МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет радіоелектроніки

Центр післядипломної освіти
Кафедра Програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Дослідження методів стохастичного програмування для задачі прийняття рішень для системи допомоги у несприятливих умовах для життя

Виконала: студентка 2 курсу, групи ІПЗздм-20-1
Ковальова Н. Ю.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення

Тип програми Освітньо-наукова

Керівник: доц. кафедри ПІ, к. т. н,
доц. Лещинська І. О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Дудар З.В.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Центр післядипломної освіти

Кафедра Програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Ковальовій Надії Юріївні
(прізвище, ім'я, по-батькові)

1. Тема роботи «Дослідження методів стохастичного програмування для задачі прийняття рішень для системи допомоги у несприятливих умовах для життя» затверджена наказом університету від «*24*» *березня* 20*22* р. № *31Стз*

2. Термін подання студентом роботи до екзаменаційної комісії «*14*»*травня* 20*22* р.

3. Вихідні дані до роботи: *електронні ресурси, задача прийняття рішень, методи стохастичного програмування.*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналітичний огляд, постановка задачі, архітектура та проектування, дослідження методів стохастичного програмування.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	17.01.2022	виконано
2	Постановка задачі	31.01.2022	виконано
3	Формування вимог	07.02.2022	виконано
4	Проектування	21.02.2022	виконано
5	Реалізація	07.03.2022	виконано
6	Тестування	05.04.2022	виконано
7	Підготовка пояснювальної записки	01.05.2022	виконано
8	Підготовка презентації та доповіді	02.05.2022	виконано
9	Нормоконтроль, рецензування	06.05.2022	
10	Занесення диплома в електронний архів	14.05.2022	
11	Допуск до захисту у зав. кафедри	16.05.2022	

Дата видачі завдання «__» _____ 2020 р.

Студент _____ Ковальова Н. Ю.

(підпис) (прізвище, ініціали)

Керівник роботи _____ доц. Лещинська І. О.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи магістра: 78 с., 22 рис., 4 додатки, 19 джерел.

ВУГЛЕКИСЛИЙ ГАЗ, ГРАВІТАЦІЯ, ЛІНІЙНЕ ПРОГРАМУВАННЯ, НЕВИЗНАЧЕНІСТЬ ЗНАЧЕНЬ, ПРИЙНЯТТЯ РІШЕНЬ, СТОХАСТИЧНЕ ПРОГРАМУВАННЯ, ТЕМПЕРАТУРА.

Об’єктом дослідження є стохастичне програмування. Мета роботи — аналіз задачі прийняття рішень в умовах ризику шляхом використання методів стохастичного програмування у системі, яка вимірюватиме умови навколишнього середовища за допомогою спеціальних датчиків, аналізуватиме отриману інформацію та при відхиленні показників за допомогою системи прийняття рішень надсилатиме повідомлення із застереженнями. Предметом дослідження є методи стохастичного програмування, задача прийняття рішень, реакція організму людини та її психіки на небезпеку природного характеру.

Explanatory note to the master's qualification work: 78 pages, 22 pictures, 4 adjuncts, 19 sources.

CARBON GAS, DECISION MAKING, GRAVITY, LINEAR PROGRAMMING, STOCHASTIC PROGRAMMING, TEMPERATURE, UNCERTAINTY OF VALUES.

The object of research is stochastic programming. The aim of the work is to analyze the problem of decision-making in conditions of risk by using stochastic programming methods in the system, which will measure environmental conditions with special sensors, analyze the information obtained and when deviating indicators using the decision-making system, and send warning messages. The subject of research: the methods of stochastic programming, the task of decision making, the reaction of the human body and its psyche to natural hazards.

Умови публікації пояснювальної записки

Я, Ковальова Надія Юріївна, студентка групи ІПЗздм-20-1, здобувач вищої освіти на другому (магістерському) рівні, кафедра Програмної інженерії, заявляю: моя кваліфікаційна робота на тему «Дослідження методів стохастичного програмування для задачі прийняття рішень для системи допомоги у несприятливих умовах для життя», що буде представлена до ЕК для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлена з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Сучасний світ має багато протиріч, ми часто стикаємося з проблемою, коли щось втрачає свій пріоритет чи інтерес. Але чи можна це контролювати та запобігти? Свідома діяльність людини нерозривно пов'язана з процесами прийняття рішень. Люди завжди приймали рішення, ґрунтуючись на своєму досвіді, інтуїції та здоровому глузді. При цьому, як правило, точний шлях, що привів до вибору рішення, сам автор рішення писати неспроможна, хоча є підстави вважати, що він якимось чином враховував та зважував усі аспекти прийнятого рішення. Вміння приймати рішення, що дають найкращі результати у різних складних ситуаціях завжди розглядалося як мистецтво.

Стохастичне програмування — розділ математичного програмування, що вивчає теорію, моделі та методи розв’язування умовних екстремальних задач з неповною інформацією про параметри їх умов, які є випадковими величинами. Стохастичне програмування є вигідним для управління та створення додатків через притаманну стохастичну природу процесу. Стохастичне програмування дозволяє нам моделювати невизначеність процесу і дозволити цьому сприяти прийняттю рішень. Крім того, це дозволяє нам розглянути, які оптимальні рішення можуть бути засновані на різних результатах невизначеного процесу. Це дозволить особам, які приймають рішення, розглянути вплив певних факторів і порівняти різні набори рішень з точки зору понесених витрат і потенційних вигод.

Кожна людина прагне себе захистити від небезпеки, адже існує безліч факторів, що негативно пливають на наше життя. Стан довкілля безпосередньо має відношення до стану та здоров'я людини, будь-яке відхилення від норми показників може спричиняти загрозу для життя. Поки організм людини спроможний за допомогою адаптаційних реакцій забезпечити стабільне функціонування, здоров’я людини перебуває в стані безпеки. Якщо ж організм потрапляє в умови, коли інтенсивність дії чинників довкілля перевищує його адаптаційні можливості, то виникає стан небезпеки для життя людини, що загрожує й розладом психічного стану. Стан повітря – один з основних життєво

важливих елементів навколишнього природного середовища, який є необхідною фізичною і біологічною умовою існування людини та джерелом життя не тільки на Землі, а й інших планетах, найвірогіднішими кандидатами на заснування колонії людей в доступному для огляду майбутньому. Авжеж, будь-якій людині буде нелегко пристосуватися до умов, які відрізняються від земних, адже нестача кисню, високий рівень радіації, атмосферного тиску чи інших показників може нести негативний влив на стан здоров'я, тому потреба в дослідженні умов навколишнього середовища є важливою складовою при зміни місця проживання. Якщо порівнювати умови на жарких Меркурії і Венері, холодних зовнішніх планетах і позбавленому атмосфери Місяці й астероїдах, то умови на Марсі є набагато більш придатними для освоєння.

Майбутнім колонізаторам знадобиться система, яка дозволить їм контролювати навколишнє середовище та спеціальні житлові приміщення. Система також зможе обробляти отримані дані, повідомляти, якщо показники не нормалізувалися, і давати поради, які вжити заходи.

Мета роботи — використовуючи методи стохастичного програмування розробити систему, яка вимірюватиме умови навколишнього середовища за допомогою спеціальних датчиків, аналізуватиме отриману інформацію та при відхиленні показників, за допомогою системи прийняття рішень, надсилатиме повідомлення із застереженнями .

В результаті роботи отримаємо систему, яка буде самостійно обробляти та контролювати значення показників навколишнього середовища. Крім того, розроблені рекомендації щодо заходів по збереженню життя знизять ризик загрози клімату на життя людини, а збір статистики допоможе колонізаторам в аналізі глобальних екологічних проблем на планетах для майбутньої колонізації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі прийняття рішень

У наш час теорію прийняття рішень (ТПР) застосовують переважно для аналізу тих проблем, які можна відносно легко й однозначно формалізувати, а результати досліджень – адекватно інтерпретувати. Методи ТПР використовують у різних галузях управління: проектуванні складних технічних і організаційних систем, плануванні розвитку міст, доборі програм розвитку економіки й енергетики регіонів, організації нових економічних зон тощо. Потреба в застосуванні засобів і методів ТПР в управлінні очевидна: швидкий розвиток і ускладнення економічних зв'язків, виявлення залежностей між окремими складними процесами та явищами, які раніше здавалися не пов'язаними один з одним, призводять до різкого зростання труднощів під час прийняття обґрунтованих рішень. Витрати на прийняття рішень зростають, наслідки помилок стають усе серйознішими, а звернення до фахового досвіду та інтуїції не завжди зумовлює вибір найкращої стратегії. Застосування методів ТПР дає змогу розв'язати цю проблему, до того ж швидко й достатньо точно і ефективно.

Кожна людина знайома із процесом прийняття рішень та у зв'язку з обставинами іноді виникають проблеми з їх прийняття. Деякі з них прості, інші ж більш складні й потребують ретельного обдумування. Тому науково-обґрунтоване рішення й ряд порад в системі буде стимулювати користувачів вжити вірний захід.

Під процесом прийняття рішення (ППР) розуміють послідовність процедур, що приводять до знаходження рішення. ППР складається з декількох основних етапів. Різні автори з різним ступенем деталізації розглядають послідовність етапів, але в загальному зберігається наступна послідовність дій:

– виявлення проблемної ситуації та постановка задачі прийняття рішення;

– формулювання поняття якості рішення та його структуризація до рівня критеріїв;

– описання характеристик зовнішнього середовища, прогнозування можливих результатів дій ППР із подальшим виявленням або конструюванням альтернативних варіантів рішень;

– оцінювання якості варіантів рішень, порівняння їх між собою та вибір одного чи декількох найвідповідніших меті;

– аналіз рішень, опрацювання плану реалізації та впровадження рішення.

Область стохастичного програмування займається оптимізацією в умовах невизначеності. Як випливає з назви, його підходи до моделювання та алгоритмічні прийоми успадковані від математичного програмування, що відокремлює його від суміжних областей аналізу рішень, теорії стохастичного керування та процесів прийняття рішень Маркова[1]. Хоча математичне програмування користується великою популярністю та широко використовується, невизначеність може бути оброблена тільки за допомогою чутливості або параметричного аналізу. Стохастичне програмування долає цей недолік шляхом явного включення невизначеності в математичне програмування.

Стохастне програмування має справу з класом оптимізаційних моделей і алгоритмів, у яких деякі дані можуть бути предметом значної невизначеності. Такі моделі доречні, коли дані розвиваються з часом, і рішення потрібно приймати до того, як спостерігати за всім потоком даних. Наприклад, колонізатори повинні планувати життя на Марсі до того, як буде реалізовано попит на подорож до нього. Така притаманна невизначеність посилюється технологічними інноваціями. Таке програмування здатне враховувати всі різні потенційні результати невизначеного компонента під час пошуку найкращого набору рішень в цілому. Стохастична програма розглядатиме всі можливі комбінації результатів дослідження та повертає набір початкових досліджень для виконання разом із тими, які слід виконувати у разі різних потенційних результатів дослідження.

1.2 Виявлення проблем та актуалізація рішень

Дуже актуальною є проблема стресового стану в людини, коли вона знаходиться в умовах ризику й складність прийняття рішення зростає. Це звичайна реакція будь-якої людини, але це може призвести до помилкових дій, які прийняті швидко й в майбутньому можуть нашкодити здоров'ю чи життю.

Для жителів нетипових умов навколишнього середовища, існує проблема управління погодними умовами задля збереження життя людей. Тому виникає необхідність аналіз екологічних умов планети Марс та спостереження за ними.

Сьогодні кожна третя доросла людина дуже чутливо сприймає будь-які погодні зміни. Дослідження граничних значень показників та їх вплив на людину дає змогу проаналізувати їх важливість та встановити ідеальні умови для життя.

1.2.1 Вплив температури на організм людини

Температура повітря відіграє важливу роль, адже при впливі на організм високих температур повітря може наступити перегрів організму. У цих умовах відбувається розширення кровоносних судин, зниження серцевого тонусу й артеріального тиску, частішання пульсу, збільшення температури шкіри. Більш значний перегрів може призвести до важких патологічних явищ з боку серцево-судинної системи і загального стану організму (теплового удару). Тепловий удар настає раптово і часто призводить до важких ускладнень; найбільш характерний симптом – дуже висока температура. Вплив на організм низьких температур викликає підвищення обміну речовин і звуження периферичних кровоносних судин, що призводить до зниження температури шкіри. Швидкість кровотоку при цьому знижується, а теплопровідність шкіри і поверхневих тканин зменшується в 6 – 7 разів. Артеріальний тиск при низькій температурі повітря має тенденцію до підвищення (особливо при м'язовому тремтінні)[2].

