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

Інститут електроенергетики
(інститут)
Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня ___магістра__
(бакалавра, спеціаліста, магістра)

студента _____Чабаненко Антона Юрійовича_____
(ПІБ)
академічної групи _____123м-19-1_____
(шифр)
спеціальності _____123 «Комп'ютерна інженерія»_____
(код і назва спеціальності)
за освітньо-професійною програмою ___«Комп'ютерна інженерія»___
(офіційна назва)
на тему Комп'ютерна система приймальної комісії навчального закладу з детальним опрацюванням бота «Консультант абітурієнта»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	проф. Цвіркун Л.І.			
розділів:				
теоретичний розділ	проф. Цвіркун Л.І.			
синтез системи	доц. Ткаченко С.М.			
розроблення програмного забезпечення	ас. Бешта Л.В.			
експериментальний розділ	доц. Ткаченко С.М.			
Рецензент				
Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2020

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____ __Гнатушенко В.В._
(підпис) (прізвище, ініціали)

«___»_____ 2020 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня _____магістр _____
(бакалавра, спеціаліста, магістра)

студенту _____Чабаненко А.Ю.____ **академічної групи** ____123м-19-1____
(прізвище та ініціали) (шифр)

спеціальності _____123 «Комп'ютерна інженерія»_____

за освітньою-професійною програмою _123 «Комп'ютерна інженерія»__
(офіційна назва)

на тему Комп'ютерна система приймальної комісії навчального закладу з детальним опрацюванням бота «Консультант абітурієнта»,

затверджену наказом ректора НТУ «Дніпровська політехніка» від 22.10.2020 № 888-с

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	На основі матеріалів виробничих практик, інших науково-технічних джерел сформулювати наукове завдання, конкретизувати предмет та мету досліджень	21.09.2020
Теоретичний	Обґрунтувати теоретичну базу розв'язання наукового завдання, якому присвячено роботу	30.10.2020
Синтез системи	Розробка комп'ютерної системи	12.11.2020
Розроблення програмного забезпечення	Розробка програмного забезпечення	26.11.2020
Експериментальний розділ	Проведення і обробка результатів експериментів	06.12.2020

Завдання видано _____ _проф. Цвіркун Л. І.
(підпис керівника) (прізвище, ініціали)

Дата видачі _07 вересня_2020 р._

Дата подання до екзаменаційної комісії ____10.12.2020 р.____

Прийнято до виконання _____ _Чабаненко А.Ю._
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 84 с., 23 рис., 2 графіка, 6 табл., 21 джерел, 1 додаток.

Об’єкт розробки: «Комп'ютерна система приймальної комісії навчального закладу з детальним опрацюванням бота «Консультант абітурієнта»

Мета роботи: Розробка телеграм бота «Консультант абітурієнта» для приймальної комісії Національного технічного університету «Дніпровська політехніка» з метою поліпшити, а також осучаснити задачу ознайомлення абітурієнта зі спеціальностями, освітню програму яких представляє вищий навчальний заклад, розгляд необхідних документів для вступу, терміни подачі документів і дати зарахування до навчального закладу.

У пояснювальній записці було проведено аналіз роботи приймальної комісії, методів відомих рішень реалізації ботів та аналіз відомих напрямків моделювання систем. На базі проаналізованих частин було сформульовано задачу на розробку телеграм бота «Консультант абітурієнта». Наступним кроком було вирішення наукового завдання результатом якого є створення моделі системи масового обслуговування комп’ютерної системи «Консультанта абітурієнта». Після вирішення наукового завдання було наведено технічні вимоги, призначення розробки, вимоги до надійності, захисту інформації комп’ютерної системи для подальшого створення системи контролю комп’ютерної системи. Далі виконувалася розробка програмного забезпечення з описом функціональних можливостей та технічних засобів для підтримки коду, створенням алгоритмів з обґрунтуванням технічних характеристик програми. Кінцевим кроком було проведення експерименту на базі створеної моделі системи масового обслуговування.

ТЕЛЕГРАМ, БОТ, КОНСУЛЬТАНТ АБІТУРІЄНТА, ПРИЙМАЛЬНА КОМІСІЯ

ЗМІСТ

Стор.

Перелік умовних позначень, символів, скорочень і термінів 6

Вступ 7

1 Стан питання та постановка задач 11

1.1 Стан питання 11

1.1.1 Особливості роботи приймальної комісії НТУ «ДП» 11

1.1.2 Аналіз інформаційної структури приймальної комісії 12

1.1.3 Аналіз користувачів мессенджерів 15

1.2 Аналіз відомих варіантів реалізації комп'ютерних систем приймальної комісії 16

1.3 Аналіз відомих напрямів моделювання комп'ютерних систем з ботами 18

1.4 Постановка задачі 21

2 Теоретичний розділ 24

2.1 Загальна характеристика комп'ютерної системи приймальної комісії 24

2.2 Обґрунтування і вибір методів дослідження 25

2.3 Загальна модель системи масового обслуговування 28

2.4 Створення моделі комп’ютерної системи 30

2.5 Висновки по розділу 33

3 Синтез системи контролю 34

3.1 Вибір і обґрунтування принципів побудови проектованої комп’ютерної системи 34

3.2 Формулювання технічних вимог до комп’ютерної системи «Консультант Абітурієнта» 35

3.2.1 Вимоги до системи в цілому 35

3.2.1.1 Вимоги до реалізації комп’ютерної системи 35

3.2.1.2 Вимоги до функцій виконуваних системою 36

3.2.2 Вимоги до видів забезпечення 37
3.2.3 Вимоги до захисту інформації 38
3.3 Вибір обладнання 38
3.4 Синтез структурної схеми системи за заданими показниками комп’ютерної системи 41
3.5 Висновки по розділу 42
4 Розробка програмного забезпечення комп’ютерної системи «Консультанта абітурієнта» 43
4.1 Призначення й сфера застосування програми 43
4.2 Обґрунтування технічних характеристик програми 43
4.3 Опис розробленої програми 47
4.3.1 Загальні відомості 47
4.3.2 Функціональне призначення 49
4.3.3 Опис логічної структури 51
4.3.4 Використовувані технічні засоби 68
4.4 Очікувані техніко-економічні показники 70
4.5 Висновки по розділу 71
5 Експериментальний розділ 72
5.1 Формулювання завдання та обґрунтування методики 72
5.2 Вимоги до експерименту 75
5.3 Результати експерименту 75
5.3.1 Сутність експерименту 75
5.3.2 Результат експерименту у фактах 76
5.3.3 Аналіз відповідності досліджень 78
5.3.4 Характеристика новизни результатів 79
5.4 Висновки по розділу 79
Висновки 80
Перелік посилань 82
Додаток А Текст програми комп’ютерної системи «Консультанта абітурієнта» 84

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

НТУ «ДП» – Національний технічний університет «Дніпровська політехніка»

ПЗСО – Повна загально середня освіта

ОКР – Освітньо-кваліфікаційний рівень

СМО – Система масового обслуговування

DHCP – Dynamic Host Configuration Protocol

FTP – File Transfer Protocol

ПК – Персональний комп’ютер

ПЗ – Програмне забезпечення

API - Application Programming Interface

Бот – спеціальна програма, що виконує автоматично і/або за заданим розкладом які-небудь дії через ті ж інтерфейси, що й звичайний користувач

Telegram – клауд-месенджер, програмне забезпечення для смартфонів, планшетів та ПК, яке дозволяє обмінюватися текстовими повідомленнями, графічними та відеофайлами, а також безкоштовно телефонувати іншим користувачам програми

Приймальна комісія - робочий орган закладу вищої освіти, що утворюється для організації прийому вступників

Абітурієнт – людина, що поступає до вищого або спеціального навчального закладу

Сервер – програмний компонент обчислювальної системи, що виконує сервісні функції по запиту клієнта, надаючи йому доступ до певних ресурсів або послуг.

Інтернет – всесвітня система сполучених комп'ютерних мереж, що базуються на комплекті Інтернет-протоколів.

ВСТУП

Національний технічний університет «Дніпровська політехніка» є одним з ключових центрів з підготовці кадрів у таких сферах як: інформаційні технологій, економіки, права, гуманітарних наук, педагогіки, соціальні та поведінкові науки, управління та адміністрування, біологія, природничі науки, механічна інженерія, електрична інженерія, автоматизація та приладобудування, хімічна та біоінженерія, електроніка та телекомунікації, виробництво та технології, архітектура та будівництво, сфера обслуговування, цивільна безпека, транспорт, публічне управління та адміністрування, міжнародні відносини.

Вибір напрямків підготовки і освітніх програм досить масштабний, в зв'язку, з чим абітурієнти змушені витрачати години, дні або тижні, щоб дізнатися цікаві для них подробиці про кожного великого представника в сфері освітніх послуг. Актуальність роботи диктує ситуація на ринку інформаційних технологій. З кожним роком його поповнюють нові програмні продукти, веб-сервіс та мобільні додатки, за допомогою яких можна побудувати не тільки комунікації, а й сприяти зміцненню бренду університету, як освітнього центру.

Метою даної магістерської роботи є розробка програмного забезпечення «Консультант абітурієнта», що дозволяє абітурієнтові в стислій формі ознайомитися з усіма спеціальностями, які пропонує університет, дізнатися необхідний пакет документів для вступу, терміни подачі документів і дати зарахування до навчального закладу.

Для досягнення поставленої мети, потрібно вирішити такі завдання:

- виявлення вимог, якими має володіти програмне забезпечення;
- виявлення вимог, якими має володіти програмне додаток для управління контентом даного програмного забезпечення;
- спроектувати архітектуру програмного забезпечення;
- розробка зручного інтерфейсу адміністративного модуля для телеграм бота;

– впровадження розробленого програмного забезпечення.

