.red
        );
      }
      break;
  }
});
ws.on('close', function () {
```

```
    // Connection closed
    let idx = clients.clientsExpK.indexOf(this);
    clients.clientsExpG.splice(idx, 1); // Removes client from list
    connections--;
    console.log('Disconnected WebSocket G (Now: ' + connections + ')');
   });
  } // END Experiment K (extern connection to clients)
  else if (req.url == '/clientExpFb') {
   // START Experiment Fb (extern connection to clients)
   clients.clientsExpFb.push(ws); // Push this to be a client in the experiment
   ws.state = SocketState.OPEN; // Sets the initial state (OPEN)
   SendExpStateToClientsFb(); // sends the current experiment state to the new client
   SendIntConStateToClientsFb(); // If serialProxy is connected or not to websocketProxy
   ws.on('message', function (msg) {
    // On message arrival from client --> node.js acts like a "loop" here
    let msgObj = JSON.parse(msg);
    switch (this.state) {
     case SocketState.OPEN:
      console.log('Attempting to take control');
      try {
       // Check if the session code putted in is equal to the generated session code OR the code is in the list of allowed
UIDs
       if (
        msgObj.SessionCode == sessionCode ||
        isUidValid(msgObj.SessionCode)
       ) {
        // If another User is already using the experiment, push the new client into queue
        if (masterClients.clientExpFbMaster != null) {
         clients.clientsMasterQueueExpFb.push(ws);
         this.state = SocketState.QUEUE;
         // Send updated queue infos to fb-clients
         SendQueueNumberFb();
         SendQueuePositionFb();
         this.send(`{"CONTROL":"queue_entered"}`);
         console.log(
          `ExpFb already in use, you are in the queue now`.red
         );
        } else {
         // Only set this new client as master if the experiment does not have an defect or is in testing mode
         if (
          ExperimentStatusFb != CommunicationType.TEST &&
```

```
        ExperimentStatusFb != CommunicationType.DEFECT
      ) {
        this.state = SocketState.CONTROL; // Set this client as master
        this.send(`{"CONTROL":"true"}`); // Send info to the client
        console.log(`Right code ExpFb --> New Master`.green);
        masterClients.clientExpFbMaster = this; // Set this client as master
        // Send updated queue infos to fb-clients
        SendQueueNumberFb();
        SendQueuePositionFb();
        // Init the usage time for new master
        m_global_fb = fbExperimentConstants.MINUTES_INIT_FB;
        s_global_fb = fbExperimentConstants.SECONDS_INIT_FB;
        // Start the Timer for new master
        TimerFb();
        ConstantsFB.ClientStatusFb = CommunicationExperiment.MASTER;
        // There is currently a master --> send it to serialProxy script
        try {
          ResetExpDataFb(); // For every new master, the experiment data has to be reset
          SendClientStateToExpFb();
        } catch (e) {
          console.log(e);
          // Connection to serialProxy on RaspberryPi failed --> send it to Fb-clients
          ConstantsFB.IntWSConStateFb =
            ConnectionStatus.NO_CONNECTION;
          SendIntConStateToClientsFb();
        }
      }
      if (ExperimentStatusFb == CommunicationType.TEST) {
        this.send(`{"CONTROL":"false"}`);
        this.send(`{"ExperimentStatusFb": 2}`);
      }
    }
  } // If session code was not right, the new client does not get the control right
  else {
    this.send(`{"CONTROL":"false"}`);
  }
 } catch (e) {
  console.log(e);
 }
 break;
// State for clients which are in the queue
```