1.2.2 Вплив вологості на організм людини

Не менш важливою є вологість повітря, яка як біокліматичний фактор впливає по-різному. По-перше, від неї істотно залежить парціальна густина кисню в повітрі (при зменшенні парціального тиску водяної пари зростає парціальна густина кисню за інших рівних умов); по-друге, вологість повітря впливає на радіаційні умови (через утворення хмарності); по-третє, від неї залежить утрата рідини в організмі[3]. У метеорологічних прогнозах, зазвичай, вказується відносна вологість, бо її зміна може безпосередньо відчуватися людиною. Повітря вважається сухим при вологості до 55%, помірно-сухим – при 56-70%, вологим – при 71-85%, дуже вологим (сирим) – вище 85% (по І. В.Бутьєвій). Зона комфорту по вологості повітря для практично здорових людей коливається від 45 до 80%. За умов посухи, коли вологість повітря не перевищує 30%, різко збільшується вологовіддача з боку організму. При відносній вологості > 80%, випаровування утруднено, відчуття жари і холоду більш неприємне. Хворі гіпертонічною хворобою і коронарним атеросклерозом дуже чутливі до коливань відносної вологості повітря. У таких хворих переважна більшість приступів настає при відносній вологості 80-95%.

1.2.3 Вплив вуглекислого газу на організм людини

CO_2 є одним з основоположних сполук для правильного функціонування нашого організму. Однак перевищення концентрації вуглекислого газу може привести до плачевних наслідків і суттєво нашкодити здоров'ю людини. Максимально допустимим рівнем вуглекислого газу можна вважати значення до 0,15%, однак, такі значення далеко не відповідають дослідженням, в яких наголошується, що вже при досягненні 0,1% CO_2, у людей починаються проблеми зі здоров'ям. Можна виділити наступні градації:300-400 ppm- нешкідливий рівень концентрації, 400-600 ppm- норма для якісної роботи на підприємстві, в офісі, вдома, оптимальний рівень для місць, де часто перебувають діти, 600-1000 ppm- уже помітно зниження концентрації, особливо, якщо людина чутлива до рівня CO2, 1000-1500 ppm - допустимий, хоч і має ряд

серйозних наслідків показник. Не рекомендується часте перебування в подібного роду приміщеннях, більше 1500 ppm- критичний рівень, при якому можуть утворитися довгострокові порушення розумової, серцево-судинної та інших систем.

1.2.4 Вплив атмосферного тиску на організм людини

Вплив атмосферного тиску на здоров’я людини неминучий, особливо це відображається на самопочутті тих, хто страждає захворюваннями серцево-судинної системи. Люди відчувають дискомфорт від змін погодних умов і магнітних бур. Для комфортного самопочуття людини, необхідно, щоб атмосферний тиск становив 750 мм.рт.ст. При високому і низькому атмосферному тиску, коли значення змінюється більш ніж на 10 одиниць як у більшу, так і у меншу сторону, з‘являється головний біль. Різкі перепади чинять негативний вплив на вміст кисню в крові. Серцево-судинна система починає гальмувати в процесі синхронізації кров’яного тиску з атмосферним зовнішнім тиском. Залежно від перепаду атмосферного тиску розширюються або звужуються кровоносні судини, викликаючи головний біль і слабкість.

1.2.5 Вплив гравітації на організм людини

Гравітація також вливає на життя та здоров’я людини. Скелет людини починає ламатися під вагою тіла при гравітації, яка вдесятеро перевищує земну. Також вчені визначили граничні умови, після яких навіть дуже розвинена мускулатура не дозволить людині встати. Вони відповідали п'яти земним гравітаціям. Окремо вчені оцінили, скільки енергії людина витрачала б на пересування з урахуванням можливостей найсильніших людей. Виявилося, що гравітація в 4,6 рази більша за земну не дозволить ходити навіть таким людям. Невагомість – пригнічує імунну систему. Зазвичай організм людини при появі певного типу вірусу, активує 99 генів, які в свою чергу, активують захисні Т-клітини – білі клітини крові, щоб знищити вірус. Однак в умовах невагомості активується тільки 8 генів з 99 і захисні функції організму ослаблюються.

Найзначніші негативні впливи невагомості:

– атрофія м’язів, зниження м’язової та кісткової маси;

– перерозподіл рідини в тілі ;

– уповільнення серцево-судинної системи;

– зменшення виробництва еритроцитів;

– порушення рівноваги та ослаблення імунної системи;

– виявлені деформації очного яблука, зорового нерва гіпофіза.

Як можна побачити, наш організм дуже вимогливий до показників, які складно досягти в наших реаліях, але їх можна контролювати для забезпечення комфортного життя.

Атмосфера Марса дуже тонка і складається в основному з вуглекислого газу, а поверхня є абсолютно пустельною. Вода, що знаходиться на Марсі дуже швидко закипає, адже на планеті дуже низький атмосферний тиск. Також Марс не має сильного магнітного поля, тому рівень радіації на його поверхні вище від Земного. Рівні на Марсі дуже високі, що перевищують максимальну дозу для життя. Таке середовище, ймовірно, буде шкідливим для здоров’я людини при перспективі життя на планеті та може мати негативний вплив на народжуваність людини.

Але, незважаючи на перелічені проблеми, існує ідея, що людство скоро зможе з легкістю створити колонії на Марсі. Головною ціллю життя на Марсі є створення комфортних умов для існування та збереження здоров’я людських колоній. Адже тяжкі умови для життя – важке випробування для багатьох людей. Психічна реакція людини на екстремальні умови, особливо у випадках значних для загрози загибелі людей, може надовго позбавити людину здібності до раціональних вчинків та діям. Відчуття небезпеки може перетворюватися на почуття приреченості, робить людину абсолютно безпорадною, розгубленою і нездатною до цілеспрямованих дій, у тому числі і до активного захисту.

Марс є найкращим рішенням для створення колонізації у Сонячній системі, оскільки він має найбільший потенціал для самодостатності. Задля усунення негативних наслідків життя на Марсі NASA та інші космічні агенції

вже зараз активно працюють над створенням і тестуванням контрзаходів. Інженери у всьому світі розробляють технології посадки та життєзабезпечення, тому вже в найближчому часі ймовірні перші кроки людини на Марсі.

Колонізаторам цієї планети було б комфортніше перебувати в спеціальних помешканнях, в яких зручно стежити за окремими показниками ззовні та зсередини, де влаштована система може повідомляти про конкретну небезпеку та давати рекомендації щодо застосування необхідних заходів у разі відхилень пристрою від норми. Це допоможе при прийнятті рішень й стане підґрунтям правильно вжитих заходів.

Так як сама ідея проживання на Марсі є ризикованою, то використання подібних технологій може викликати наступні труднощі:

– незначна кількість колонізаторів або їх відсутність на перших етапах проекту колонізації, наприклад, «Mars One» [4];
– проблеми реалізації ідеї колонізації Марсу;
– проблеми підключення мережі Інтернет на планеті.

1.3 Огляд існуючих аналогів

Незважаючи на те, що завдання контролю умов навколишнього середовища у несприятливій місцевості є досить актуальним та поширеним, але не менш важливим є психічний стан людини, що також враховується під час створення застосунку. Існує небагато відомих систем-аналогів.

Аналогом системи, яка має на меті допомогти людям використовувати технології, щоб жити психічно здоровим життям, є застосунок eCBT (див. рис. 1.1).

Рисунок 1.1 – Логотип застосунку eCBT

У застосунку реалізовано кілька стратегій та методів, що використовуються у когнітивно-поведінковій терапії. eCBT calm допоможе оцінити рівень стресу, потренувати уважність та вміння розслаблятися при тривожному стані, а також навчить співвідносити думки зі своїми почуттями та поведінкою. В кінцевому результаті ви станете спокійнішим у повсякденному житті і краще зрозумієте причини своїх емоцій та вчинків. Існують три версії програми, кожна з яких призначена для певного виду психічних розладів: Mood (афективні розлади), Calm (тривожні розлади), Trauma (посттравматичний стресовий розлад).

Іншим аналогом є MoodKit — це єдиний у своєму роді застосунок, розроблений, щоб допомогти вам застосувати ефективні стратегії професійної психології у вашому повсякденному житті (див. рис. 1.2).

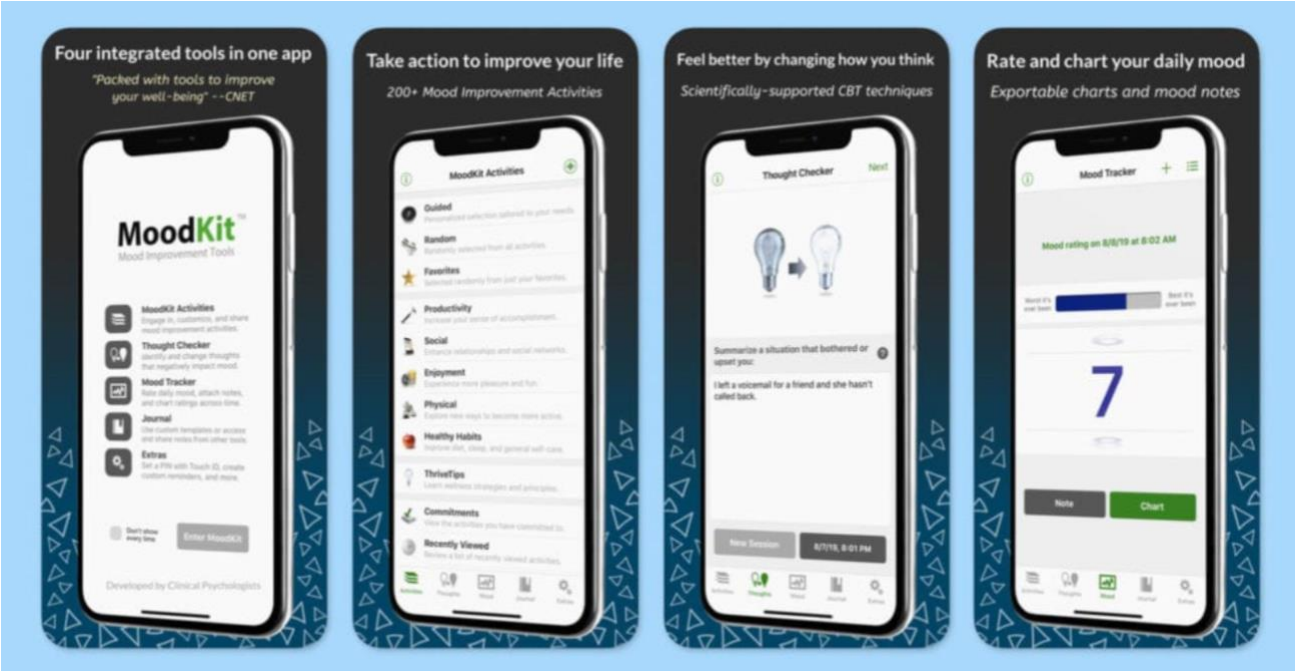


Рисунок 1.2 – Застосунок MoodKit

Інструмент MoodKit Activities надає широкий спектр пропозицій щодо конкретних кроків, які можна зробити, щоб покращити свій настрій. Заходи разом із супровідними прикладами та порадами нададуть вам конкретні способи застосування психологічних принципів і прийомів, які, як відомо, зменшують

негативні почуття та покращують самопочуття. Перевірка думок проведе вас через короткий процес визначення ситуацій, в яких ви відчуваєте страждання, уточнює почуття та думки, які ви відчували в цих ситуаціях, визначає, чи ці думки представляють загальні моделі упередженого чи спотвореного мислення, змінює таке мислення та оцінює вплив на ваші почуття.

Наступні аналоги відмінні від попередніх, вони розроблені для контролю та керування показниками та не розраховані на допомогу людині в критичних ситуаціях чи відхилень від норми значень встановлених в систему датчиків.

Система UbiBot пропонує новітній спосіб контролювати умови навколишнього середовища там, де вони потрібні. Уся синхронізація з платформою UbiBot IoT відбувається за допомогою WiFi. Дані доступні через смартфон чи Інтернет. Принцип роботи Ubibot схожий на принцип роботи нашої системи та зображений на рисунку 1.3.

Рисунок 1.3 – Принцип роботи системи Ubibot

UbiBot GS1 призначений для промислового використання і може бути підключений до мережі та мобільних даних за допомогою кабелю Ethernet RJ45. Показники навколишнього середовища вимірюються 24/7 у заданому темпі. Потім він завантажує значення та зберігає їх в приватних каналах на платформі UbiBot. Тому користувач може переглядати показання в зручний для нього час та в будь-якому місці. У системі є датчики температури, вологості повітря, рівня вуглекислого газу в повітрі, швидкості вітру, освітленості, кислотності середовища, температури ґрунту (див. рис. 1.4).