Основний привілеєм кожної установи вищого навчального закладу є випуск кваліфікованих фахівців, затребуваних на ринку праці. В умовах високої конкуренції серед вищих навчальних закладів, необхідно підтримувати комунікацію з абітурієнтами.

Таким чином, рішення розробити чат-бота було продиктовано бажанням оптимізувати діяльність співробітників і студентів, що проходять практику в приймальній комісії університету, вимушених витрачати більшу кількість часу на трансляцію інформації, що міститься у відкритих джерелах. Практичну значимість розробленого чат-боту надає прив'язка до конкретним користувачам - абітурієнтам НТУ «ДП». Як джерела інформації використовувалася офіційна документація для розробників на мові JavaSvript, накази та інші документи, які містять інформацію про форми навчання та напрямках підготовки.

Мета і завдання дослідження. *Метою роботи* є розробка і дослідження моделей комп'ютерної системи «Консультант абітурієнта» приймальної комісії Національного технічного університету «Дніпровська політехніка» на базі системи масового обслуговування для визначення доцільності впроваджуваної комп'ютерної системи, а також її ефективності.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– дослідити структуру можливих методів реалізації подібних моделей комп’ютерних систем;
– на основі дослідження існуючих моделей подання знань розробити модель системи масового обслуговування компютерної системи «Консультант абітурієнта»;
– розробити систему контролю у компютерній системі, ввести вимоги до видів забезпечення створюваної системи, функціональні вимоги;
– розробити схему алгоритма функціональної реалізації компютерної системи;

- програмно реалізувати отриманні моделі, методи та алгоритми і створити компютерну систему «Консультант абітурієнта».
- провести експеремент на базі створеної системи масового обслуговування для доведення ефективної роботи створеної компютерної системи.

Об'єкт дослідження – комп'ютерна система «Консультант абітурієнта» приймальної комісії Національного технічного університету «Дніпровська політехніка».

Предмет дослідження – модель системи масового обслуговування компютерної системи «Консультант абітурієнта».

Методи дослідження. Для досягнення поставленої мети використовувалися методи теорії штучного інтелекту, теорії системи масового обслуговування, теорії нечітких множин, математичної статистики.

Наукові положення:

1. Розроблені вимоги до створення та моделювання комп’ютерних систем з детальним опрацюванням бота для надання консультаційних послуг у сферах діяльності приймальних комісій вищих навчальних закладах.

2. Виділені і охарактеризовані етапи створення моделей комп’ютерних систем з детальним опрацюванням бота, розробки вимог для таких систем, створення систем підтримки та контролю на основі вибору апаратного забезпечення для комп’ютерних систем такого типу, розробки програмного забезпечення для комп’ютерних систем з ботами на основі Telegram Bot API.

Наукові результати:

1. Одержані нові теоретичні відомості про створення и обґрунтування доцільності моделі комп’ютерної системи з ботами за допомогою теорії системи масового обслуговування.

2. Запропонований метод створення комп’ютерних систем з детальним опрацюванням бота для реалізації консультаційних послуг у сфері приймальної комісії вищих навчальних закладах.

3. Обґрунтовано застосування теорії системи масового обслуговування для моделювання комп’ютерних систем з ботами та проведення експериментальних досліджень на базі СМО за для вирішення протиріччя доцільності розробленої комп’ютерної системи.

Обґрунтованість і достовірність наукових положень, висновків і рекомендацій підтверджуються тим, що в роботі використані: сучасні методи розробки комп’ютерних систем з ботами завдяки вибору передових рішень з моделювання таких комп’ютерних систем, вибору сучасного обладнання для здійснення розробки програмного забезпечення та його подальшої підтримки, вибору ефективних мов програмування та платформ редагування коду, експериментальні підтвердження результатів теоретичних досліджень.

Практичне значення отриманих результатів полягає в розробці методу реалізації та підтримки комп’ютерної системи з детальним опрацюванням бота на базі інтерфейсу прикладного програмування, а також завдяки експериментальним висновкам, що доводять доцільність створеної комп’ютерної системи.

1 СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧ

1.1 Стан питання

1.1.1 Особливості роботи приймальної комісії НТУ «ДП»

Приймальна комісія Національного технічного університету «Дніпровська політехніка» [1] (далі – приймальна комісія) — робочий орган, що утворюється для організації прийому вступників. Строк повноважень приймальної комісії становить один календарний рік. Приймальна комісія працює на засадах демократичності, прозорості та відкритості. Діяльність приймальної комісії здійснюється відповідно до законодавства України, Положення про приймальну комісію закладу вищої освіти, затвердженого наказом Міністерства освіти і науки України від 15 жовтня 2015 року № 1085, а також правил прийому до закладу вищої освіти, статуту закладу вищої освіти та положення про приймальну комісію закладу вищої освіти, яке затверджується Вченою радою закладу вищої освіти.

Для виконання покладених на приймальну комісію завдань і здійснення нею своїх функцій наказом керівника закладу вищої освіти утворюються наступні підрозділи приймальної комісії:

- предметні екзаменаційні комісії;
- комісії для проведення співбесід;
- фахові атестаційні комісії;
- предметні комісії;
- апеляційна комісія (апеляційні комісії);
- відбіркова комісія (відбіркові комісії), у разі потреби.

Основні завдання та обов'язки приймальної комісії:

- приймальна комісія забезпечує інформування вступників, їх батьків та громадськість з усіх питань вступу до закладу вищої освіти;
- організовує прийом заяв та документів, приймає рішення про допуск вступників до участі в конкурсі (до участі у вступних випробуваннях);

- подає до Єдиної державної електронної бази з питань освіти отримані від вступників дані про них, вносить зміни до статусів заяв вступників в Єдиній базі;
- координує діяльність усіх структурних підрозділів закладу вищої освіти щодо підготовки та проведення конкурсного відбору;
- організовує і проводить консультації з питань вступу на навчання та вибору спеціальності, що найбільш відповідає здібностям, нахилам і рівню підготовки вступників;
- організовує та контролює діяльність технічних, інформаційних і побутових служб щодо створення умов для проведення вступної кампанії;
- забезпечує оприлюднення на веб-сайті закладу вищої освіти документів щодо забезпечення прийому до цього закладу, передбачених законодавством;
- приймає рішення про зарахування вступників за формами навчання і джерелами фінансування.

Склад приймальної комісії та її підрозділів, за винятком осіб, які входять до них згідно з посадовими обов'язками, щороку поновлюється не менш ніж на третину.

Допускається включати до складу приймальної комісії науково-педагогічних (педагогічних) працівників інших навчальних закладів.

Одна й та сама особа може бути відповідальним секретарем не більше ніж три роки поспіль.

До складу приймальної комісії, предметних екзаменаційних, відбіркових і фахових атестаційних комісій та апеляційної комісії не можуть входити особи, діти яких вступають до цього закладу вищої освіти в поточному році.

1.1.2 Аналіз інформаційної структури приймальної комісії

Інформаційною структурою приймальної комісії є методи взаємодії, консультування та спілкування з абітурієнтами [2]. Приймальна комісія

повинна мати вивірений і сучасний підхід до майбутніх студентів Вищого навчального закладу, це відображає його статус.

можливим підкреслити два способи взаємодії з абітурієнтами: очний і дистанційний.

Під очними способами мається на увазі:

- консультування членів приймальної комісії в університеті;
- перебування абітурієнта на дні відкритих дверей;
- причетність абітурієнта в наукових подіях, що проводяться університетом (олімпіади, конференції та ін.)
- причетність абітурієнта в креативній події, що проводяться університетом;
- співучасть абітурієнта в екскурсіях, що проводяться в НТУ «ДП».

Під дистанційними способами мається на увазі:

- телефонна консультування абітурієнта працівником приймальної комісії;
- консультування поштою абітурієнта працівником приймальної комісії;
- консультування по електронній пошті абітурієнта працівником приймальної комісії;
- консультування абітурієнта працівником приймальної комісії за допомогою соціальних мереж («Facebook», «Instagram») [3, 4];
- консультування абітурієнта працівником приймальної комісії за допомогою мессенджерів («Viber», «Telegram», «WhatsApp»).

На нинішній період, приймальна комісія динамічно освоює дистанційний спосіб взаємодії з абітурієнтами. Процедура обслуговування «Старих», очних засобів взаємодії проходить нестабільно і досить довгий час, незважаючи на явні переваги дистанційних методів взаємодії, з числа яких: загальнодоступність, низька вартість, практичність для всіх учасників взаємодії, ясність, ймовірність не втратити історію листування та інше.

Мессенджери ще нещодавно вважалися поза платформою, однак ця кваліфікаційна робота викликана усунути і компенсувати дане упущення.

Розберемо детальніше ключові мінуси очних способів взаємодії:

1. Обмеження в часі. Приймальна комісія має точний режим роботи, який обмежується буднями з дня на день. Двері приймальної комісії відкриваються о 9:00, в загальному випадку слушним часом для відвідування є середина дня, а це приблизно з 11:00 по 13:00 годину, якраз в цей час більшість абітурієнтів приходять за консультацією в Приймальну комісію НТУ «ДП». Поява більшої частини абітурієнтів в день саме в цей час викликає перевантаженість працівників приймальної комісії, а з 13:00 до 14:00 зазвичай працівники йдуть на обід, що викликає собою часто невдоволення відвідувачів і в свою чергу відкидає деяку частину абітурієнтів від консультацій, а в подальшому і від можливості вступу до НТУ ДП. Закривається приймальна комісія о 18:00, працівники залишають свої робочі зони і повернуться тільки наступного дня, нерідко університет втрачає можливих студентів в зв'язку з обмеженим періодом консультування і функціонування.

2. Незадовільна «прозорість» спілкування. При спілкуванні з працівником приймальної комісії за допомогою мережі Інтернет – наприклад, соціальні мережі, електронна пошта або мессенджери, зберігається історія листування. До неї у всякий час є можливість повернутися при потребі виконання процедури апеляції.