```
case SocketState.QUEUE:
  try {
    // Check if the message from clinet has the item 'QuitMaster' --> Then leave the queue
    if (msgObj.hasOwnProperty('QuitMaster')) {
      if (msgObj.QuitMaster == true) {
        console.log('Client has left the queue by clicking');
        RemoveClientfromQueueFb(this);
        this.state = SocketState.OPEN; // Set this client as normal client (out of queue)
        // send updated queue infos to fb-clients
        SendQueueNumberFb();
        SendQueuePositionFb();
        this.send(`{"CONTROL":"queue_left"}`); // Send new state info it to client
      }
    }
  } catch (e) {
    console.log(e);
  }
  break;
// State for client is master
case SocketState.CONTROL:
  // Check if the message from client has the item 'QuitMaster' --> Then quit the control rights via function
  if (msgObj.hasOwnProperty('QuitMaster')) {
    if (msgObj.QuitMaster == true) {
      console.log('The master has quitted');
      // Reset the timer
      m_global_fb = 0;
      s_global_fb = 0;
      QuitMasterFb();
    }
  }
  // Check if the message from client has the item 'Extend' --> Then extend the usage time (only once)
  else if (msgObj.hasOwnProperty('Extend')) {
    if (msgObj.Extend == true) {
      console.log('The master wants a 2 minute extension');
      ConstantsFB.TimeExtensionFb = true;
    }
  }
  // This means, the normal control data has been send by client --> try to transfer this data to the serialProxy script
  else {
    try {
      if (experiments.experimentFb.readyState === WebSocket.OPEN) {
```

```
            try {
              experiments.experimentFb.send(msg); // Send data to Rasp
            } catch (e) {
              console.log(e);
              // Connection to serialProxy on RaspberryPi failed --> send info to Fb-clients
              ConstantsFB.IntWSConStateFb = ConnectionStatus.NO_CONNECTION;
              SendIntConStateToClientsFb();
            }
          }
        } catch (e) {
          console.log(
            `Socket attempted to send signals to closed experiment`.red
          );
        }
      }
      break;
    }
  });
  ws.on('close', function () {
    // Connection between client and websocketProxy is closed
    let idx = clients.clientsExpFb.indexOf(this); // Get the index of this client in array
    // Remove master
    if (masterClients.clientExpFbMaster == this) {
      console.log('The master is gone');
      QuitMasterFb();
    }
    RemoveClientfromQueueFb(this);
    clients.clientsExpFb.splice(idx, 1); // Removes client from list (the other cells of array are going one place forward)
    connections--;
    console.log('Disconnected WebSocket Fb (Now: ' + connections + ')');
  });
} // END Experiment Fb (extern connection to clients)
else if (req.url == '/clientExpE') {
  // START Experiment E (extern connection to clients)
  clients.clientsExpE.push(ws);
  ws.state = SocketState.OPEN;
  ws.on('message', function (msg) {
    // On message arrival from external client --> node.js acts like a "loop" here
    let msgObj = JSON.parse(msg);
    switch (
      this.state // State of external Client can be OPEN (default), Queue or CONTROL (Master)
```

```
) {
  case SocketState.OPEN:
    try {
      if (msgObj.hasOwnProperty('controlReqE')) {
        // Is triggered on Control Request
        if (masterClients.clientExpEMaster != null) {
          // Push Client to QUEUE when there is a master
          clients.clientsMasterQueueExpE.push(ws);
          this.state = SocketState.QUEUE;
        } else {
          // Set Client to Master when there is no queue
          setMasterE(this);
        }
      }
    } catch (e) {
      console.log(e);
    }
    break;
  case SocketState.QUEUE:
    try {
    } catch (e) {
      console.log(e);
    }
    break;
  case SocketState.CONTROL: // Message from master client
    try {
      if (msgObj.hasOwnProperty('hRef')) {
        // Check if control parameter message is received
        try {
          if (experiments.experimentE.readyState === WebSocket.OPEN) {
            // Check if experiment is online
            try {
              experiments.experimentE.send(msg); // Send Client data to Rasp
            } catch (e) {
              console.log(e);
              // Connection to RaspberryPi failed
              eExperimentConstants.IntWSConStateE =
                ConnectionStatus.NO_CONNECTION;
            }
          }
        } catch (e) {
```