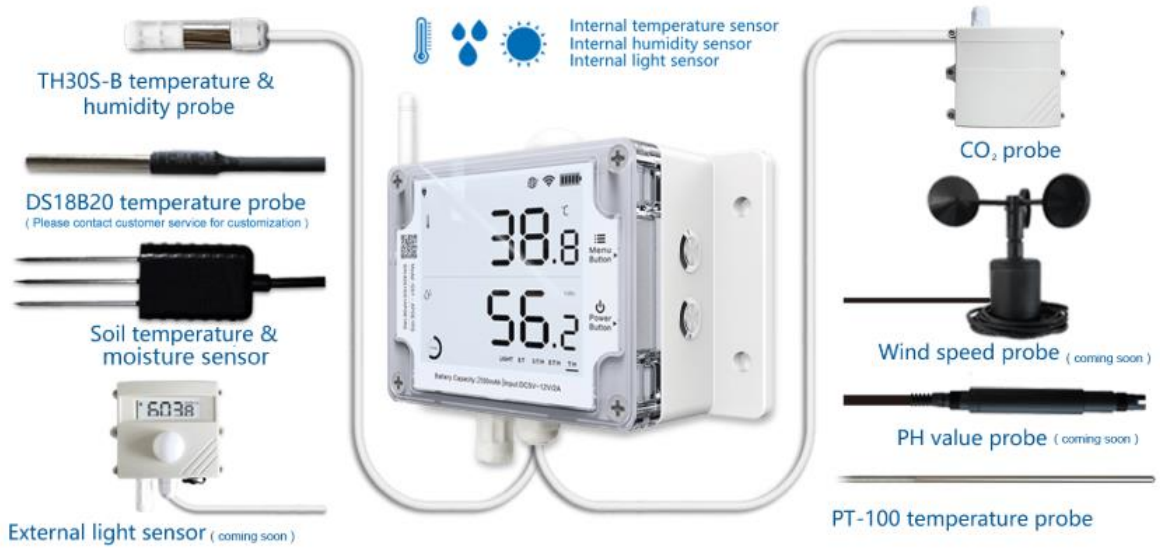


Рисунок 1.4 – Складові пристрою UbiBot GS1

До аналогів, що грунтуються на вимірюванні показників у несприятливих умовах та підтримці життєдіяльності людини, відноситься система контролю умов у апараті Deepsea Challenger (DCV1) (див. рис. 1.5).

Deepsea Challenger (DCV 1) - батискаф, на якому канадський режисер Джеймс Кемерон самостійно 26 березня 2012 року здійснив занурення на дно в «Безодню Челленджера» (Маріанська западина).

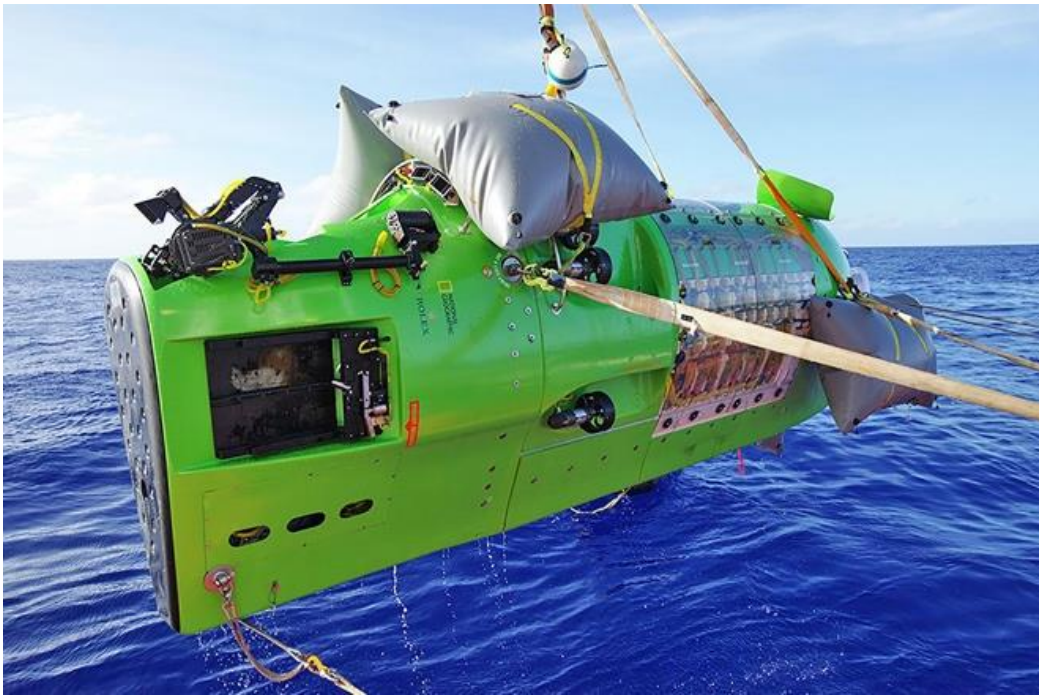


Рисунок 1.5 – Апарат Deepsea Challenger (DCV1)

Цей підводний апарат побудований в Сіднеї, Австралія, проектно-дослідницькою компанією Acheron Project Pty Ltd і містить наукове обладнання: системи життєзабезпечення, прилади моніторингу та контролю, акумулятори, 3D-камери і світлодіодне освітлення. Щоб електроніка піддавалася впливу зовнішнього тиску та не приходила в контакт з морською водою, її занурили у силіконове масло. Періодично вимірювані дані відправляються на корабель супроводу для оцінки ситуації і в разі необхідності для повернення апарату на поверхню.

1.4 Постановка задачі

Проектування та створення системи допомоги у несприятливих умовах навколишнього середовища є складним та відповідальним завданням, адже потребує досліджень сприятливих умов для життя людей та точного вимірювання даних, хоча можливе включення невизначеності, та існує багато непередбачуваних ризиків. До того ж необхідно вирішити задачу прийняття рішень, щоб людині не потрібно було думати над цим питанням та вагатися у разі небезпеки. В галузі задач прийняття рішень існує ряд актуальних проблем. Нижче наведені основні проблеми:

- збирання інформації. Визначення ненормованих результатів вимірювання показників та ознайомлення з алгоритмом роботи всіх датчиків для визначення ефективності розв’язання проблеми.
- впровадження. Прийняття рішень є головною складовою в системі. Процес може бути простим та стрімким у виконанні, чи навпаки, але для будь-якої проблеми, якщо рішення буде розроблено, то що-небудь, звичайно, відбудеться.
- оцінювання наслідків прийнятих рішень. Перевірка виконання та виправлення дій, які були реалізовані.

На основі виконаного аналізу були сформульовані наступні задачі:

- аналіз алгоритму методів стохастичного програмування;
- аналіз роботи датчиків вологості повітря, температури повітря,

атмосферного тиску, визначення вуглекислого газу у повітрі, прискорення вільного падіння;

- розробка алгоритму визначення результатів вимірювання показників, що не в нормі;
- розробка спеціальних порад для уникнення небезпеки;
- розробка апаратного рішення (розробка прототипу системи, збереження і перевірка даних, що передаються);
- розробка простого та зрозумілого інтерфейсу, що відображає основну інформацію результату вимірювання обраного показника та відображення поради, якщо результат не в нормі;
- тестування роботи;
- використовуючи розроблену програмну систему, вирішити задачу прийняття рішень;
- зробити висновки.

Метою даної кваліфікаційної роботи є дослідження методів стохастичного програмування у задачі прийняття рішень для системи допомоги у несприятливих умовах для життя на прикладі планети Марс.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Програмна система допомоги у несприятливих умовах для життя здійснюватиме вимірювання показників стану довкілля за допомогою цифрових датчиків, виконуватиме аналіз отриманої інформації для прийняття рішення, повідомлятиме про результати та буде давати рекомендації зі збереження життя у разі відхилень від норми.

Функціональні вимоги до реалізації програмної системи [5] наступні:

- перегляд отриманих даних про значення атмосферного тиску, температури, вологості, рівню вуглекислого газу в повітрі, гравітації на Web-платформі та мобільній платформі з вказаними координатами вимірювання, датою та часом;
- ведення статистики для кожного користувача. Результати показників збираються, аналізуються та виводяться у вигляді графіків у застосунку;
- реалізація алгоритму прийняття рішення;
- отримання повідомлень про перевищення чи недостачу до допустимої норми вимірювання та отримання поради щодо вжиття заходів для збереження власного здоров’я та колег;
- адміністрування системи: адміністратор може змінювати значення відхилень від норми для різних рівнів загрози для показників та змінювати текст порад.

Системні вимоги до реалізації програмної системи для усіх частин продукту наступні:

- серверна частина застосунку має бути розроблена на платформі Node.js з використанням фреймворка Express.js. Express є мінімалістичним і гнучким веб-фреймворком для серверу Node.js, що надає багато функцій для усіх видів застосунків. Дані системи повинні зберігатися в базі даних MySQL;
- веб-застосунок має бути створений за допомогою бібліотеки React.js, що дозволяє створювати інтерфейси з декларативних компонентів. Для

маніпулювання даними використовується MobX бібліотека, яка є контейнером станів для застосунків JavaScript;

- мобільний застосунок має бути створений за допомогою технології Java Android. Бібліотека Retrofit 2 використовується для відправлення запитів до API;
- IoT-пристрій має бути створений на платформі NodeMCU(v3) з WI-FI модулем ESP8266. Використати модуль 3-осьового акселерометра MPU6050, цифровий датчик температури і вологості DHT22, датчик атмосферного тиску BMP180, датчик вуглекислого газу MQ135 у якості датчиків вимірювання. У якості середовища розробки скриптів обрати ESPlorer, а мову програмування – Lua. Використати бібліотеку http для відправлення запитів до API.

Програмна система має дотримуватися наступних обмежень:

- користувачі та IoT-пристрої повинні мати стабільне підключення до Інтернету;
- користування системою можливе тільки зареєстрованим користувачем;
- IoT-пристрій працює лише за наявності електроенергії;
- кожна окрема система потребує персональну інфраструктуру (IoT-станцію та сервер);
- мобільний застосунок працює лише на операційній системі Android.

У базі даних програмної системи не повинна проводитися процедура очищення, інші пристрої не повинні мати доступ до даних кожного IoT-пристрою та користувачів .

У режимі роботи системи не повинно бути простою, користувач не повинен очікувати відгуку системи більше 5 секунд.

У системі повинна бути українська та англійська мови локалізації для інтерфейсу користувача, проте у базі даних інформація має бути збережена англійською мовою.

Інтерфейс мобільного застосунку повинен підтримуватися з боку як старих, так і новітніх мобільних телефонів з операційною системою Android.

Інтерфейс веб-застосунку повинен підтримуватися з боку як старих, так і новітніх версій таких популярних браузерів, як Google Chrome, Safari, Opera, Mozilla Firefox.

3 ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 UML проектування

Діаграми Use Case та розгортання системи були використані у процесі UML проектування програмної системи.

Use Case - це діаграма, яка демонструє взаємодію різноманітних користувачів з системою та їх взаємозв'язок з різноманітним використанням системи [6].

Таким чином для демонстрації взаємодії IoT-пристрою з системою була спроектована діаграма UseCase IoT-пристрою (рис. 3.1).