3. Великі тимчасові витрати. Організація спеціалізованих заходів у інтересах абітурієнтів, які мають всі шанси стати «майданчиком для спілкування» абсолютно для всіх зацікавлених сторін (абітурієнти, батьки, працівники університету) займає величезну кількість часу, що вкрай рідко окупається числом можливих зацікавлених у вступі абітурієнтів.

Підводячи висновок короткого розгляду, є можливість зробити висновок, що прийшов час розглядати дистанційні форми взаємодії з абітурієнтами як новий основний вид консультаційних послуг приймальної комісії, особливо, за допомогою мережі Інтернет. А безпосередньо, консультування абітурієнтів за допомогою:

- офіційного веб-сайту;
- сторінок і аккаунтів у соціальних мережах;

– мессенджерів.

При цьому мессенджери до сьогодення вважаються неохопленою платформою, що ймовірно може бути більш зручним засобом налагодження контакту між абітурієнтами та працівниками приймальної комісії.

1.1.3 Аналіз користувачів мессенджерів

Мессенджери стають найбільш поширеним каналом комунікації, захоплюючи в тому числі і громадське місце з підтримкою чатів і каналів. Вже у даний час можна сказати про те, що канали в мессенджерах стали комфортною основою користування контенту для безлічі користувачів соціальних мереж.

Використання мессенджерів зв’язано не тільки з індивідуальним спілкуванням, а також з спілкування в публічних чатах і читання публічних каналів, що є одним з трендів 2020 року. Чати і канали перетворюють мессенджери в соціальні мережі та медіа, що в свою чергу залучає нових користувачів до цих платформ. Засновником тенденцій тут безперечно є безкоштовний мессенджер «Telegram» [5], його успіх у сфері швидких листувань і орієнтує мессенджери розширюватися настільки значними темпами.

У споживанні новинного контенту користувачі Телеграма вибирають добірки новин з різних новинних каналів. Хоч ці канали і не завалюють своїх користувачів різного роду інформацією, як це роблять ЗМІ, вони подають інформацію яка відповідна тематиці каналу, а з огляду на це користувач знає, що йому чекати від цього каналу. Найчастіше телеграм канали публікують інформацію стисло і по факту, але ця інформація в основному має важливий, короткий посил для користувача, в результаті він отримує те чого хоче і не витрачає багато часу на пошук інформації.

Резюмуючи, тільки і залишається зауважити, що тренд збільшення інтересу до мессенджерів безперервний. У сукупному фоні подвійного збільшення обсягу згадок, фаворитом зростання з чотириразовим показником став Телеграм. Збільшення інтересу до Телеграму пов'язаний, з активним

застосуванням публічних каналів, з року в рік кількість користувачів українськомовних каналів збільшилося в рази, з сотень до десятків тисяч. Більш інтенсивне збільшення показали канали з рідкісним авторським контентом - розважальним, освітнім і канали з важливою інформативною денною повісткою і політикою.

Згідно з аналізом, проведеним самостійним виданням «Telegram-store», вік і соціальний статус користувачів в українськомовному секторі ділиться відповідно:

- вік з 16 до 35 року;
- отримує або має вищою освітою;
- має роботу інтелектуального характеру (значна частка it-експертів).

Функціонал мессенджерів, зокрема Telegram [3], націлений в першу чергу на мобільну аудиторію, яка стрімко використовує телефони з метою роботи або відпочинку: користувачеві зовсім не потрібно заходити на веб сайт, цілком достатньо відправити повідомлення боту.

1.2 Аналіз відомих варіантів реалізації комп'ютерних систем з приймальної комісії

Для реалізації бота у мессенджерах дуже часто використовують спеціально існуючі платформи для їх швидкої реалізації. Використання цих платформ для реалізації простих ботів можливо, але для більш детального опрацювання потрібно мати навики у розробці програмного продукту, написанні коду та його реалізації. Найчастіше ці спеціалізовані платформи мають безплатний бар’єр переходячи який користувач цих платформ має платити або разову суму для подальшого використання, або має оформити щомісячну підписку на неї, як правило всі можливі функції доступні користувачеві тільки коли він заплатив за користування платформою для розробки, це означає, що повна реалізація комп’ютерної системи «Консультант абітурієнта» неможлива при участі подібної платформи для реалізації ботів. Як

приклад існуючих спеціальних платформ для реалізації ботів можна перелічити наступні рішення:

1. Flow XO [6]

Підтримувані платформи реалізації: Facebook, Slack, SMS та Telegram.

Вартість: безкоштовно до п’яти ботів, платні тарифи від $19 в місяць.

Мова інтерфейсу: англійська.

Можна створити стандартне привітання для сайту або навчити бота розуміти запити і загальні фрази від клієнтів.

Для розробників сервіс пропонує отримувати і передавати дані, використовуючи HTTP API, збирати і використовувати інформацію про користувачів, а також пов'язувати процеси.

2. Botmother [7]

Підтримувані платформи реалізації: Facebook Messenger, Telegram, Viber, «ВКонтакте», «Одноклассники».

Вартість: безкоштовно за першого бота, потім — від 799 рублів в місяць.

Мова інтерфейсу: російська.

Розробники називають конструктор «Wordpress, тільки для ботів». Щоб створити бота, потрібно з'єднати між собою компоненти. Бот працює з медіа та вміє відправляти запити до API сторонніх додатків. Платежі в месенджерах бот приймає через PayOnline.

3. Gupshup [8]

Підтримувані платформи реалізації: Facebook Messenger, Telegram, Viber, Skype, «ВКонтакте» та інші.

Вартість: по запиту.

Мова інтерфейсу: англійська.

У платформу інтегровані інструменти для обробки природної мови. Складні діалоги можна проектувати за допомогою древа. Для досвідчених розробників створена IDE Bot Builder. Також сервіс випустив платформу для об'єднання в один інтерфейс 30 каналів комунікації, а також вбудовані CRM-інструменти для відстеження маркетингових кампаній.

Для реалізації більш детальних та опрацьованих ботів потрібно використовувати власні знання у програмуванні та реалізації програмного коду. Мова програмування може бути будь-якою, чіткого регламенту реалізації немає, але треба пам’ятати про те, що писати треба тією мовою якою зручно і яка буде займати найменше простору. Прикладом мови програмування є JavaScript (JS) — динамічна, об'єктно-орієнтована прототипна мова програмування, Python — інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією.

Реалізація програмного коду обумовлена тільки API платформи реалізації коду. Якщо платформа реалізації коду не підтримує ту чи іншу мову програмування слід або змінити платформу, або використовувати іншу мову програмування.

Прикладний програмний інтерфейс (англ. Application Programming Interface, API) — набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення. Спрощено – це набір чітко визначених методів для взаємодії різних компонентів. API надає розробнику засоби для швидкої розробки програмного забезпечення. API може бути для веб-базованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек.

1.3 Аналіз відомих напрямів моделювання комп'ютерної системи з ботами

Моделлю називається представлення об’єкта, системи чи поняття в деякій абстрактній формі, що є зручною для наукового дослідження [9].

В загальному випадку модель має структуру, зображену на рисунку 1.1. Тут X – множина вхідних змінних системи, Y – множина вихідних змінних системи, P – множина параметрів, F – функція, функціонал, алгоритм або формальне представлення залежності змінних Y від змінних X.

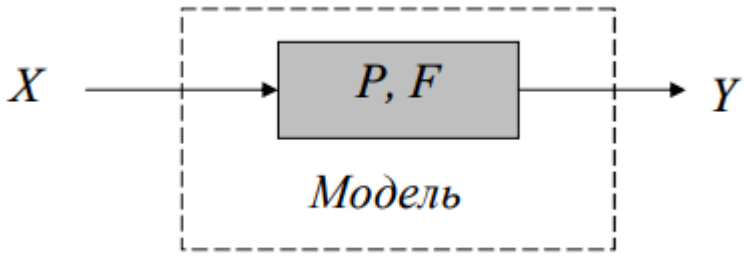


Рисунок 1.1 – Загальна структура моделі

Існують два способи побудови моделей. При першому способі в результаті ретельного вивчення системи встановлюються закони функціонування системи, які потім відтворюються за допомогою моделі. Поведінку системи, таким чином, досліджують на моделі. Параметри моделі P в цьому випадку пов’язані з реальними процесами, що протікають в системі, і мають фізичну інтерпретацію. Тому моделі такого типу називають фізичними моделями.

При другому без усякого фізичного обґрунтування припускається вид залежності F, невідомі параметри якої P потім відшукуються за даними спостережень за змінними системи X, Y. Параметри P в цьому випадку не зв’язані з фізикою реальних процесів, що протікають в системі, або, точніше, цей зв’язок досліднику залишається невідомим. Тому моделі такого типу називають нефізичними моделями. В літературі зустрічаються також терміни моделей типу сірого та чорного ящику, які еквівалентні термінам фізичної та нефізичної моделі. Для фізичної моделі закони функціонування системи досліднику відомі, тому ящик є прозорим, сірим. Для нефізичної моделі сутність системи залишається для дослідника скритою, потаємною, тобто ящик є чорним.

Задача моделювання (або пряма задача) полягає у відшуканні значень вихідних змінних Y при відомих значеннях вхідних змінних X, відомій моделі F та визначених параметрах P (див. рис. 1.1).

Серед великої кількості методів моделювання, що існують, виділимо такі методи: аналітичне моделювання, математичне моделювання, імітаційне моделювання.