```
            console.log(
              `Socket attempted to send signals to closed experiment E`.red
            );
           }
          }
        } catch (e) {
          console.log(e);
        }
        break;
    }
  });
  ws.on('close', function () {
    // Connection between client and websocketProxy is closed
    let idx = clients.clientsExpE.indexOf(this); // Get the index of this client in array
    // Remove master on disconnect
    if (masterClients.clientExpEMaster == this) {
      console.log('The master of exp E is gone');
      kickMasterE();
    }
    RemoveClientfromQueueE(this);
    clients.clientsExpE.splice(idx, 1); // Removes client from list (the other cells of array are going one place forward)
    connections--;
    console.log('Disconnected WebSocket E (Now: ' + connections + ')'.yellow);
  });
} // END Experiment E (extern connection to clients)
  else if (req.url == '/webcamS') {
  // START webcamS (intern communication side) (Exp G and ExpK webcam)
  ws.on('message', function (msg) {
    webcamImg = msg;
    //send to clients
    clients.clientsWebcam.forEach(function each(client) {
      if (client.readyState === WebSocket.OPEN) {
        client.send(msg);
      }
    });
  });
} // END webcamS (intern communication side)
else if (req.url == '/webcam') {
  // START webcam (extern communication side to clients) (Exp G and ExpK webcam)
  //Register itself
  clients.clientsWebcam.push(ws);
```

```
} // END webcam (extern communication side to clients)
else if (req.url == '/webcamFromFb') {
  // START webcam Fb (intern communication side to webcam)
  console.log('Connected Webcam Floating Ball');
  ws.on('message', function (msg) {
    webcamImg = msg;
    // Send to clients
    clients.clientsWebcamFb.forEach(function each(client) {
      if (client.readyState === WebSocket.OPEN) {
        client.send(msg);
      }
    });
  });
  // On closed websocket communication should be cut
  ws.on('close', function () {
    connections--;
    console.log(`Disconnected WebSocket of webcam Fb`.yellow);
  });
} // END webcam Fb (intern communication side to webcam)
else if (req.url == '/webcamClientFb') {
  // START webcam Fb (extern communication side to clients)
  // Register itself
  clients.clientsWebcamFb.push(ws);
  // On closed websocket communication should be cut
  ws.on('close', function () {
    connections--;
    console.log(
      'Disconnected WebSocket FbWebcamClient (Now: ' + connections + ')'
    );
  });
} // END webcam Fb (extern communication side to clients)
// If no correct websocket url or path has been used.
else {
  console.log(
    `Unauthorized WS from: ${req.socket.remoteAddress} with argument: ${req.url}`
      .red
  );
  connections--;
  // Close this unauthorized connection immediately
  ws.close();
}
```

```
});
// Exporting functions to be accessible from other modules
export { wssExt, SendIntConStateToClientsFb };

serverExt.listen(settings.EXT_PORT);
serverInt.listen(settings.INT_PORT); // Turn this on if intern port is used
```

codes.html