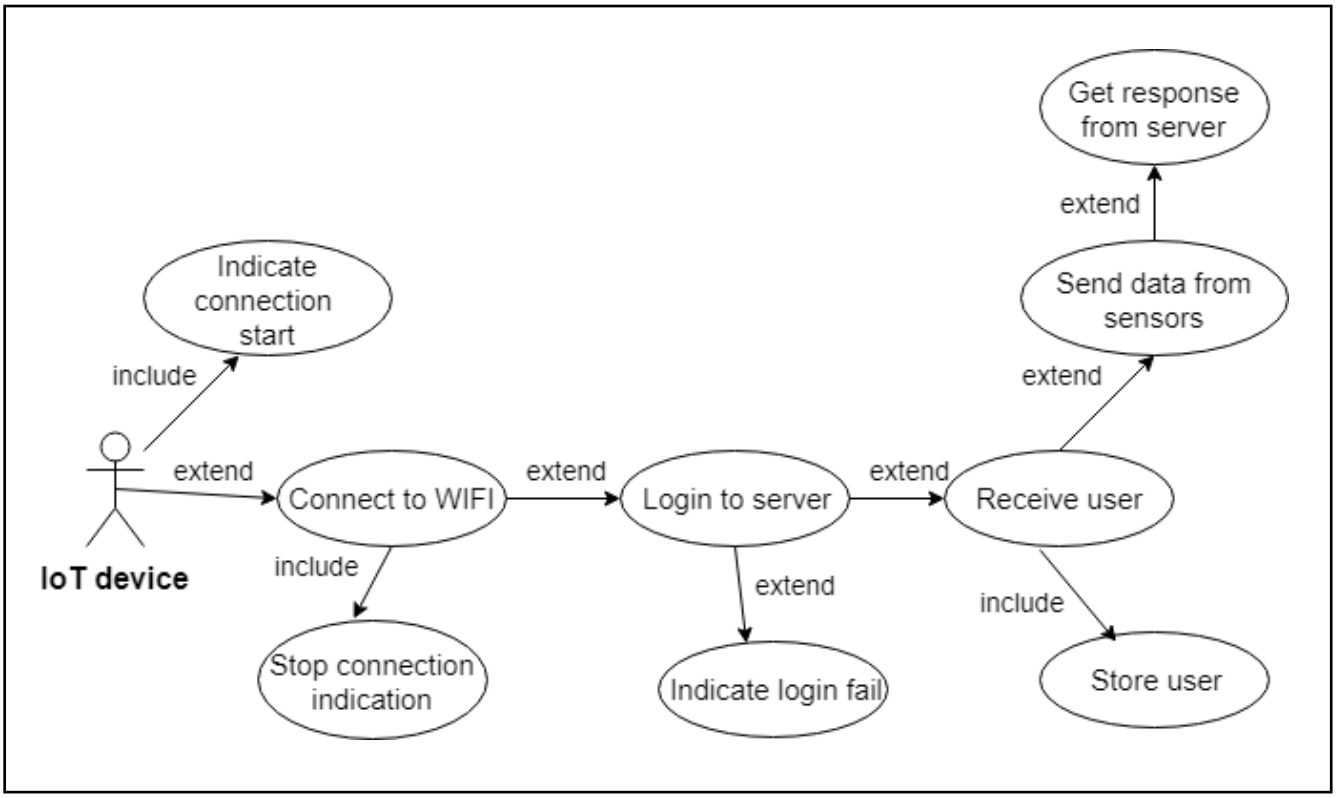


Рисунок 3.1 – Діаграма UseCase IoT-пристрою

Use Case демонструє взаємодію у вигляді система-користувач та класифікує користувачів системи та їх різноманітні випадки використання і моделює основний потік подій у певний час використання.

На діаграмах Use Case веб-застосунку (рис. А.1 додатку А) та мобільного застосунку (рис. 3.2) зображені основні дії незареєстрованих користувачів, зареєстрованих користувачів та адміністратора.

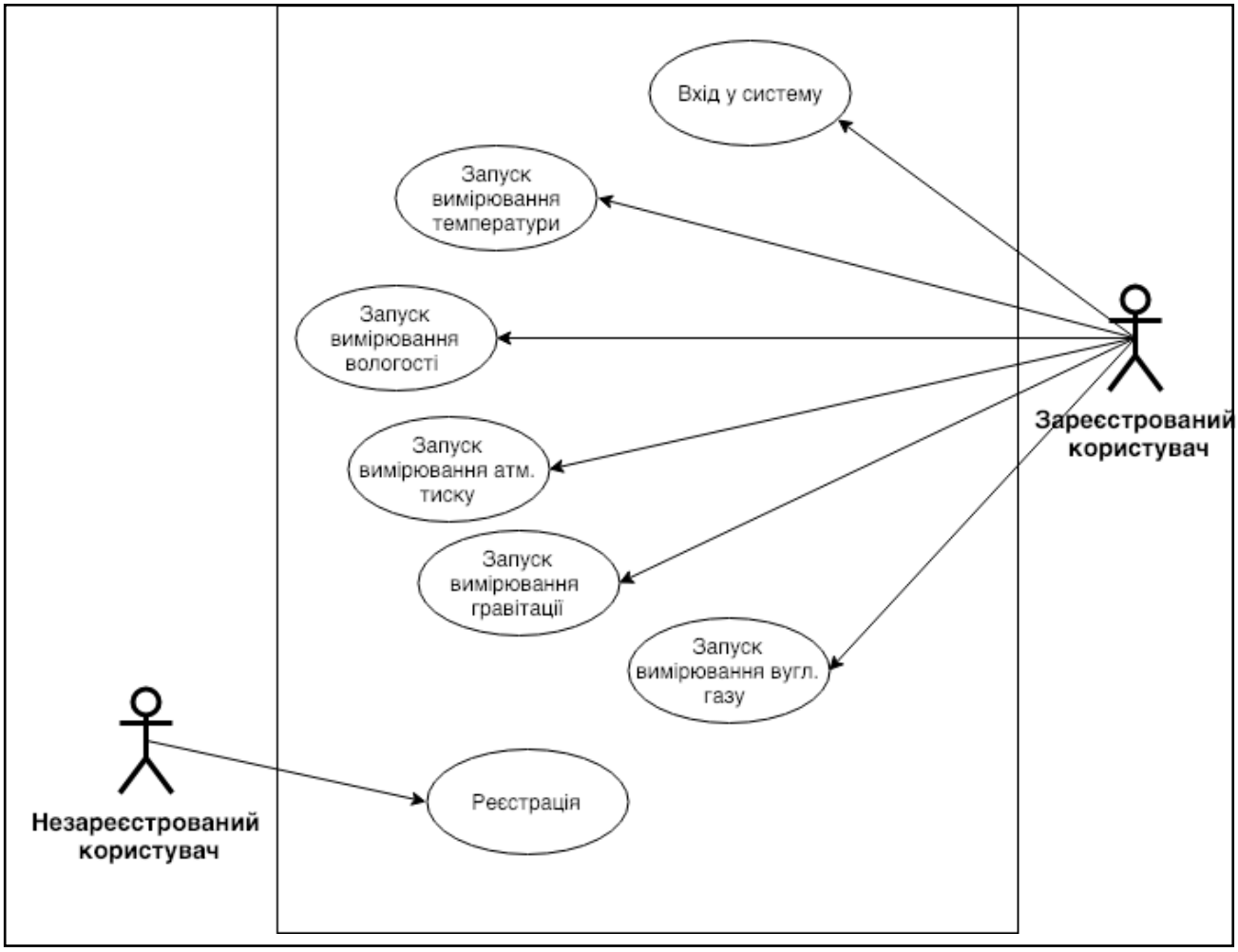


Рисунок 3.2 – Діаграма UseCase мобільного застосунку

Діаграма розгортання демонструє будову системи, у вузлах якої знаходяться розподілені компоненти системи та з'єднання. У свою чергу з'єднання виступають як маршрути для передачі даних між апаратним вузлам [7].

Основним призначенням діаграм розгортання є візуалізація частин програми, що існують під час виконання. Даний вид діаграми розгортання програмної системи містить графічні зображення пристроїв, процесів і процесорів, а також їх зв'язків та зображений на рисунку А.2 додатка А. Цей вид диграми розгортання є єдиним місцем для системи, де демонструються особливості реалізації загалом на відміну від діаграми логічного уявлення.

3.2 Проектування архітектури

3.2.1 Загальна характеристика архітектури системи

Продумана архітектура важлива для будь-яких проектів, незважаючи на їх розмір та важливість. Про архітектуру проекту треба дбати заздалегідь, адже при її відсутності систему буде неможливо підтримувати і розширювати. Дуже часто саме від правильної побудови архітектури залежить успіх проекту, адже вона економить зусилля, час і гроші.

Архітектура програмної системи містить п’ять елементів: API, середовища хостингу, інтерфейс користувача, бізнес-логіку та моделювання даних. Вирішено використовувати архітектуру клієнт-сервер, оскільки цей тип архітектури є найпопулярнішим рішенням для програмних систем з інтерфейсом прикладного програмування, що використовує базу даних для збереження інформації [8]. Схема архітектури програмної системи показана на рисунку 3.3. Цей тип архітектури є одним із видів архітектурних моделей програмного забезпечення, що є домінуючою концепцією при створенні розподілених мережевих додатків і забезпечує взаємодію та обмін даними між ними.

Система клієнт-сервер має такі переваги:

- дані централізовані в системі, яка зберігається в одному місці;
- модель ефективна в доставці ресурсів клієнту, а також вимагає недорогого обслуговування;
- нею легко керувати, а дані можна легко доставити клієнту;
- оскільки дані централізовані, ця система є більш безпечною та забезпечує додаткову безпеку даних;
- у рамках цього типу моделі більше клієнтів і серверів можна вбудувати в сервер, що робить продуктивність надзвичайною та підвищує загальну гнучкість моделі.

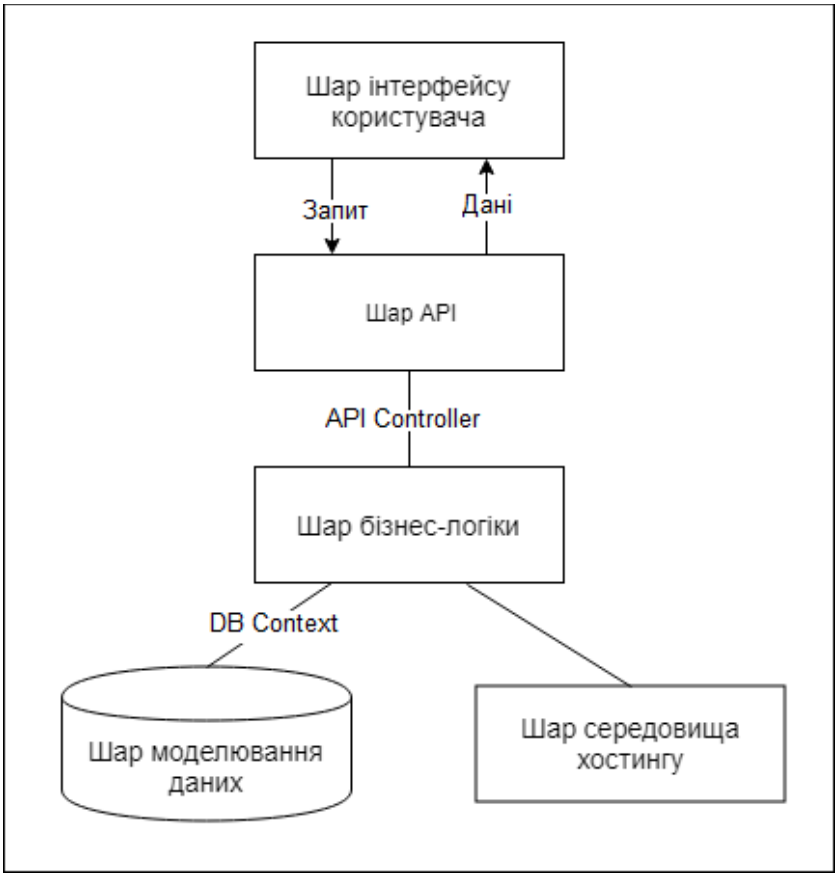


Рисунок 3.3 – Схема архітектури системи

Головним шаблоном архітектурного проектування було взято MVC. Він поділяє розподіл даних програми,які є необхідні для інтерфейсу користувача та керівної логіки на три окремих компоненти:

- вид – надає взаємозв’язок клієнта з поданням. Складники представлення передають показники клієнтові у комфортному вигляді та посилають роботу контролера для керування показниками ;
- модель – окреслює дані для програми (як правило, вони знаходяться у базі даних);
- контролер – керує складниками, одержує сигнали через роботу користувача (видозміна положення курсору миші, нажаття кнопки, введення показників у текстове поле) та віддає дані у модель.

3.2.2 Рівень середовища хостингу

Локальна машина є середовищем хостингу в процесі створення системи. Для API в системі було застосовано таку платформу хостингу як Heroku [9], для бази даних – db4free. Для розгортання веб-застосунку було використано статичне сховище сайтів, а для мобільного застосунку – Google Play.

У перший рік користування система має підтримувати до 5000 клієнтів. Щоб досягнути цієї цифри потрібно проводити масштабування системи. Для кращої продуктивності системи будуть використовуватися підвищені потужності сервера, для легкого та швидкого доступу до даних будуть застосовані декілька баз даних.

3.2.3 Рівень моделювання даних

Системні дані зберігаються в базі даних під керуванням сервера MySQL 8.0.Server. Від серверної частини дані збережені окремо. Розміщенням бази даних є db4free [10] - сервіс для тестування баз даних MySQL. До даних db4free.net доступ здійснюється за допомогою новітньої версії phpMyAdmin 4.8.4 [11].

Рядок підключення до бази даних використовується для доступу до бази даних із серверної частини. За допомогою SQL-авторизації здійснюється вхід до бази даних.

Взаємозв’язок рівня бізнес-логіки з рівнем моделювання даних застосовується контекст бази даних, який для отримання необхідної інформації звертається до таблиць бази даних, а після їх отримання повертає назад серверної частини бізнес-логіки.

3.2.4 Рівень бізнес-логіки

Головними пунктами обробки даних та алгоритмами системи є:

– одержання порад якщо буде спостерігатися відхилення показників вимірювальних датчиків від заданої норми для кожного клієнта;

- одержання показників із бази даних, передання до контролерів API після їх обробки;
- авторизація завдяки електронної пошти та паролю клієнта;
- хешування паролів клієнтів;
- створення токенів авторизації.