Моделювання аналітичне, якщо представлення залежності F вихідних змінних Y від вхідних її змінних X має аналітичний вигляд, тобто представлений у вигляді відомих аналітичних функцій. Нагадаємо, що функція називається аналітичною, якщо вона розкладається у ряд Тейлора. Аналітичні функції диференційовані безліч разів і тому до них можуть застосовуватись методи математичного аналізу. Перевагою цього методу моделювання є можливість отримання залежності Y=f(X) в явному вигляді і застосування до неї методів класичного математичного аналізу. Якщо є можливість побудувати аналітичну модель системи, то завжди віддають перевагу цьому методу моделювання. Зауважимо, що відшукання залежності Y=f(X) може виявитись настільки складним, що досліднику доведеться застосовувати спеціальне програмне забезпечення, а для деяких систем доводиться відмовлятись від пошуку абстрактної залежності Y=f(X) і задовольнятись наближеним розв’язком, що знаходиться чисельними методами.

Деякі системи настільки складні, що не дивлячись на те, що опис їх функціонування піддається опису аналітичними функціями, знаходження залежності Y=f(X) у явному вигляді виявляється неможливим. Наприклад, усі задачі математичного програмування мають досить простий аналітичний опис, але розв’язок задачі може бути знайдений тільки в результаті виконання певної кількості кроків. Іншими словами відомий алгоритм відшукання точного розв’язку задачі, але сам розв’язок не може бути записаний в аналітичній формі. Такий метод моделювання називають математичним моделюванням. Зауважимо, що алгоритм F відшукання точного розв’язку задачі може бути реалізований дослідником самостійно, за допомогою спеціального програмного забезпечення або за допомогою чисельних методів. Існують системи, опис яких не піддається опису аналітичними функціями, але процес функціонування їх може бути описаний алгоритмом імітації. Під імітацією розуміють відтворення

за допомогою комп'ютерної програми процесу функціонування складної системи в часі. У результаті багатократних прогонів імітаційної моделі дослідник отримує інформацію про властивості реальної системи. Такий метод моделювання називають імітаційним моделюванням. Стисло визначення методів моделювання представлені на рисунку 1.2.

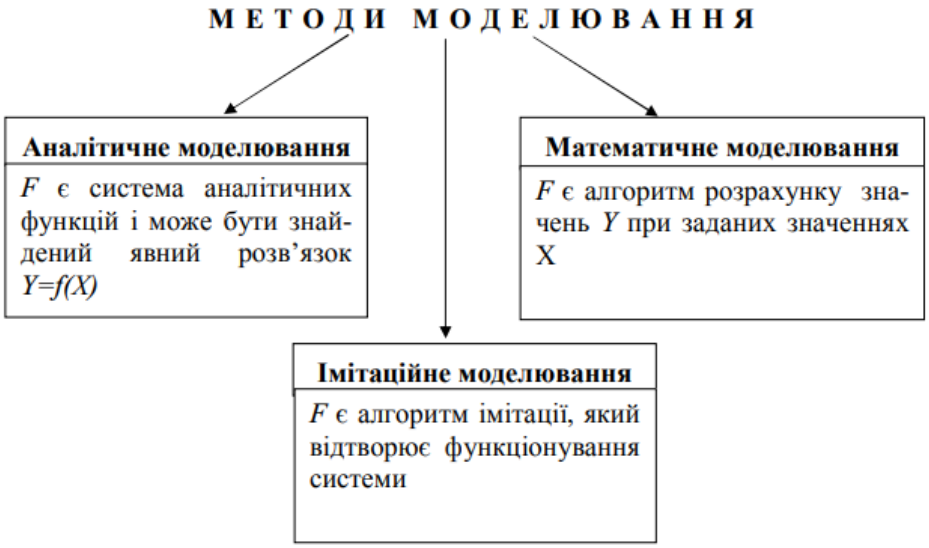


Рисунок 1.2 – Методи моделювання

1.4 Постановка задачі

Постановка задачі включає в себе декілька ключових етапів:

- Створення моделі обслуговування користувачів консультанта абітурієнта
- Створення системи контролю роботи консультанта абітурієнта
- Розробка програмного забезпечення
- Проведення експерименту на базі розробленої моделі обслуговування користувачів

Інформаційна структура «Консультант абітурієнта» приймальної комісії повинна бути створена за принципом вибір питання-отримання відповіді. Так само в неї повинні входити принципи зворотного зв'язку з користувачем, налагоджений метод зміни інформації, тобто актуальність інформації повинна бути постійною. Від цього фактору залежить чи буде «Консультант абітурієнта» корисним доповненням, або марною частиною майбутнього функціоналу приймальної комісії.

Питання та відповіді будуть формуватися заздалегідь. Це дозволяє в майбутньому використовуючи менше зусиль доповнювати відповідь на заздалегідь прописаним питання, або ж зовсім замінювати питання і відповіді.

На рисунку 1.3 зображено як будуть виконані питання та відповіді у «Консультант абітурієнта». Як видно з зображення, питання будуть формуватися у деякому пункті питання який в свою чергу містить певну кількість питань та відповідей, які реалізовані заздалегідь. В кваліфікаційній роботі магістра передбачено, що таких глобальних пунктів питань та відповідей буде що найменше три.

Зберігання питань-відповідей є важливою частиною роботи по розробці «Консультант абітурієнта». При обмірковувати можливих варіанти реалізації зберігання даних в роботі, було висунуто рішення реалізувати певну базу даних в якій буде зберігатися вся текстова та візуальна частина питань та відповідей на них. Зберігання та редагування такої бази даних буде реалізовано шляхом створення певного функціоналу у самому консультанті абітурієнта, до якого буде мати доступ тільки адміністратор.

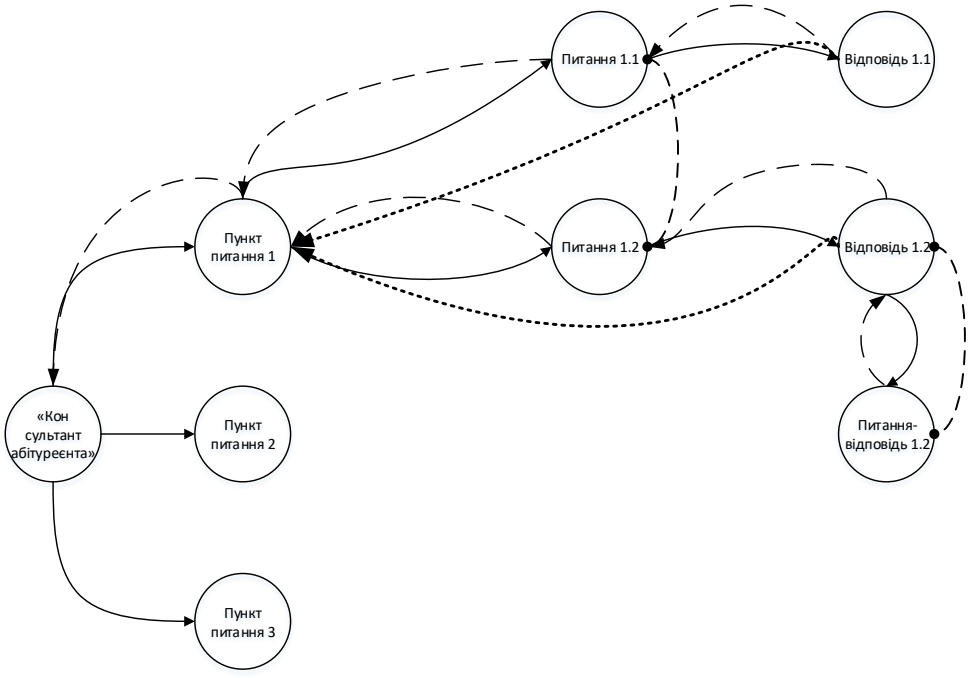


Рисунок 1.3 – Формування питань та відповідей

Для того щоб зрозуміти як має функціонувати комп’ютерна система «Консультант абітурієнта» було побудовано функціональну модель яка зображує взаємодію користувача з консультантом абітурієнта приймальної комісії.

На рисунку 1.4 можна побачити як само буде функціонувати консультант абітурієнта, а саме:

– запит користувача до консультанта, звернення комп’ютерної системи консультанта до створеної бази даних, відповідь на запит користувача;
– запит користувача до зворотного зв’язку, підключення працівника приймальної комісії до системи консультанта абітурієнта, відповідь на поставлене користувачем запитання.

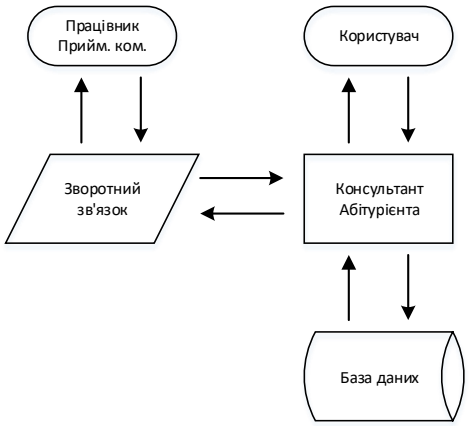


Рисунок 1.4 – Взаємодія користувача з «Консультантом абітурієнта»

Спираючись на дані функціонального та структурного аналізу, а також у зв'язку з тим, що всі можливості консультанта абітурієнта повинні бути реалізовані за допомогою Telegram Bot API програмування [10], а також розробки бази даних для отримання та відправки інформації можу затвердити, що кваліфікаційна робота магістра обґрунтовується доцільною розробкою комп'ютерної системи приймальної комісії Національного технічного університету «Дніпровська політехніка» з детальним опрацюванням бота «Консультант абітурієнта».

2 ТЕОРЕТИЧНИЙ РОЗДІЛ

2.1 Загальна характеристика комп'ютерної системи приймальної комісії

Об’єкт дослідження: Комп'ютерна система приймальної комісії Національного технічного університету «Дніпровська політехніка» з детальним опрацюванням бота «Консультант абітурієнта».