```
<html>
   <head>
      <title>Experiment Codes</title>
   </head>
   <body>
         <h1>Current Code (will be resetted every 30 seconds)</h1>
         <div style="font-size: 800%" id="code"> Wait... </div>
         <progress id="codeTimeout" value="30" max="30"></progress>
         <script type="text/javascript">
        try {
                                    socket = new WebSocket('ws://localhost:8080/getCodes'); //
ws://localhost:8080/getCodes
                                    console.log('WebSocket status '+socket.readyState);
                                    socket.onopen    = function(msg) {
                                    };
                                    socket.onmessage = function(msg) {
                                     console.log(msg.data);
                                     var obj = JSON.parse(msg.data);
                                     document.getElementById("code").innerHTML=obj.code;
                                     document.getElementById("codeTimeout").value=30;
                                    }
                           }catch(e)
                           {
                           }
                           setInterval(function(){
         document.getElementById("codeTimeout").value=document.getElementById("codeTimeout").value-
0.1;},100);
         </script>
         </body>
</html>
```

ДОДАТОК Б

ВІДГУК

Керівника економічного розділу

на кваліфікаційну роботу бакалавра на тему:

«Розробка програмного забезпечення для управління та контролю дослідницьких експериментів в межах проекту Lab4All з використанням мови програмування JavaScript. »

Студента групи 122-20-1 Хайтул Іллі Володимировича

Керівник економічного розділу

доц. каф. ПЕП та ПУ, к.е.н **Л.В. Касьяненко**

ДОДАТОК В

Hochschule Reutlingen • Alteburgstraße 150 • 72762 Reutlingen

TEC Technik

Рецензія

17. April 2024

Oleksandr Balakhontsev
Project Coordinator

T./F +49 7121 271-7139

oleksandr.balakhontsev@reutlingen-university.de

на кваліфікаційну роботу бакалавра на тему «Розробка програмного забезпечення для управління та контролю дослідницьких експериментів в межах проекту Lab4All з використанням мови програмування JavaScript»
ст. гр. 122-20-1 Хайтул Іллі Володимировича.

Представлена кваліфікаційна робота бакалавра присвячена розробці програмного забезпечення для серверної частини лабораторних робіт із віддаленим доступом.

Студент Хайтул І.В. взяв участь у проєкті LAB4All – Лабораторії для всіх: кібер-фізичні лабораторії із відкритим доступом для українських університетів (LAB4All – Open Access Cyber-Physical Laboratories for Ukrainian Universities), що виконується в рамках програми Німецької служби академічних обмінів (DAAD) “Підтримка інтернаціоналізації українських університетів: Формуємо цифрове майбутнє разом”.

В ході виконання кваліфікаційної роботи Ілля Хайтул виконав наступне:

– Провів рефакторинг коду та оптимізував його для підвищення його читабельності та підтримки.

– Доробив та вдосконалив структуру та логіку програмного коду, що сприяло покращенню стабільності програмного продукту та його ефективності.

– Забезпечив зменшення кількості похибок та впровадив більш ефективну та зрозумілу функціональність програмного коду для підвищення стійкості застосунку.

– Оптимізував розуміння коду для інших та майбутніх членів команди шляхом стандартизації форматування та коментування.

Суттєві зауваження до кваліфікаційної роботи відсутні.

В цілому, вважаю що кваліфікаційна робота виконана в повному обсязі та у відповідності до завдання. Робота заслуговує на оцінку **«Відмінно» (95 балів)**, а студент Хайтул І.В. – на присвоєння кваліфікації фахівця з інформаційних технологій.

Доцент каф. Електропривода,
О.В. Балахонцев

Hochschule Reutlingen
Reutlingen University
Alteburgstraße 150 • 72762 Reutlingen • Deutschland • T. +49 (0)7121 271-0 • F. +49 (0)7121 271-1101 • www.reutlingen-university.de

ДОДАТОК Г

Дніпровський університет приймає обмежувальні примітки з наступним формулюванням, яке було погоджено в Центральній екзаменаційній комісії і це може бути використане без додаткового затвердження проректором з навчальної роботи. Обмеження може бути зроблено лише в необов'язковій частині - до розділів або сторінок; доповнення або зміни тексту, що виходять за рамки цього, не допускаються.

Ця дипломна робота базується на внутрішніх та конфіденційних даних Університету Ройтлінген. Ця дипломна робота не може бути доступною для неавторизованих третіх осіб без прямої згоди компанії та автора. Відтворення та публікація дисертації, навіть у вигляді витягів, не дозволяється без спеціального дозволу. Період ембарго закінчується через п'ять років після подання дисертації 26.06.2029.

Угоди про конфіденційність, які виходять за рамки цього, підписуються виключно проректором з навчання.

ДОДАТОК Д

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файла	Опис
Пояснювальні документи	
Кваліфікаційна робота_Хайтул_І_В.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота_Хайтул_І_В.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
DiplomKhaitul.zip	Архів. Містить коди програми і відкомпільовану програму
Презентація	
Презентація_Хайтул_І_В.pptx	Презентація кваліфікаційної роботи