3.2.5 Рівень прикладного програмного інтерфейсу (API)

REST-запити [12] використовуються для взаємодії з клієнтськими застосунками: щоб одержати інформацію – GET-запити, щоб надіслати – POST-запити. Від сервера до клієнта інформація надсилається у форматі JSON.

Після авторизації користувачу потрібно надати дані для входу, щоб сервер згенерував токен доступу даного користувача,

За допомогою контролерів APIController API взаємодіє з сервером. Потім дані надходять до контролера, взаємодіючи з рівнем бізнес-логіки, і надсилає дані клієнту.

3.2.6 Рівень інтерфейсу користувача

Для користувачів система представлена у вигляді веб-застосунку, мобільного застосунку для Android та IoT-пристрою для вимірювання та обробки показників рівню вуглекислого газу у повітрі, вологості, температури, гравітації та атмосферного тиску.

Через відправлення REST-запитів та отримання показників з сервера, мобільний та веб-застосунок взаємодіють з сервером . Коли є помилки, застосунок одержує відповідне сповіщення.

Тільки при відправленні запитів до API IoT-пристрій співдієдіє з сервером. Одержані значення виміряних показників передають POST-запити.

3.3 Проектування бази даних

Для зберігання інформації програмна система застосовує базу даних mars_assistant, яка працює з таблицями СУБД MySQL 8.0 Server. Дана серверна

система ефективно функціонує у взаємодії з Інтернет-сайтами та веб-застосунками. Схему бази даних наведено на рисунку 3.4.

База даних складається з трьох таблиць:

– користувачі (Users), в ній зібрані всі приватні дані про користувачів системи, а саме: ідентифікатор, індивідуальний номер IoT-пристрою, електронна пошта, секретного паролю та статус;

– датчики (Sensors), в ній зібрані результуючі дані вимірювань, а саме: місце, дата та час вимірюваної локації, вид датчика, ідентифікатор користувача;

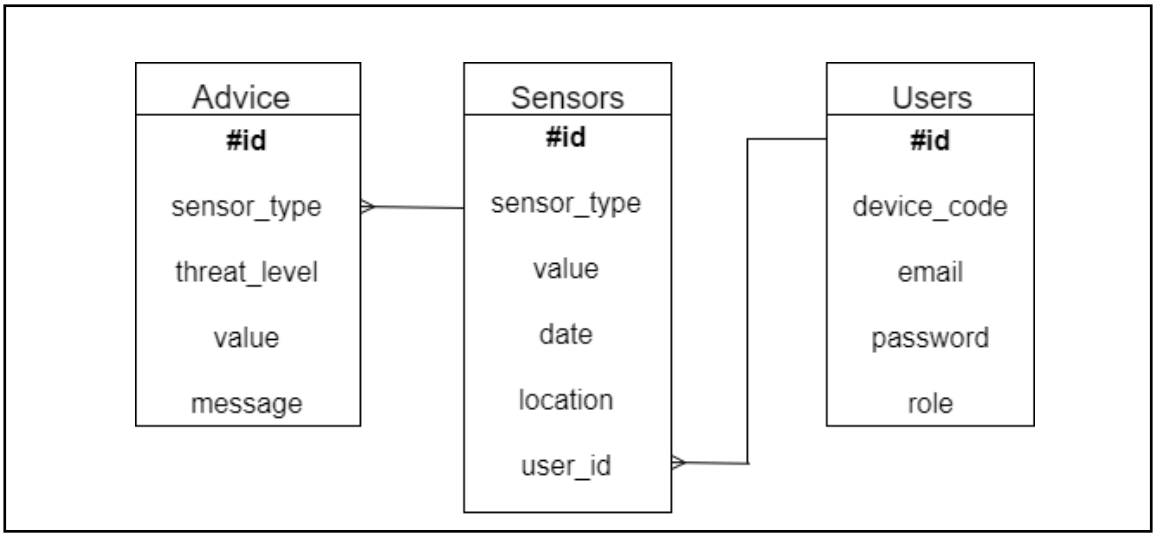


Рисунок 3.4 – Схематичне зображення бази даних mars_assistant

– поради (Advice): в ній зібрані поради, що будуть пропонуватися, а саме: ідентифікатор попередження, рівень загрози, вид датчика, значення рівня загрози та попереджуючий текст.

3.4 Проектування UI / UX дизайну системи

UX – це User Experience (з анг. «досвід користувача»). Це такий досвід користувача, що визначається тим, як він взаємодіє з застосунком/сервісом та які враження отримує.

UI – це User Interface (з анг. «інтерфейс користувача»). Тобто це те, як виглядає інтерфейс, графічний макет програми.

Початковим інтерфейсом для розглядання слугуватиме головна сторінка застосунку (рис. 3.5).

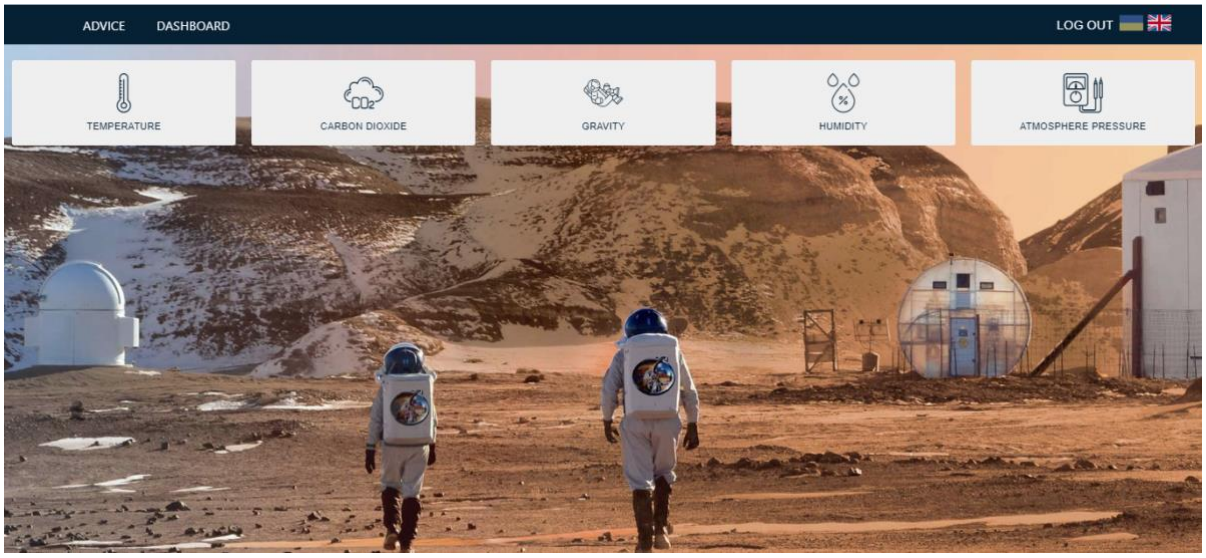


Рисунок 3.5 – Основна сторінка застосунку

Основними компонентами навігації головного інтерфейсу є 5 кнопок. Вони містять зображення показників вимірювань, для зручності орієнтування при натисненні на кнопку, також на них присутні й написи з назвою показника.

Для швидкого орієнтування користувачу серед елементів та вибору потрібного показника для запуску вимірювання вони розташовані у верхній веб-сторінці.

Двоє колонізаторів, що рухаються по планеті Марс є фоновим зображенням головної сторінки. Ця фотографія дозволяє користувачам поринути в марсіанський світ та показує, що система може працювати й допомагати людям не тільки на Землі.

На наступному екрані можна одержати результати вимірювання метрики запиту (рис. 3.6). Це основний інтерфейс, оскільки містить всю важливу інформацію про вибране користувачем вимірювання метрики, а також дані, характерні для цього інтерфейсу. Даний інтерфейс створений простим та зрозумілим, щоб будь-який користувач міг легкістю ним користуватися. При виборі користувачем кнопки одного з показників колір зміниться на помаранчевий.

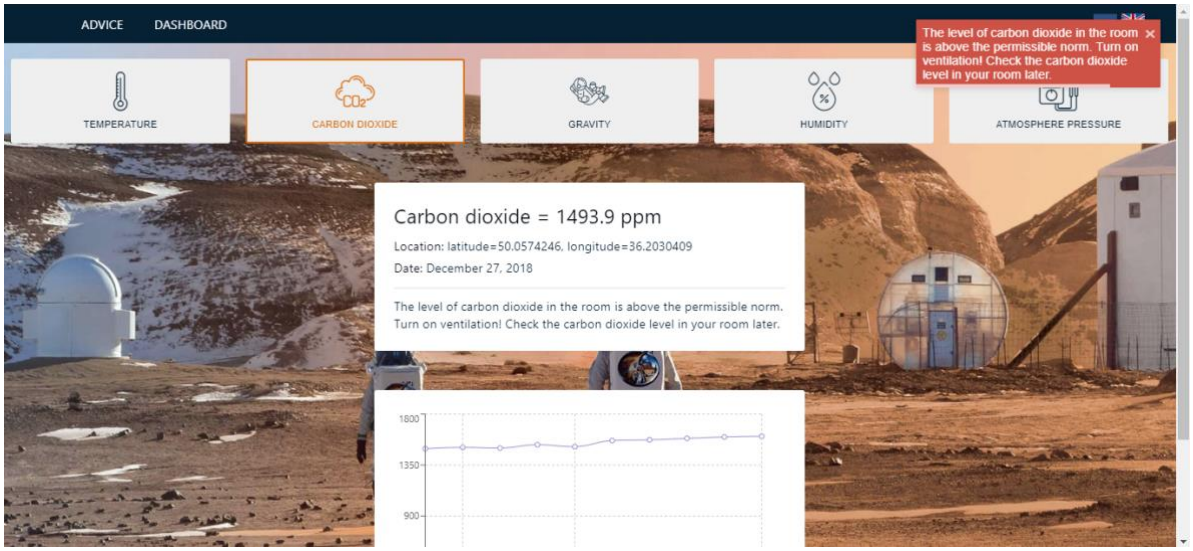


Рисунок 3.6 – Інтерфейс результату отриманих даних при запиті вимірюваного датчика

Інтерфейс системи є дуже важливим, так як при правильному користуванні користувач з легкістю зможе дослідити потрібний йому показник зі швидким результатом, отримати пораду та бути впевненим в своїй безпеці. Однак при появі небезпеки користувач побачить червоне push-повідомлення. Воно з'явиться у верхньому куті зліва дисплею й користувач відразу його неодмінно помітить. При дотриманні дій та застосування необхідних заходів, вказаних у повідомленні, користувач збереже своє життя та інших членів команди.

3.5 Розробка математичної моделі для задачі прийняття рішень в умовах невизначенності

Однією із ключових проблем людини є проблема прийняття рішень. Це викликано тим, що існує велика кількість неузгоджених один з одним критеріїв з великою часткою невизначеності та нестачею інформації для прийняття виваженого рішення.

Стохастичне програмування представляє об’єднання методів планування з вирішення оптимізаційних задач із врахуванням вірогідного (стохастичного) проведення процесів. Говорячи про стохастичні чи випадкові процеси, мають

на увазі процеси зміни в часі стану будь-якого елементу системи у поєднанні з ймовірними закономірностями[13].

Завданнями лінійного й частково інших видів програмування можуть слугувати завдання стохастичного програмування, у разі якщо аргументи головної функції чи системи обмежень можуть бути розглянені в якості випадкових величин. Під час розв'язання задач в стохастичній постановці звично користуються двома підходами.

Перший підхід полягає в пошуку середнього значення всіх даних випадкових параметрів, що не є ефективним завжди. Це викликано тим, що при деяких випадкових величинах через синергетичні явища рішення може бути прийняте зовсім далеке від оптимального чи може привести до ускладнень бажаного рішення поставленої задачі.

Ціль наступного підходу складається з багатоступеневого та поетапного наближення до потрібного результату. Якщо,для прикладу, на першому етапі утверджується первинний вигідний план, що при розв’язанні детермінованого завдання створеного на основі максимізації або мінімізації головної функції. Далі на другому кроці він коригується згідно з дійсними визначеними статистичними показниками та параметрами.