Загальна характеристика: «Консультант абітурієнта» приймальної комісії розробляється на базі існуючої системи створення, читання та відправлення повідомлень, мессенджера Telegram [5]. Система розроблюється за допомогою вбудованого прикладного програмного інтерфейсу Telegram Bot API [10] на програмній платформі Node.js [11] та завдяки мові програмування JavaScript [12]. Комп’ютерна система представляє собою консультанта для абітурієнтів який допомагає прийняти рішення, що до спеціальності навчання, а також допомагає у вирішені основних питань до вступної кампанії університету за допомогою своїх функцій.

Доступ до консультанта абітурієнта мають всі бажаючі користувачі з смартфона та з персонального комп’ютера користувача. Для можливого користування консультантом абітурієнта потрібно скористатися офіційним сайтом Telegram [5], встановити існуючий додаток до смартфона або встановити програмне забезпечення на персональний комп’ютер.

Функціонування «Консультанта абітурієнта» буде підтримувати існуючий сервер у головному офісі приймальної комісії. Консультант абітурієнта розробляється з можливостями адміністрування та модерації користувачів які мають на увазі редагування інформації та видалення інформації, можливість зворотного зв’язку з представниками приймальної комісії.

2.2 Обґрунтування і вибір методів дослідження

Дослідження «Консультанта абітурієнта» приймальної комісії Національного технічного університету «Дніпровська політехніка» обґрунтовується тим, що об'єкт дослідження досить новий і має недостатній дослідницький інтерес. При розробці подібних об’єктів, дослідження проводяться вкрай рідко, в недостатньому обсязі і недостатньо інформативно.

Вибором метода дослідження являється система масового обслуговування [13]. Завдяки цьому методу можна буде детально дослідити комп’ютерну систему консультанта абітурієнта яка здійснює обслуговування вимог, що до неї надходять, та дати висновок доцільності системи що створюється.

Теорія Систем масового обслуговування (СМО) присвячена розробці методів аналізу, проектування і раціональної організації систем, що відносяться до різних областей діяльності.

СМО (системи масового обслуговування) – це моделі систем, в які в випадкові моменти часу ззовні або зсередини надходять заявки (вимоги). Вони повинні тим чи іншим чином бути обслужені системою. Тривалість обслуговування найчастіше випадкова. СМО являє собою сукупність обслуговуючого обладнання і персоналу при відповідній організації процесу обслуговування.

Завдання аналізу СМО полягає у визначенні ряду показників її ефективності, які можна розділити на наступні групи:

- показники, що характеризують систему в цілому: число зайнятих каналів обслуговування, число обслужених, які очікують обслуговування або отримали відмову заявок в одиницю часу;
- імовірнісні характеристики: ймовірність того, що заявка буде обслужена або отримає відмову в обслуговуванні, що всі прилади вільні або певне число їх зайнято, ймовірність наявності черги;

– економічні показники: вартість втрат, пов'язаних з доглядом не обслужених з тих чи інших причин заявок з системи, економічний ефект, отриманий в результаті обслуговування заявки.

Частина технічних показників (перші дві групи) характеризують систему з точки зору споживачів, інша частина характеризує систему з точки зору її експлуатаційних властивостей. Часто вибір перерахованих показників, може покращувати експлуатаційні властивості системи, а й погіршувати систему з точки зору споживачів і навпаки. Використання економічних показників дозволяє вирішити зазначене протиріччя і оптимізувати систему з урахуванням обох точок зору.

Основними елементами моделі масового обслуговування є клієнт (Заявка або вимога на обслуговування, або просто "об'єкт обслуговування") і сервіс (обслуговує пристрій, засоби обслуговування і т.п.). клієнти надходять в систему обслуговування з джерела. Надійшовши в сервіс, вони можуть відразу ж потрапити на обслуговування або очікувати в черзі, якщо сервіс зайнятий. Після завершення процедури обслуговування сервіс автоматично "вибирає" з черги (якщо вона є) одного з клієнтів з тим, щоб приступити до його обслуговування. Якщо ж черга відсутня, то сервіс стає незайнятим до прибуття нового клієнта.

Надходження клієнтів в систему обслуговування характеризується інтервалом між їх послідовними надходженнями, а обслуговування - часом обслуговування клієнта. У загальному випадку ці параметри можуть бути і випадковими як, наприклад, в роботі поштового відділення, і детермінованими, як, наприклад, прибуття на співбесіду кандидатів на вакантну посаду.

В аналізі систем обслуговування певну роль відіграє довжина черги, яка може бути кінцевою, як в буферній зоні між двома послідовними обслуговуючими пристроями, і нескінченною, як у обслуговуючих операторів поштових відділень.

Важливим фактором при аналізі систем обслуговування є дисципліна черги (або принцип побудови черги), яка визначає порядок, відповідно до якого

вибираються клієнти з черги для обслуговування. Найбільш поширений принцип побудови черги заснований на правилі "першим прийшов - першим обслуговується", це правило часто позначається абревіатурою FIFO – від англійського First-In-First-Out. Серед інших правил, що визначають принципи побудови черг, введемо правило "останнім прийшов - першим обслуговується" зазвичай позначається як LIFO – від англійського Last-In-First-Out і дисципліну черги, яка визначається випадковим правилом відбору клієнт, іноді позначається як SIR0 – від англійського Service-In-Random-Out. Крім того, клієнти можуть вибиратися з черги відповідно до заданого пріоритету. Наприклад, у виробничому цеху термінові роботи виконуються раніше звичайних.

При аналізі систем з чергами важливим фактором є також поведінка індивідуума, що потребує обслуговування. Такі індивідууми, які виступають в ролі клієнтів, при наявності паралельного обслуговування можуть перейти з однієї черги до іншої в надії скоротити тривалість свого вимушеного очікування. Вони можуть також відмовитися від очікування в черзі, так як люди зазвичай не переносять тривалої бездіяльності, або покинути чергу, простоявши в ній якийсь час і прийшовши до висновку, що і так вже занадто багато часу втрачено.

Структура обслуговуючої системи може включати один сервіс або кілька таких засобів обслуговування, що працюють паралельно (наприклад, робота декількох клерків поштового відділення). Крім того, сервіси можуть бути розташовані послідовно (наприклад, обслуговування являє собою комплекс робіт, які виконуються послідовно на різних верстатах).

Джерело, що генерує клієнтів, які можна обслуговувати, може мати кінцеву або нескінченну потужність. Джерело кінцевої потужності обмежує число клієнтів, що надходять на обслуговування і навпаки, джерело нескінченної потужності завжди має клієнтів в достатку. Можна побудувати безліч моделей систем масового обслуговування, варіюючи перераховані вище операційні характеристики систем.

2.3 Загальна модель системи масового обслуговування

В даному розділі розглядаються загальні системи масового обслуговування [13], в яких є як вхідний потік клієнтів, так і вихідний потік обслужених клієнтів. Час між послідовними надходженнями клієнтів і час обслуговування є експоненціально розподіленими випадковими величинами.

При розгляді загальних систем масового обслуговування передбачається, що система функціонує протягом досить великого інтервалу часу, після закінчення якого в її роботі настає стаціонарний режим. Цей режим функціонування обслуговуючої системи протиставляється перехідному (або несталий) режиму, який превалює в самий початковий період функціонування системи.

У розглянутій в цьому розділі загальної моделі системи масового обслуговування передбачається, що інтенсивність надходження клієнтів, та інтенсивність вихідного потоку залежать від стану системи, що означає їх залежність від числа клієнтів в системі обслуговування. Наприклад, збирач плати за проїзд по автомагістралі в годинник інтенсивного руху прагне прискорити збір мита. Або в майстерні з фіксованою кількістю верстатів інтенсивність їх поломки зменшується в міру зростання числа аварійних верстатів, бо лише працюючі верстати можуть виходити з ладу.

Введемо наступні позначення.

n - число клієнтів в системі обслуговування (в черзі і на обслуговуванні);

λ_n - інтенсивність надходження в систему клієнтів за умови, що в системі вже знаходиться n клієнтів;

μ_n - інтенсивність вихідного потоку обслужених клієнтів за умови, що в системі знаходиться n клієнтів;

P_n - ймовірність того, що в системі знаходиться n клієнтів.

У загальній моделі системи масового обслуговування встановлюється функціональна залежність ймовірностей P_n від λ_n і μ_n. Ці ймовірності використовуються потім при визначенні функціональних характеристик

обслуговуючої системи, таких як середня довжина черги, середній час очікування і середній коефіцієнт використання сервісів.

Ймовірності P_n визначається з діаграми інтенсивностей переходів, представленої на рисунку 2.1. Обслуговуюча система знаходиться в стані n, якщо в ній є n клієнтів. Ймовірність появи більше ніж одного нового клієнта протягом малого проміжку часу h прагне до нуля при h → 0. Це означає, що при n > 0 стан n може бути змінено в двох можливих напрямках: n - 1, коли з інтенсивністю μ_n обслужений клієнт вибуває з системи, і n + 1, коли клієнти надходять з інтенсивністю λ_n. Стан 0 може змінитися лише до стану 1, коли має місце надходження клієнта з інтенсивністю λ_0. Зауважимо, що μ_0 не визначене, тому, що клієнти не можуть вибувати з порожньої системи обслуговування.