Головна задача стохастичного програмування заключається в пошуку екстремальних значень функціональної цілі (1) для системи обмежень (2) та гранично прийнятні значення змінних (3).

$$z = c_0 + c_1x_1 + c_2x_2 + \ldots + c_nx_n \rightarrow \max(\min), \qquad (1)$$

$$\begin{cases} a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = b_i \quad (i = 1,\ldots,l) \\ a_{j1}x_1 + a_{j2}x_2 + \ldots + a_{jn}x_n \geq b_j \quad (i = l+1,\ldots,p) \\ a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kn}x_n \leq b_k \quad (i = p+1,\ldots,m) \end{cases}, \qquad (2)$$

$$d_s \leq x_s \leq l_s, \quad (x_s \geq 0) \qquad (s = 1, \ldots, n). \qquad (3)$$

При плануванні оптимізації на основі умови (1) – (3) необхідно включати значення параметрів c_i, a_{ij} , b_j , d_i , e_i ($i = 1, \dots, n$; $j = 1, \dots, m$), що містяться в постановці задачі. У практичних розрахунках всі ці значення вважаються детермінованими, їх встановлені значення від випадкових чинників не залежать. Однак лише параметри d_i і e_i, ,що мають гранично допустимі значення x_j, будуть детерміновані, а решта параметрів c_i, a_{ij} , b_j є непередбаченими величинами.

Наприклад, температура повітря залежить від кута падіння сонячних променів, а кут падіння — від широти місцевості. Отож є всі причини вважати, що значення b_j , c_i та a_{ij} (1)-(3) задачі лінійного програмування є випадковими величинами, які по суті є прийняттям рішення в умовах випадкових значень цих значень.

Задачі зі стохастичними параметрами (1)-(3) часто називають задачами стохастичного програмування. Щоб її вирішити, потрібен опис непередбачених величини. З іншого аспекту, при описі повноти непередбачених величин ми розглянемо два випадки:

- відомо, наскільки може змінюватися випадкова величина й вона матиме назву як абсолютно невизначена задача планування, діапазон випадкових величин відомий, тому задача матиме назву як задача планування при абсолютній невизначенності;
- розподіл випадкових величин відомий, тому задача матиме назву як задача планування в передумовах ризику.

Абсолютно невизначеними параметрами вважаємо, які на основі аналізу первинного та виробничого характеру можна визначити масштаб проблеми для кожного випадкового параметра задачі (1)-(3) та те, як вони можуть змінитися протягом періоду планування:

$$min(c_i) \le c_i \le max(c_i),$$
$$min(a_{ij}) \le a_{ij} \le max(a_{ij}),$$
$$min(b_j) \le b_j \le max(b_j). \quad (4)$$

Розрахуємо плани для двох граничних випадків. Песимістичним вважатиметься план, де індикатор приймає мінімальне можливе значення $min\ b_j$, а інший індикатор прийматиме максимальне значення max a_{ij}. На нижній межі стоятиме значення показника c_i. . Підставляючи в модель граничні значення цих параметрів (1)-(2), виходить песимістичний план життя за умов $min\ x_i$ (i=1, … , n) , виконання гарантоване, але велика загроза для життя людини.

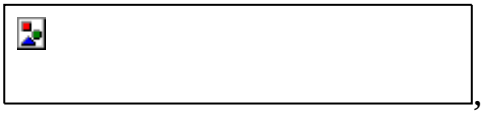
Оптимістичним план вважатиметься, якщо наявні показники прийматимуть більші значення $max\ b_j$, інші $min\ a_{ij}$ і $max\ c_i$.

Розв'язування задачі лінійного програмування (1)–(2) при наявних значеннях показників ми знайшли оптимальний графік роботи системи $max\ x_i$, який мав би найбільший ефект і не становив би загрози перебування на Марсі, але його виконання не гарантується. Візьмемо на замітку, що постановка задачі в песимістичних висловлюваннях часто несумісні.

Наступним кроком буде задача для другого випадку. Згадаємо, що при неперервній випадковій величині Y функція розподілу, що задана ймовірностями $F(y)$, випадкові величини в діапазоні від a до b мають однакову ймовірність:

$$P(a < Y < b) = F(b)-F(a). \qquad (5)$$

Коли випадкова величина Y належить нормальному закону розподілу, тоді функцію розподілу густини в (5) запишемо :

 ,

де

 ,

$M(Y)$, $\sigma(Y)$ – це середньоквадратичне відхилення та математичне сподівання випадкової величини Y відповідно.

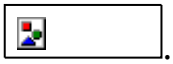
Є чотири можливі постановки обмежень для вирішення задачі:

а) визначити

$$P(Y< 0)\geq \alpha\,,\ (0\leq \alpha \leq 1), \qquad (6)$$

тобто визначити такі параметри нормального закону розподілу, при яких буде виконуватися умова (6) для деякого заданого α. Загалом, чіткого вирішення такої проблеми немає, але ймовірність порахувати можна:

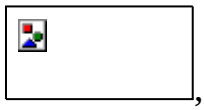
$$P(Y< 0) = P(-\infty <Y< 0) = F(b)-F(a) = \Phi^*(0) - \Phi^*(-\infty) = \Phi^*(0),$$

 , де  .

Тоді

$$P(Y< 0) = $$  .

Враховуючи (6), маємо

 ,

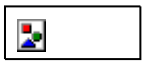
тоді

 ,

де $\Phi^{*-1}(\alpha)$ – обернена функція нормального розподілу.

Вводимо позначення $\tau_\alpha = \Phi^{*-1}(\alpha)$

Запишемо

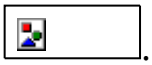
 . (7)

Маємо, що умова (6) задовольнятиме кожне математичне очікування $\bar{y}$, пов’язаному з відношенням (7) та з середнім квадратичним відхиленням σ,

б) визначити

$$P(Y< 0)\leq \alpha\,,\ (0\leq \alpha \leq 1). \quad (8)$$

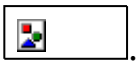
Математичне сподівання знайдемо за формулою

 . (9)

в) визначити

$$P(Y > 0)\geq \alpha\,, \quad (0\leq \alpha \leq 1). \quad (10)$$

Потім математичного сподівання знайдемо

 . (11)

г) для випадку

$$P(Y > 0)\leq \alpha\,, \quad (0\leq \alpha \leq 1). \quad (12)$$

Тоді математичне сподівання знайдемо

 . (13)

Добре відомо, що задачі лінійного програмування включають цільові функції та граничні умови.

Спершу розглянемо цільову функцію

c_i – випадкові величини.

Оптимізація математичного очікування для цільової функції

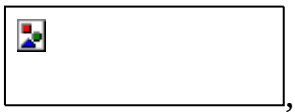
Також її можна записати як :

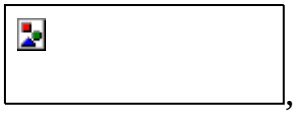
 (14)

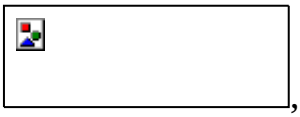
де  - математичне сподівання випадкової величини c_i.

Наступним кроком розглянемо обмеження. Наведені варіанти обмежень в стохастичній задачі:

 ,

 ,

 ,

 , (15 - 18)

де a_{ij} , b_j – випадкові величини;

α_j – задані рівні ймовірності.

Запишемо

 (19)

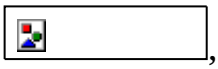
де y_i – випадкова величина.

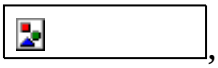
Зазвичай, випадкові величини c_i, a_{ij}, b_j, y_j вважаються нормально розподіленими і мають відомі перші моменти, а саме: математичне сподівання та дисперсію, як показано в таблиці 1.

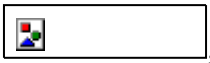
Таблиця 1 - Відповідність математичного сподівання та дисперсії до випадкової величини

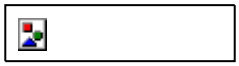
Випадкова величина	Математичне сподівання	Дисперсія
c_i		σ_i^2
a_{ij}		σ_{ij}^2
b_j		θ_j^2
y_j		σ_{yi}^2

Підставивши (19) в (15-18), маємо варіанти задач:

 , (а)

 , (б)

 , (в)

 (г) (20)

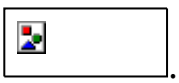
Показали, що при незалежних b_j і a_{ij} випадкова величина (19) матиме математичне сподівання:

 (21)

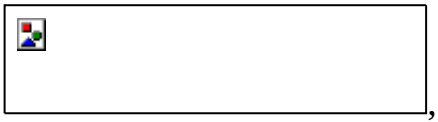
та дисперсію

 (22)

Підставимо ці значення в залежність (15), отримаємо формулу (20), маємо

 . (23)

Підставимо в (21), (22) значення, маємо

 ,

або, перетворивши, маємо

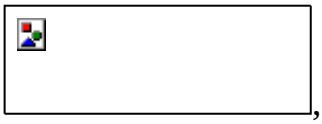
 . (24)

При обмеженнях в постановці задачі стохастичного програмування (24) порівнювати з відповідними обмеженнями задачі лінійного програмування

 ,

Тоді ми бачимо, що обмеження (24) мають дві характеристики:

– перехід від визначених значень a_{ij}, b_j до математичних сподівань випадкових величин  ;
– поява додаткового члену, що враховує усі вірогідні характеристики задачі.

 , (25)

Використовуючи позначення (25) співвідношення (24) запишемо так:

 (26)

Кореляція (26) є детермінованим еквівалентом задачі з можливими обмеженнями (20). Подібні детерміновані еквіваленти можна отримати для інших варіантів (20).

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Деталі реалізації серверної частини та API

Перш за все для зберігання системної інформації була розроблена база даних. Фреймворк Express 4 і knex міграції, за допомогою яких можливі зміни структури баз даних, використовувалися в роботі для створення бази даних. Потім формується міграція бази даних, на основі моделі коду створюються скрипти для побудови таблиць. На основі даної міграції створюється нова база даних. Далі у розробці є проектування системи бізнес-логіки. За допомогою розроблених сервісів відбувається отримання даних з бази та їх передача до API контролерів, за допомогою токенів - авторизація в системі та алгоритми формування підказок. На рисунку 4.1 показано програмну реалізацію одержання вимірюваних даних кінцевого результату з IoT-пристрою.

```
async getLast(userId, sensorType) {
       try {
      const sensor = await pool.query(
         `SELECT * FROM sensors WHERE user_id='${userId}' AND
sensor_type='${sensorType}' ORDER BY id DESC LIMIT 1;`
      );
      const advices = await pool.query(
         `SELECT advice.message, advice.value, advice.threat_level
FROM advice, sensors
          WHERE advice.sensor_type = ${sensorType} AND sensors.id =
${sensor[0].id}`
      );

      const adviceMessage = this.getAdviceMessage(advices,
sensor[0].value);
      if (adviceMessage !== 'Result is normal') {
         const title = this.getSensorTypeById(sensorType);
         notificationSevice.send(null, `${title} has value
${sensor[0].value}`, adviceMessage);
      }
      if (!sensor.length) throw "Not Found";

      return {status: 200, error: null, response: {sensor:
sensor[0], advice: {message: adviceMessage}}};
   } catch (error) {
      return {status: 404, error, response: {}};
   }
}
```

Рисунок 4.1 –Код одержання вимірюваних даних з IoT-пристрою

Порівняння оцінки вимірювань з нормованими даними, повернення тексту рекомендацій показано реалізацією алгоритму на рисунку 4.2.

```
getAdviceMessage(advices, sensorValue) {
   return advices.reduce((prev, advice) => {
      if (advice.value > sensorValue && advice.threat_level === 1
         || advice.value < sensorValue && advice.threat_level ===
2) {
         prev = advice.message;
      }
      return prev;
   }, 'Result is normal');
}
```

Рисунок 4.2 – Код порівняння оцінки вимірювань з нормованими даними

Наступним кроком було створення підсистеми авторизації. На рисунку А.3 додатка А зображена діаграма послідовного здійснення реєстрації користувача . Щоб отримати токен доступу до облікового запису, клієнт зобов'язаний надати підсистемі правильний номер пристрою, пароль профілю та адресу електронної пошти. Зібрані дані надсилаються на сервер авторизації й проходять перевірку. Якщо пошук здійснено успішно, дані генеруються токеном і надсилаються назад на сервер авторизації. Потім сервером авторизації створюється об’єкт формату JSON , про токен додається до отримана інформація і надсилається клієнту. Як наслідок, файл формату JSON надсилається користувачеві та він отримує дані, де зазначена інформація цього токена про клієнта.

Для зв’язку між сервером і клієнтською частиною та для отримання клієнтом інформації з бази даних створюється контролер API.

Щоб оминути несанкціоноваий доступ до даних для запитів, що потребують конфіденційної інформації, додається заголовок x-access-token.

4.2 Деталі реалізації веб-застосунку

Веб-застосунок розроблюється через технології React.js. На рисунку 4.3 показаний принцип роботи застосунку.

Компоненти використовуються для перегляду інформації та взаємодії з користувачем. Вони складаються з трьох частин: HTML-шаблонів для відтворення результатів, для проектування дизайну [5] та логіки компонентів каскадних таблиць стилів CSS, що розроблені за допомогою мови JavaScript.

Частина, відповідальна за логіку компонента, — це служба, яка отримує дані. Потім вона обробляє отримані дані, приймає їх у вигляді моделей та формує HTML-документ розмітки, який відображається у вікні браузера.

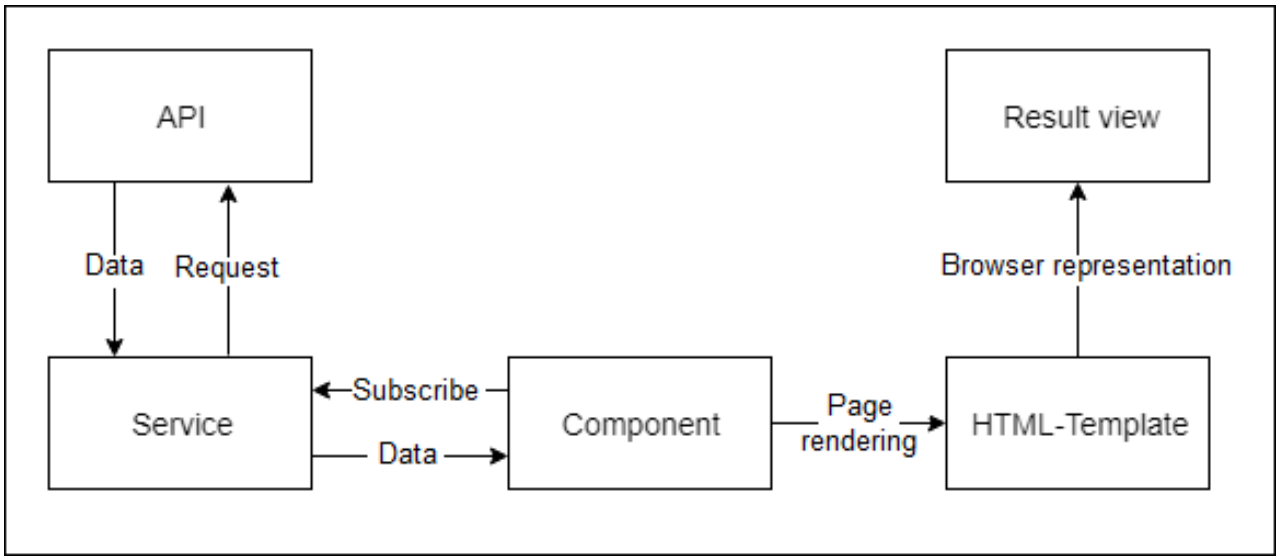


Рисунок 4.3 – Принцип роботи веб-застосунку

Функція fetch() використовується для обробки API, а її параметрами є URL-адреси та об’єкт з деякими параметрами. Функція надсилає дані запиту службі HTTPRequest , яка робить REST-запит до API і чекає відповіді від сервера. Відповідь отримується у вигляді даних у форматі JSON, а служба повертає функцію fetch().

Програмна реалізація отриманих кінцевих результатів вимірювань окремого показника зображена на рисунку 4.4.

Якщо дані отримані успішно, компоненту необхідна інформація надається, інакше компонент отримає повідомлення про помилку.

```
async getSensor(sensorType) {
    this.isFetching = true;
    try {
        const responseJSON = await fetch(`http://mars-
assistant.herokuapp.com/api/sensor/${UserStore.user.id}/${sensorTyp
e}/?last=true`, {
            method: 'GET',
            headers: {
                'Content-Type': 'application/json',
                'x-access-token': TokenStore.token,
            },
        });
        const response = await responseJSON.json();

        runInAction(() => {
            const { sensor, advice } = response.response;
            const msUTC = Date.parse(sensor.date);
            const date = new Date(msUTC);
            const formattedDate = date.toLocaleString("en-US",
{day: 'numeric', month: 'long', year: 'numeric'});

            this.sensors[sensorType] = {advice, sensor: {...sensor,
date: formattedDate}};
            this.isFetching = false;
            this.showedSensor = sensorType;
        })
    } catch (error) {
       runInAction(() => {
            this.error = error;
            this.isFetching = false;
       })
    }
}
```

Рисунок 4.4 – Код отримання кінцевих результатів вимірювання

4.3 Деталі реалізації мобільного застосунку

Розроблення мобільного застосунку створюється за технологією Android Java. Додаток протестовано на платформі Android версії 7.1.2 (Nougat).

На рисунку 4.5. показаний принцип роботи мобільного застосунку.

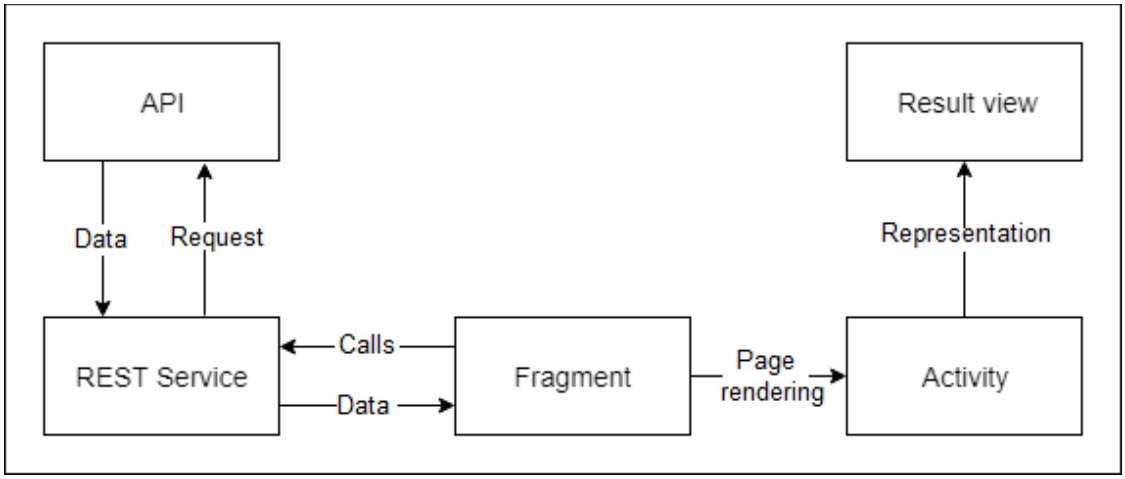


Рисунок 4.5 – Принцип роботи мобільного додатку

Макети використовуються для перегляду інформації та взаємодії з користувачами. Вони розроблені як файли розмітки XML. Частина XML-макету основної сторінки для появи порад показана на рисунку 4.6.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">


    <TextView
        android:id="@+id/advice_label"
        android:layout_width="125dp"
        android:layout_height="wrap_content"
        android:text="@string/date"
        android:layout_marginEnd="10dp"
        android:textColor="@color/darkBlue" />

    <TextView
        android:id="@+id/advice"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/darkBlue" />
</LinearLayout>
```

Рисунок 4.6 – Принцип роботи мобільного додатку

Діяльності (англ. activity) використовується для відображення макета на екрані мобільного пристрою.

Діяльність є однією з головних будівельних блоків програми на платформі Android. Вони діють як точка входу для користувача для взаємодії з

програмою та є центром навігації. Основною частиною слугує діяльність при розробці застосунків для Android . На екрані вони допомогають розмістити всі компоненти інтерфейсу користувача.

Із макета дільність отримує елемент через рідкісного ідентифікатора з класу Resource. Описує діяльність головної сторінки, що показана на рисунку 4.7.

Будь-яке вікно застосунку є відокремленим фрагментом, який відображає дані.

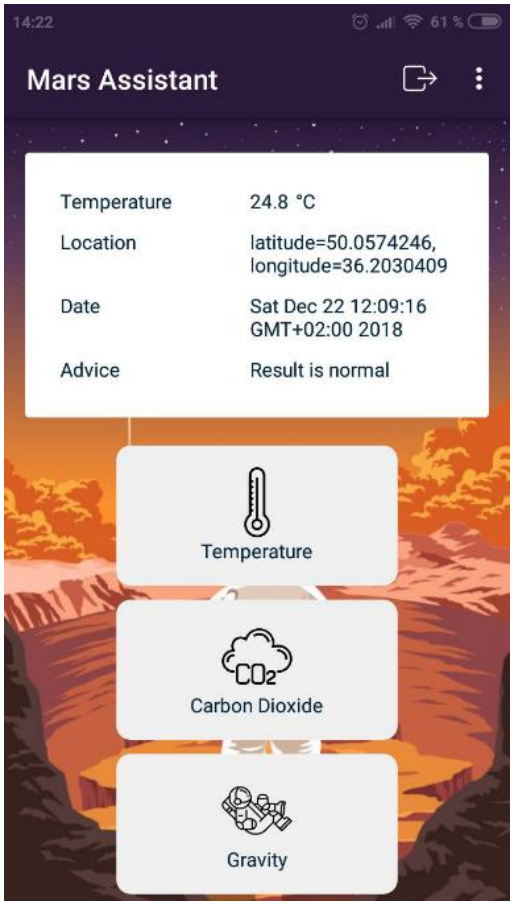


Рисунок 4.7 – Основна сторінка мобільного застосунку з зображенням результатів вимірювань

Сервіси використовуються для взаємодії з серверною частиною. На рисунку 4.8 показаний процес їх дії.

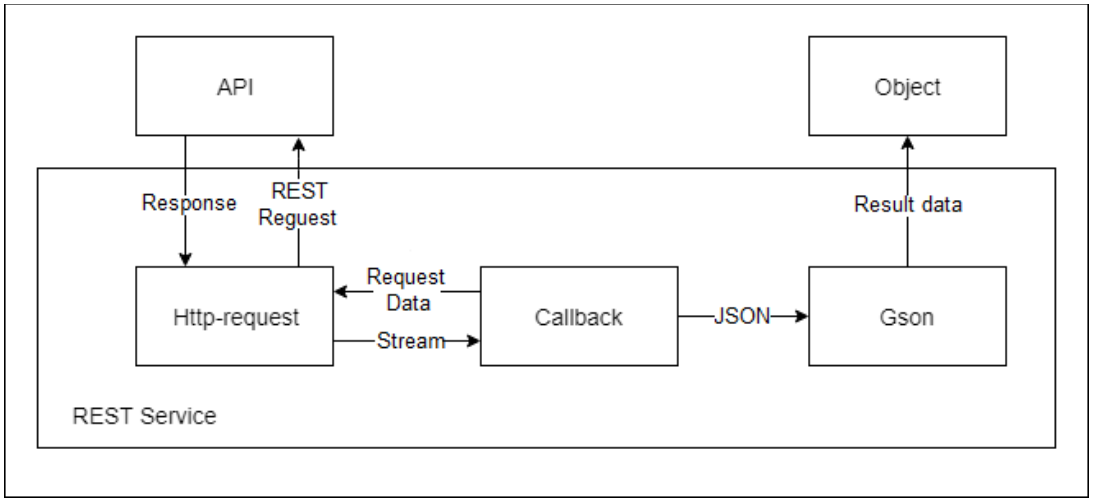


Рисунок 4.8 – Схема відправлення REST-запитів

За допомогою Retrofit 2 реалізований процес відправки запитів. Бібліотека чудово підтримує REST API, легко тестується та налаштовується.

Принцип роботи подібний на схеми, за якими запити надсилаються у веб-застосунок, а отримання даних відбувається через клас Callback та результати зпиту записує у рядок.

Бібліотека Gson перетворює отримані дані в класи моделей для відображення даних. У разі успішного виконання запиту створюється і повертається новий об’єкт моделі, інакше викидається помилка, яку обробляє фрагмент, що є зручним для відображення інформації про помилку.

4.4 Деталі реалізації IoT-пристрою

З використанням технологій NodeMCU реалізовані IoT-пристрої . Принцип дії роботи застосунку показано на рисунку 4.9. NodeMCU (v3) використовується як мікропроцесор. Він збирає дані з датчиків температури, атмосферного тиску, вологості, акселерометра, рівня вуглекислого газу і обробляє їх.

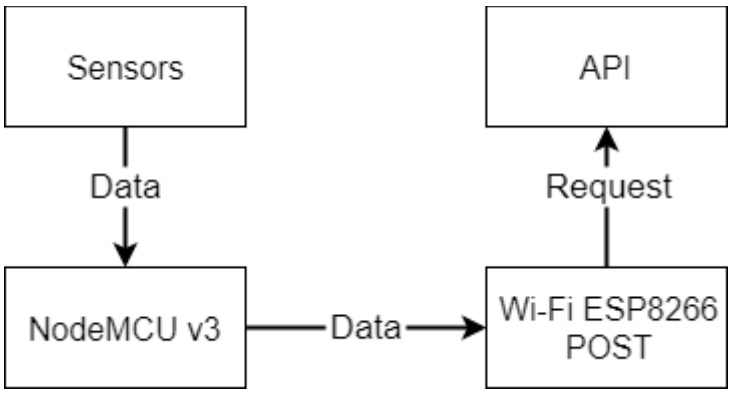


Рисунок 4.9 – Схема роботи IoT-пристрою

У результаті NodeMCU (v3) виводить опрацьовані дані, і передає дані модулю Wi-Fi ESP8266 , а модуль відправляє отриманий результат вимірювання на сервер через POST-запит шляхом підключення до мережі.

Програмна реалізація функції даних для відправлення з IoT-пристрою зображена на рисунку 4.10.