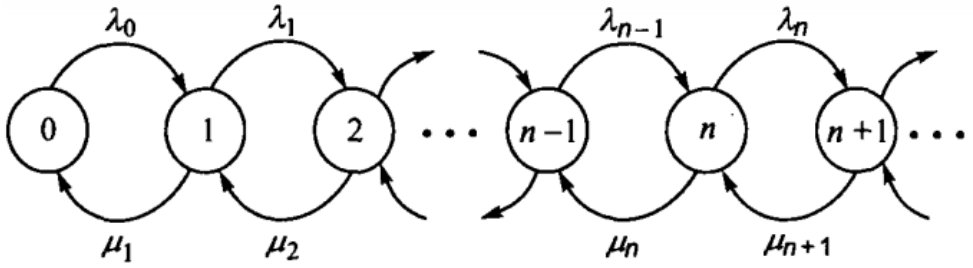


Рисунок 2.1 – Діаграма інтенсивностей переходу

При виконанні умов стаціонарності очікувані інтенсивності вхідного і вихідного потоків в стані n (n > 0) повинні бути рівні. Так як стан n може змінюватися лише до стана n – 1 та n + 1, звідси випливає що:

Очікувана інтенсивність вхідного потоку в стані n = $\lambda_{n-1}P_{n-1} + \mu_{n+1}P_{n+1}$.

Аналогічно

Очікувана інтенсивність вихідного потоку в стані n = $(\lambda_n + \mu_n)\, P_n$.

Прирівнюючи ці дві інтенсивності, отримуємо наступне рівняння балансу.

$$\lambda_{n-1}P_{n-1} + \mu_{n+1}P_{n+1} = (\lambda_n + \mu_n)\, P_n,\ n = 1,2, \ldots \quad (2.1)$$

Як видно з рисунку 2.1, рівняння балансу, відповідне n = 0, має вигляд

$$\lambda_0 P_0 = \mu_1 P_1. \quad (2.2)$$

Рівняння балансу вирішуються рекуррентно, послідовно висловлюючи ймовірності P_1 черезр P_0 наступним чином: для n = 0 маємо

$$P_1 = \left(\frac{\lambda_0}{\mu_1}\right) P_0. \quad (2.3)$$

Для n = 1 отримаємо

$$\lambda_0 P_0 + \mu_2 P_2 = (\lambda_1 + \mu_1)\, P_1. \quad (2.4)$$

Підставляючи сюди $P_1 = (\lambda_0/\mu_1)\, P_0$ і спрощуючи отриманий вираз, маємо

$$P_2 = \left(\frac{\lambda_1 \lambda_0}{\mu_2 \mu_1}\right) P_0. \quad (2.5)$$

Методом індукції можна показати, що

$$P_n = \left(\frac{\lambda_{n-1}\lambda_{n-2}\ldots\lambda_0}{\mu_n \mu_{n-1}\ldots\mu_1}\right) P_0, n = 1,2,\ldots \quad (2.6)$$

Значення P_0 визначається з рівняння $\sum_{n=0}^{\infty} P_n = 1$.

2.4 Створення моделі СМО у комп’ютерній системі

Моделювання системи масового обслуговування [13] у комп’ютерної системи «Консультант абітурієнта» приймальної комісії НТУ «ДП» обумовлюється наявністю сервісно-користувацьких відношень у створюваній системі. Вирішення завдання створення СМО моделі є необхідним кроком, для розвитку подальшої розробки програмного забезпечення, та проведення експерименту на базі створеної моделі. Розробка моделі масового обслуговування комп’ютерної системи починається з постановки задач та оголошення змінних.

Постановка задачі:

Спираючись на інформацію з попередніх пунктів розділу, а так само на зовнішні джерела інформації, потрібно створити СМО модель для комп'ютерної системи "консультант абітурієнта" яка повинна розраховувати в процентному еквіваленті кількість користувачів які залишилися в системі обслуговування за певний проміжок часу, тобто являються ще не обслуженими з певних причин. На рисунку 2.2 зображена структурна схема системи обслуговування комп’ютерної системи «Консультанта абітурієнта».

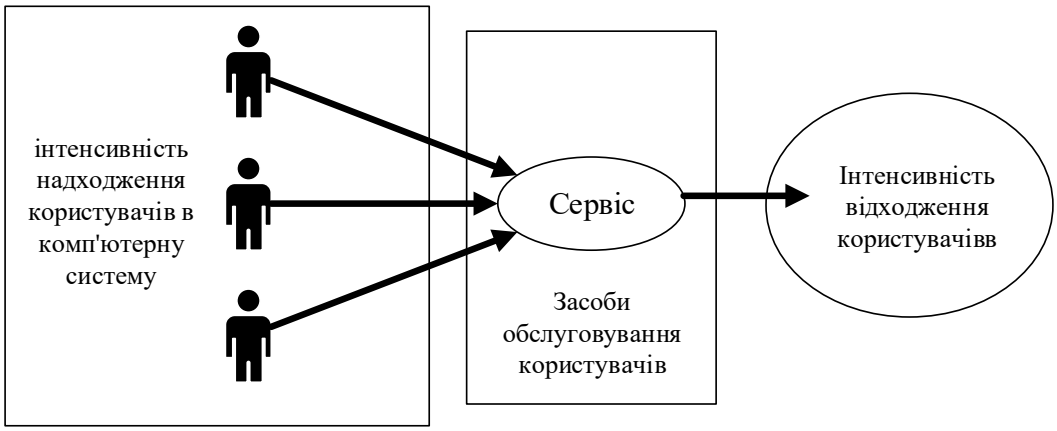


Рисунок 2.2 – Структурна схема системи обслуговування

Введемо наступні позначення.

n - число користувачів в комп’ютерній системі «Консультант абітурієнта»;

λ_n - інтенсивність надходження в комп’ютерну систему користувачів за умови, що в системі вже знаходиться n користувачів;

μ_n - інтенсивність вихідного потоку обслужених користувачів за умови, що

в системі знаходиться n користувачів;

P_n - ймовірність того, що в системі знаходиться n користувачів.

За формулою 2.7 буде створена математична модель систем масового обслуговування комп’ютерної системи «Консультанта абітурієнта»

$$P_n = \left(\frac{\lambda_{n-1}\lambda_{n-2}\dots\lambda_0}{\mu_n\mu_{n-1}\dots\mu_1}\right)P_0 \qquad (2.7)$$

Створення СМО відбувається у середовищі MATLAB [14] за допомогою інтерактивного інструменту для моделювання, імітації та аналізу динамічних систем – Simulink [15]. Розроблене компанією The MathWorks. Дає можливість будувати графічні блок-діаграми, імітувати динамічні системи, досліджувати працездатність систем і вдосконалювати проекти. Simulink повністю інтегрований з MATLAB, що забезпечує швидкий доступ до широкого спектра інструментів аналізу і проектування. На рисунку 2.3 зображено модель СМО комп’ютерної системи.

Опис моделі:

блок n – константа, визначається у ручному режимі, потрібна для визначення кількості користувачів в комп’ютерній системі;

блок Subsystem – підсистема моделі СМО, зображена на рисунку 2.4;

блок Display – виведення результату.

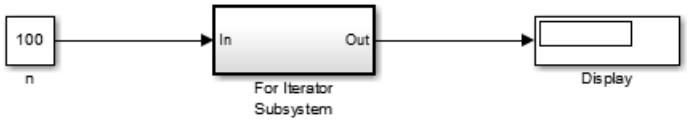


Рисунок 2.3 – Модель СМО комп’ютерної системи

На рисунку 2.4 зображено підсистему моделі СМО комп’ютерної системи

Опис підсистеми моделі:

блок In – вхід підсистеми;

блок Data Store Memory (A,B) – зберігає всі дані під час моделювання системи;

блок Data Store Write (A,B) – записує всі дані під час моделювання системи;

Блок Data Store Read (A,B) – зчитує всі дані під час моделювання системи;

блок Uniform Random Number – генерація випадкового сигналу в діапазоні від 0 до 100;

блок Data Type Conversion (int8) – використовується для перетворення нецілих чисел в дійсні числа;

блок Product – використовується для множення або ділення чисел, які приходять на вхід блоку;

блок Constant - застосовується для використання статичного числа;

блок Scope – Осцилограф, який застосовується для виведення графіку на екран;

блок Out – вихід підсистеми.

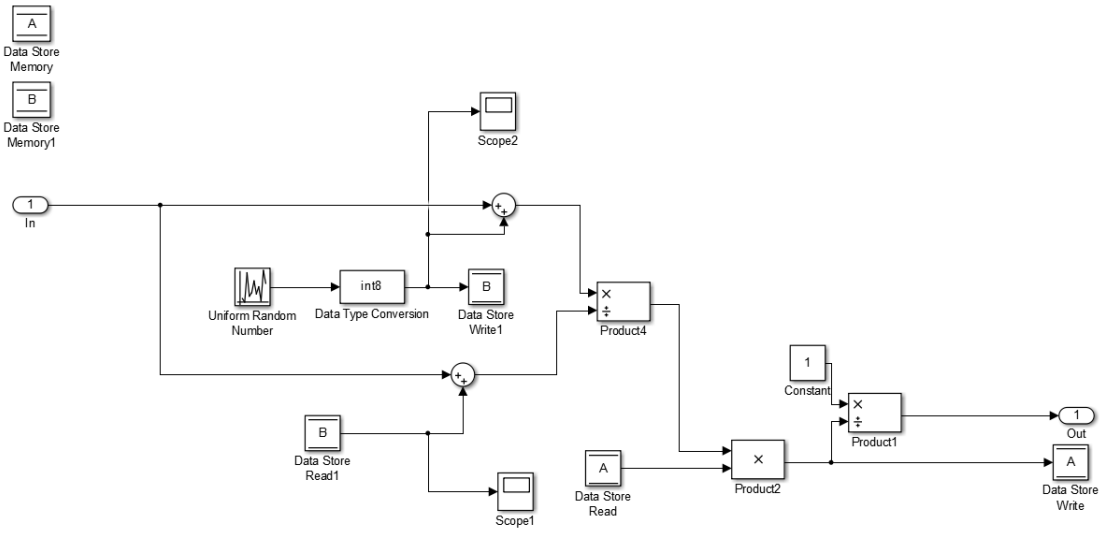


Рисунок 2.4 Підсистема моделі СМО комп’ютерної системи

2.5 Висновки по розділу

За науковим завданням кваліфікаційної роботи магістра до приймальної комісії Національного технічного університету «Дніпровська політехніка» впроваджуються комп’ютерна система «Консультанта абітурієнта» з детальним опрацюванням бота який буде надавати актуальну інформацію про вступну кампанію приймальної комісії, консультувати абітурієнтів, що до вибору спеціальностей та інших важливих деталей. В цьому розділі описується і обґрунтовується доцільність наукового завдання по створенню моделі майбутньої комп’ютерної системи. Проводиться вибір системи за допомогою якої можна буде провести моделювання створюваної комп’ютерної системи, а саме створення моделі системи масового обслуговування для комп’ютерної системи. У пунктах розділу було описано і обґрунтовано вибір моделі системи, а кінцевим етапом даного розділу являється створення моделі системи масового обслуговування для комп’ютерної системи «Консультант абітурієнта».