```
function send_to_server(value, sensorType)
if wifi.sta.status() == 5 then  --STA_GOTIP
    local http=require("http")
    http.post("http://mars-
assistant.herokuapp.com/api/sensor/"..user_id.."/"..sensorType,
    'Content-Type: application/json\r\nx-access-token:
'..user_token..'\r\n',
    '{"value": "'..value..'"}',
    function(code, data)
        if (code < 0) then
          print("HTTP request failed")
        else
          print(code, data)
        end
    end)
    else
    print("Still connecting...")
    end
end
```

Рисунок 4.10 – Код відправлення даних з IoT-пристрою

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У ході тестування програмного забезпечення були застосовані такі види тестування:

- тестування бази даних (Data and Database Integrity Testing);
- функціональне тестування (Functional Testing);
- тестування інтерфейсу користувача (User Interface Testing);
- тестування кросбраузерності (Cross Browser Testing);

Функціональне тестування — це розкриття відмінностей між фактичною та очікуваною поведінкою реалізованої функції відповідно до вимог та специфікацій [14]. Функціональне тестування у цій роботі — це комплексна перевірка того, чи відповідають функції заданим вимогам, а також перевірка всіх функціональних можливостей роботи.

На функціональне тестування даного проекту були поставлені такі завдання:

- тестування процесу реєстрації юзера;
- тестування авторизації;
- використовування неавторизованих користувачів для тестування програми;
- зміна паролю;
- дослідження показників IoT-пристроєм;
- одержання правильних результатів вимірювання;
- повідомлення користувачам про відхилення показників від норми;
- одержання рекомендацій щодо відхилень показника від норми;
- проглядання статистики.

Тестування бази даних – виконання тестів для запису, видалення й оновлення інформації в базах даних та перевірка точності отриманих даних.

Протягом даного тестування були зроблені такі дії:

- перевірка на відповідність усіх підключень бази даних до проекту;
- перевірка єдності даних і зв’язків;

– перевірка на правильність операцій, таких як запис, видалення та зміна інформації.

Кросс-браузерне тестування – це тестування поводження програми в різних браузерах. Для тестування використовувалися браузери:

– Google Chrome;
– Opera;
– Firefox.

Тестування інтерфейсу користувача–це вид тестування, який використовується для визначення того, чи є зручним об’єкт (наприклад, веб-сторінка, інтерфейс користувача чи пристрій) для використання за призначенням. Отже, тестування ергономіки вимірює ергономіку об’єкта чи системи. Ергономічне тестування зосереджується на конкретному об'єкті або невеликій групі об'єктів, тоді як взаємодія людини і комп'ютера в цілому - розробляє загальні принципи.

Тестування інтерфейсу користувача є найважливішим в створенні проекту, оскільки він має найбільшу кількість помилок і недоліків. При тестуванні використовувалися такі дії :

– тестування форми аутентифікації\авторизації;
– тестування дисплею порад;
– тестування перегляду статистичних даних;
– загальне тестування сторінок сайту, а саме: перевірка сторінок при різних розширеннях екрану, різних браузерах, коректність відображення усіх елементів дизайну та їх відгуків на дії користувача.

Компонентне (модульне або unit) тестування тестує функціональність окремих слабо пов’язаних частин програми і знаходить дефекти (або баги) в частинах додатка, що доступні та можуть бути протестовані порізно (модулі програм, а саме: класи, методи, функції, поведінка класів і т.д.).

Зазвичай unit тестування виконується викликаючи код, що потрібно протестувати, при сприянні середовищ розробки, відомі як фреймворки (frameworks - каркаси) для інструменти для дебагінгу чи модульного тестування.

Протягом написання програми були зроблені модульні тести, що засвідчують якість остаточного продукту. На рисунку 6.1 зображено таблицю покриття коду при виході інтеграційними й unit-тестами.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
All files	98.92	94.36	99.49	100	
components	99.17	93.95	100	100	
reducers	100	100	100	100	
actions	99.15	93.86	100	100	
helper/validation	98.7	94.72	99.07	100	
src/config	99.1	98.51	100	100	
src/screens	100	95.83	100	100	

Рисунок 5.1 – Таблиця покриття коду при виході модульними тестами

Інтеграційне тестування використовується для перевірки взаємозв'язку між компонентами на високих рівнях абстракції, а також взаємодії з різноманітними частинами програмної системи[15].

Перевірка функціональних та нефункціональних вимог є головним завданням системного тестування в системі. Завдяки такому підходу знаходяться дефекти, невірне використання системних ресурсів , а саме: непередбачені комбінації даних на користувацькому рівні, несумісність з робочим середовищем, несподівані сценарії використання, відсутність деякої функціональності або неправильна її реалізація, незручність при користуванні і т. д.

Приймальне (acceptance) – це тип тестування, який проводиться протягом здачі завершеного програмного продукту (або готової окремої частини програмного продукту) замовнику. Ціллю приймального тестування є детермінація готовності програмного продукту, яке реалізується шляхом виконання тестових сценаріїв (які були підготовлені раніше) і випадків, що створювалися на основі специфікації щодо розроблюваного програмного забезпечення.

Наслідками приймального тестування можуть бути:

– надсилання на допрацювання проекта;

– ухвалення і прийняття його замовником.

Тестування програмного продукту виконувалося на рівні модульного тестування. Коли брався деякий функціонал веб застосунку чи системи повністю та проводилось його тестування як негативне так і позитивне. Наприклад, тестування функціональності реєстрації корситувача проходило таким чином:

– відкриття форми реєстрації користувача;
– заповнення всіх необхідних даних;
– оцінка валідації даних;
– оцінка коректності роботи запиту перевірки присутності користувача в системі;
– оцінка коректності відповіді в тому інциденті, якщо дані були не коректні.

На рівні інтеграційного тестування були протестовані: спільна робота різних компонентів системи таких як: сайт, база даних, мобільний застосунок у взаємодії кожного компоненту системи між собою.

Ціллю даного тестування було визначення того, що у разі передачі даних між компонентами їх зміна чи пошкодження не відбувається. Тести проводилися через СУБД (Система управління базами даних) Microsoft SQL Server Management Studio та PostMan. СУБД застосовувалася для перевірки коректності даних у БД (База даних) та їх правильність, а інша для того щоб надсилати запити на API, де вони повинні оброблятися. Потім, після відпрацювання логіки API, повинні обновлятися дані у базі.

Тестування системи проводилося за допомогою створення різних Use-case сценаріїв та проходження по ним. У системному тестуванні були використані наступні аналізи: аналіз поведінки застосунку на різних платформах й браузерах, аналіз застосованих ресурсів та аналіз швидкості відгуку застосунку.

Приймальне тестування проводилося за окремими тестовими кейсами, що відображають головні випадки використання юзером інформаційної системи. На цьому етапі проводилося тестування головного функціоналу програми, а саме:

– процес авторизації;

- вимірювання показників IoT-пристроєм;
- отримання коректних результатів вимірювання;
- повідомлення юзера у разі відхилень показників від норми;
- отримання рекомендацій у разі відхилень показника від норми;
- перегляд статистичних даних.

ВИСНОВКИ

Шляхом досліджень і виконаної кваліфікаційної роботи було досліджено метод стохастичного програмування для прийняття рішень людиною за несприятливих умов для життя, розроблено програмну систему для моніторингу умов навколишнього середовища та психологічного стану та здоров’я людей під час небезпеки чи загрози життю. Проаналізовано предметну область та розроблено архітектуру програмної системи [16]. Протягом проектування системи були розроблені діаграми UML і створювалися макети дизайну системи UI/UX. Система розроблена на основі специфікації системних вимог [17] і технологій Node.js, React.js, Java Android і NodeMCU.

Розроблено чотири компоненти програмної системи: серверну частину, доступ до якої здійснюється через інтерфейс прикладного програмування (API), веб-застосунок і мобільний додаток, які взаємодіють з API, і пристрій IoT, який визначав значення концентрації різних датчиків, а саме: вуглекислого газу, температури, вологісті, сили тяжіння, атмосферного тиску, а також відправка даних на сервер, що відображатиме статистичні дані вимірювань, порівняння отриманих даних з прийнятними значеннями для відправки рекомендацій [18] .

Система допомагає користувачам визначати умови їхнього перебування на іншій планеті, в приміщенні чи на вулиці, отримувати повідомлення, якщо показники відхиляються від норми, і інструктувати дії [19] для порятунку себе та решти колонізаторської команди.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Journal of Biopharmaceutical Statistics Volume 30, 2020 - Issue 3//Стаття.URL: https://www.tandfonline.com/journals/lbps20 (дата звернення: 20.03.2022).

2. Михайленко Н. М. Аналіз біокліматичних чинників як умова рекреаційної діяльності. Географія та туризм. - 2014. - Вип. 27. - 129-136с.

3. Царик П.І. Оцінка ступеня сприятливості рекреаційних ресурсів клімату і погоди Поділля , – Тернопіль : Тайп, 2015. – 147–157с.

4. Human Settlement on Mars//Стаття. URL: https://www.mars-one.com (дата звернення: 20.03.2022).

5. Вигерс К. Разработка требований к программному обеспечению. – Петербург: БХВ-Петербург, Русская Редакция, 2016. – 736 с.

6. Хассан Г. UML-проектирование систем реального времени параллельных и распределенных приложени. – Москва: Пер. с англ. - М.: ДМК Пресс, 2011. – 704 с.

7. Леоненков А. Самоучитель UML. Эффективный инструмент моделирования информационных систем. – Петербург: БХВ-Петербург, 2001. – 304 с.

8. Наконечний С. І., Савіна С. С. Математичне програмування: Навч. посіб. — К.: КНЕУ, 2003. — 452 с.

9.Mars-assistant//Хостинг проекту. URL: https://dashboard.heroku.com/apps/mars-assistant (дата звернення: 20.03.2022).

10. Добро пожаловать на db4free.net//Електронна документація програмного засобу. URL: https://www.db4free.net/ (дата звернення: 20.03.2022).

11. Bringing MySQL to the Web//Електронна документація програмного засобу. URL: https://www.phpmyadmin.net/ (дата звернення: 20.03.2022).

12. Representational State Transfer (REST) //Стаття. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/ rest_arch_style.html (дата звернення: 20.03.2022).

13. Жильцов О.Б, Кулян В.Р., Юнькова О.О. Математичне програмування (з елементами інформаційних технологій): навч. посіб. — Київ: ДП «Видавничий дім «Персонал», 2008. — 184 с.

14. Бэнкс А. React и Redux: функциональная веб-разработка. – Санкт-Петербург: Питер, 2018. – 336 с.

15. Robert C. Martin. Clean Code. A Handbook of Agile Software Craftsmanship – Pearson Education, Inc., 2009. – 462 с.

16. Technopreneurship in Ukraine. How to Boost Enterpreneurial Competence Development in the Ukrainian IT Industry (The Ukrainian IT Educational System: Basic Facts and Urgent Needs. A. Mendes, Z.Dudar, V. Kauk, T. Shatovska, I. Revenchuk, A. Chupryna, D. Fedasyuk, V. Yakovina, I. Lyutak) edited by Hans Lundberg.//- Linnaueus University Copycenter, 2016.- 160p. ISBN: 978-91-88357-68-7.

17. Bondarenko M. F., Dudar Z. V., Revenchuk I.A. Information Systems and Technologies Used in Distance Form of Education at the University.- Informational and Communication Technologies Technologies – Theory and Practice: Proceedings of the International Scientific Conference ICTMC-2010 Devoted to the 80th Anniversary of I.V. Prangishvili. Nova Science Publishers. Series: Computer Science, Technology and Applications. - 2012.- P.485-490. ISBN: 978-1-61470-050-0.

18. Чалий С. Ф. Моделювання пояснень щодо рекомендованого переліку об’єктів з урахуванням темпорального аспекту вибору користувача / Чалий С. Ф., Лещинський В. О., Лещинська І. О. // Системи управління, навігації та зв’язку. 2019. Т. 6. № 58. С. 97-101.

19. Чалий С. Ф. Модель інтерфейсу пояснень з темпоральними параметрами в рекомендаційній системі / Чалий С. Ф., Лещинський В. О., Лещинська І. О. // Системи управління, навігації та зв'язку. 2020. Вип. 2(60). С. 105-109.