3 СИНТЕЗ СИСТЕМИ КОНТРОЛЮ

3.1 Вибір і обґрунтування принципів побудови проектованої комп’ютерної системи

Побудова комп'ютерної системи включає в себе формування технічних вимог, функціональних вимог , вимог до видів забезпечення та вимог до захисту інформації на базі яких буде зроблений вибір устаткування для створення структурної схеми комп'ютерної системи «Консультанта абітурієнта» приймальної комісії. Проектування комп'ютерної системи здійснюватиметься з урахуванням створеної структурної схеми та схеми функціональної структури.

Вибір саме таких принципів побудови комп'ютерної системи консультанта абітурієнта обґрунтовується специфікою створення і обслуговування та функціонального призначення ботів. Для створення такої комп'ютерної системи буде потрібно певне апаратне рішення яке у подальшому буде описано у наступних пунктах цього розділу.

Комп'ютера система «Консультанта абітурієнта» приймальної комісії Національного технічного університету «Дніпровська політехніка» повинна виконувати наступні функції:

- надавати інформацію для відвідувачів;
- пошук інформації;
- створювати інформаційні блоки;
- видаляти інформацію;
- зберігати інформаційні блоки у базі даних;
- редагувати інформацію;
- надавати зворотний зв'язок для користувачів.

Виходячи з переліку приведених функцій буде створено схему функціональної структури, яка зображена на рисунку 3.1.

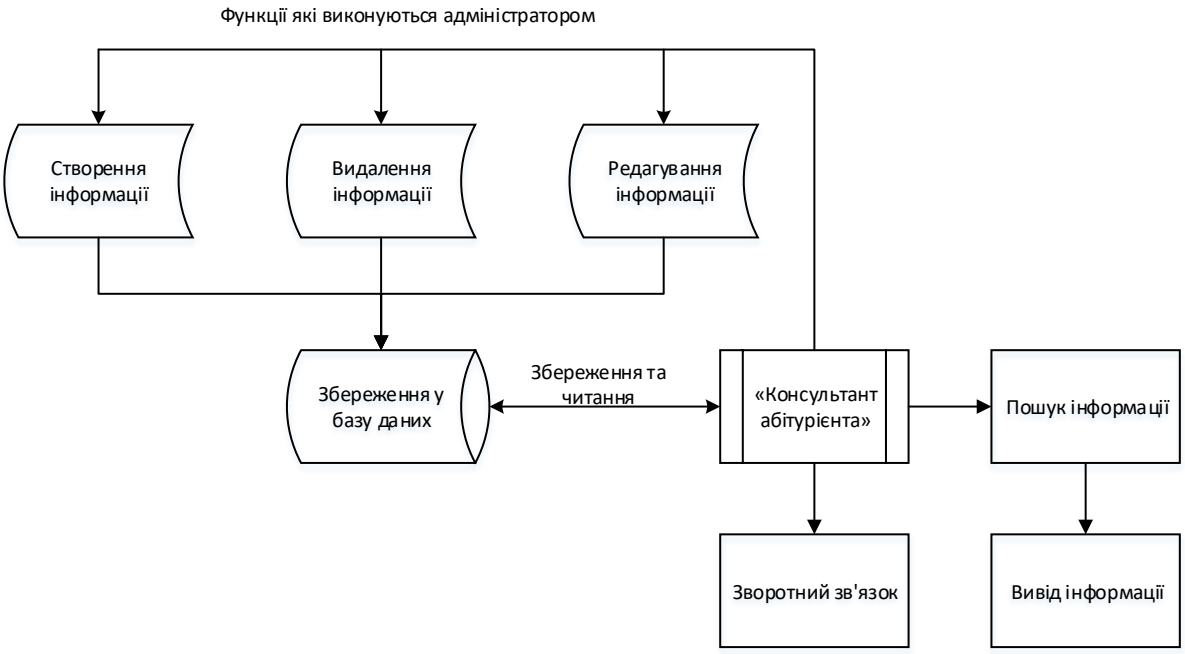


Рисунок 3.1 – Функціональна структура комп’ютерної системи «Консультанта абітурієнта»

3.2 Формулювання технічних вимог до комп’ютерної системи «Консультант Абітурієнта»

3.2.1 Вимоги до системи в цілому

3.2.1.1 Вимоги до реалізації комп’ютерної системи

Комп’ютерна система «Консультант абітурієнта» приймальної комісії повинна бути розроблена виключно за допомогою мови Telegram Bot API [10] програмування, для досягнення задуманих функцій консультанта абітурієнта, а також з міркувань безпеки системи, яка буде встановлена на сервер приймальної комісії. При написанні коду розробник консультанта абітурієнта повинен мати досвід у розробці такого роду додатків, розуміти мову Telegram Bot API програмування, та функціонал яку вона пропонує. Розробник консультанта абітурієнта повинен представити схему алгоритму програми, а також схеми алгоритму підпрограм. До обов’язків програміста також входить розробка дизайну консультанта абітурієнта, розробник повинен дотримуватися загального стилю університету, доповнюючи його, а не виділяти консультанта абітурієнта з загального стилю навчального закладу. Вибір мови Telegram Bot

API програмування, а також платформи реалізації коду програми буде представлений у другому розділі кваліфікаційної роботи магістра

3.2.1.2 Вимоги до функцій виконуваних системою

Комп'ютерна система «Консультанта абітурієнта» приймальної комісії Національного технічного університету «Дніпровська політехніка» має завжди функціонувати безперервно мати зручний та зрозумілий інтерфейс.

«Консультанта абітурієнта» повинен мати наступний функціонал:

- поле привітання абітурієнта;
- меню вибору дії, у ньому повинно бути реалізовано список доступних дії (кнопок) для користувача з наступними пунктами-виконувачами:
    - ВСТУП на Бакалавра на базі ПЗСО;
    - ВСТУП на Бакалавра на базі ОКР;
    - ВСТУП на Магістра;
    - Зворотний зв'язок;
- кожен з виконувачів має виводити повідомлення про вибрану функцію, а також містити в собі ще одні пункти-виконувачі:
    - Опис спеціальностей;
    - Основна інформація;
- пункт «Опис спеціальностей» при натисканні на нього повинен виводитись список всіх доступних спеціальностей, користувач зможе ввести код спеціальності в поле вводу після чого буде виведена детальна інформація про спеціальність, при виводі списку спеціальностей користувачеві буде представлена інструкція користування функціоналом;
- пункт «Основна інформація» має виводити всю загальну інформацію, а також має за собою індивідуальний пункт-виконувач який виводить інформацію що до вартості навчання за кожною спеціальністю;

– кожний пункт-виконувач повинен мати дії зворотного шагу та дії повернення до головного меню «Консультанта абітурієнта»;

пункт зворотний зв’язок повинен з’єднувати користувача «Консультанта абітурієнта» з працівником приймальної комісії для подальшої онлайн-консультації.

3.2.2 Вимоги до видів забезпечення

За результатами що одержані у попередніх розділах формуються наступні технічні вимоги.

Для того щоб створити «Консультанта абітурієнта», необхідно мати персональний комп’ютер, або ноутбук з мінімальними характеристиками які вказані в таблиці 3.1.

Таблиця 3.1 Мінімальні характеристики персонального комп’ютера

Тип	Найменування
Процесор	Intel Pentium N4200
Оперативна пам’ять	4 ГБ
Жорсткий диск	500 ГБ
Відео адаптер	Intel HD Graphics 505
Операційна система	Windows 7/8/10

Так як для Telegram Bot API [10] програмування непотрібно багато потужностей, то можна сказати, що персональний комп’ютер у якого характеристики кращі за мінімально вказані у таблиці 3.1 підходить для розробки консультанта абітурієнта.

Для користувачів «Консультанта абітурієнта» приймальної комісії мінімальні характеристики ті ж самі, що і для розробника програмного-додатку, що вказані у таблиці 3.1.

Для того, щоб консультант абітурієнта працював, його необхідно встановити на сервер. Для цього підходить будь-яке спеціалізоване устаткування з наступними мінімальними характеристиками:

- процесор Intel Xeon E5-2603v3;
- оперативна пам'ять від 8 ГБ DDR4-2133;
- об’єм пам’яті від 256 ГБ;
- мережевий адаптер Ethernet 10Gb 2-port 560SFP;
- контролери та опції H241 12Gb 2-ports Ext Smart Host Bus Adapter.

3.2.3 Вимоги до захисту інформації

При розробці комп’ютерної системи дані будуть зберігатися у відділеному сервері бази даних, доступ до якої буде відкрито у обмеженої кількості персоналу. Захист інформації буде реалізовано за допомогою серверних технологій захисту інформації таких як SSH-ключі, фаєрвол, PKI та SSL/TLS шифрування. На розробку додаткових функцій захисту інформації можна витратити багато часу і людських сил. Але з огляду на те що, захистом даних займаються технологій захисту інформації та серверний адміністратор, то в розробці додаткових систем захисту інформації «Консультанта абітурієнта» немає необхідності.

3.3 Вибір обладнання

Створення комп’ютерної системи «Консультанта абітурієнта» приймальної комісії передбачає собою розгляд існуючих рішень апаратної частини і вибір оптимального варіанту для проведення його подальшої розробки, впровадження у мережу Національного технічного університету «Дніпровська політехніка» та подальшої підтримки безперебійного функціонування впровадженої комп’ютерної системи.

Обладнання для створення комп’ютерної системи повинно бути сучасним та надійним, цього вимагають новітні тенденції розробки комп'ютерних систем. З огляду на раніше описані вимоги, складається список з необхідного

апаратного забезпечення яке прийматиме участь в розробці, впровадженні та підтримці комп'ютерної системи

1. Для створення комп’ютерної системи консультанта абітурієнта потрібно мати персональний комп’ютер який має можливість виходу до глобальної мережі інтернет та відповідає вимогам використовуваного програмного забезпечення для створення «Консультанта абітурієнта». Рекомендується використовувати персональний комп’ютер типу (Рисунок 3.1) ARTLINE Business B26 v14 [16]. У таблиці 3.2 наведені апаратні характеристики персонального комп’ютера.

Рисунок 3.2 – ARTLINE Business B26 v14

Таблиця 3.2 – Апаратні характеристики ARTLINE Business B26 v14

Модель процесора	Intel 2-Core i3-6320 3.9GHz
Відеокарта	Intel HD
Оперативна пам’ять	8Gb DDR4-2400
Об’єм накопичувача	120GB SSD
Модель материнської плати	H310MHP
Корпус	QUBE QB05M U3
Блок живлення	400W
Охолодження процесора	BOX
Операційна система	Windows 10

2. Основні функції збереження даних, передачі файлів та IP – адресації виконуються на серверах (Рисунок 3.2) DELL R730XD (24x2.5) SFF [17] які встановлено у першому корпусі НТУ «ДП». Їх наявність обумовлюється створенням надійної системи підтримки стабільної роботи комп'ютерної системи консультанта абітурієнта.

Рисунок 3.3 – DELL R730XD (24x2.5) SFF

Збереження інформації у комп'ютерній системі виконується на сервері бази даних.

Сервер баз даних виконує обслуговування та управління базою даних та відповідає за цілісність та збереження даних, а також забезпечує операції введення-виведення при доступі клієнта до інформації. У таблиці 3.3 наведені апаратні характеристики Сервера бази даних.

IP – адресація у мережі НТУ «ДП» виконується за допомогою виділеного сервера DHCP.

Сервер DHCP (Dynamic Host Configuration Protocol) запускає однойменну службу, яка автоматично розподіляє IP-адреси між пристроями, підключеними до однієї мережі в межах внутрішнього середовища. DHCP – сервер управляє IP-адресами в локальній мережі і займається автоматичним присвоєнням індивідуальної динамічної IP-адреси кожному новому пристрою. Це дозволяє налаштовувати параметри клієнта мережі безпосередньо на сервері, а не на окремих пристроях. Адреси роздаються мережевим пристроям на певний проміжок часу. У таблиці 3.3 наведені апаратні характеристики Сервера DHCP.

Передача файлів у комп’ютерній системі консультанта абітурієнта виконується за допомогою сервера FTP.

FTP сервер – це сервер, що працює по File Transfer Protocol (протоколу передачі файлів). Використовується для обміну файлами між комп'ютерами по локальній мережі і інтернету, ця технологія є однією з найбільш затребуваних для скачування і завантаження даних з віддалених серверів, або на них. FTP Протокол побудований на архітектурі «клієнт-сервер» і використовує різні мережеві з'єднання для передачі команд і даних між клієнтом і сервером. У таблиці 3.2 наведені апаратні характеристики Сервера FTP

Таблиця 3.3 – Апаратні характеристики DELL R730XD (24x2.5) SFF

Модель процесора	2 x Intel XEON Six Core E5-2620 V3 2.40GHz(SR207)
Дисковий контролер	RAID-Контроллер PERC H330
Оперативна пам’ять	32GB (4x8GB) DDR4 ECC Registered
Накопичувачі	2x SSD INTEL SATA 2.5" 1.92TB 5x Western Digital Gold HDD SATA 6TB
Мережева карта	DELL 2 x 10G SFP+ & 2x 1G RJ45 Intel X520/i350
Блок живлення	2 шт. Блок питания DELL R730/R630/R530(495W/750W)

3.4 Синтез структурної схеми системи за заданими показниками комп’ютерної системи

З огляду на попередні пункти розділу створюється структурна схема комп’ютерної системи (Рисунок 3.4) яка описує роботу консультанта абітурієнта у студентському містечку Національного технічного університету «Дніпровська політехніка» та по за його межами. На рисунку 3.4 можна побачити взаємодію користувача з безпровідними точками доступу, які встановлені по всій території НТУ «ДП». У першому корпусі встановлено сервер бази даних, сервер DHCP та сервер FTP, вони взаємодіють з персональними комп’ютерами приймальної комісії, які встановлено у четвертому корпусі для надання серверної підтримки комп’ютерної системи.

Доступ до консультанта абітурієнта має будь який користувач мережі НТУ «ДП» та будь якої іншої мережі поза межами студентського містечка.

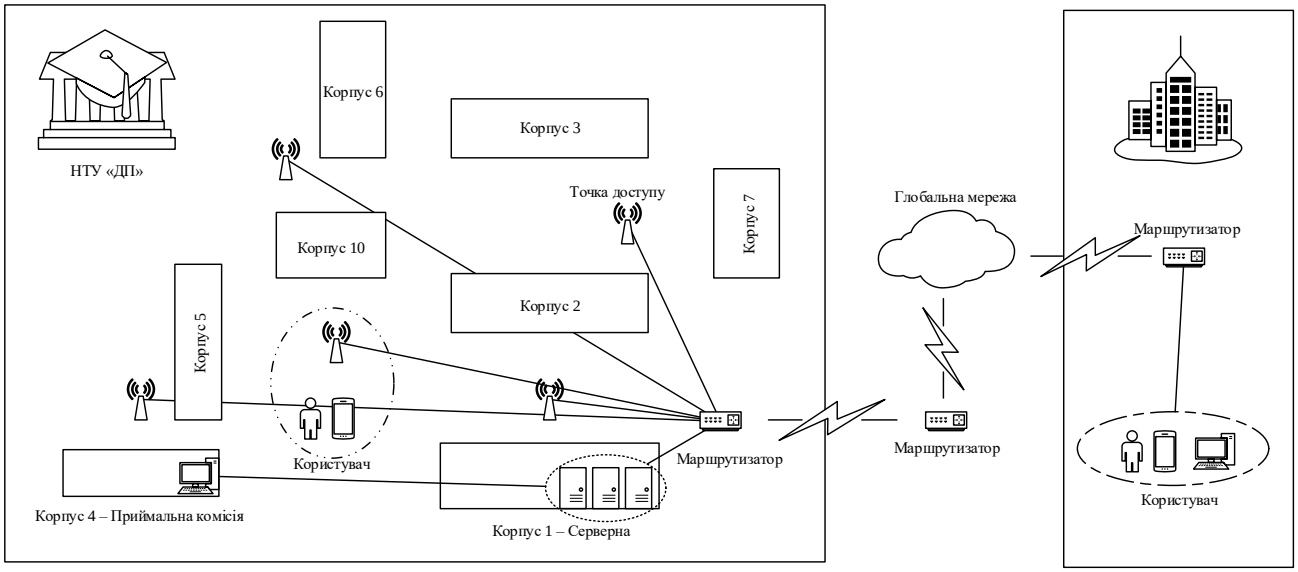


Рисунок 3.4 – Структурна схема комп’ютерної системи «Консультанта абітурієнта»

3.5 Висновки по розділу

При створенні системи контролю в розробленій комп'ютерній системі "Консультант абітурієнта" були враховані всі поставлені вимоги для реалізації системи контролю. Створення системи контролю включало в себе вибір оптимальних, сучасних існуючих рішень апаратних частини системи контролю для розробки програмного забезпечення комп'ютерної системи і її підтримки в майбутньому. Створення системи контролю комп'ютерної системи було розпочато з висування пропозиції апаратного рішення з описом його докладних характеристик.

Для підтримки комп'ютерної системи вибір був зроблений на користь реалізації трьох серверів з серверними технологіями бази даних, протоколу динамічної конфігурації IP-адрес і протоколу передачі файлів. Кінцевим етапом даного розділу являється створення структурної схеми системи контролю на якій показані засоби управління комп'ютерною системою, а так само їх розташування.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ «КОНСУЛЬТАНТА АБІТУРІЄНТА»

4.1 Призначення й сфера застосування програми

Основне призначення розробленої програми – це поліпшення роботи приймальної комісії Національного технічного університету «Дніпровська політехніка», оптимізація роботи приймальної комісії по відношенню до абітурієнтів, орієнтуючись на зростання сучасних тенденцій, спираючись на мессенджери та їх зростаючий потенціал функціональних можливостей, а також покращення доступності шуканої інформації.

Сфера застосування програмного забезпечення, це консультаційні послуги користувачам мессенджера Telegram. Надання послуг онлайн-консультацій, зворотного зв'язку, для роз'яснення користувачам «Консультанта абітурієнта» вступної компанії навчального закладу та основних напрямків підготовки спеціалістів у вищому навчальному закладі.

4.2 Обґрунтування технічних характеристик програми

Розробка програмного забезпечення комп'ютерної системи «Консультанта абітурієнта» приймальної комісії НТУ «ДП» починається з постановки задач.

Програмне забезпечення яке розробляється для комп'ютерної системи консультант абітурієнта повинно бути розроблено на сучасній мові програмування з урахуванням всіх аспектів доцільності та гнучкості у використанні, та майбутньої модернізації.

Програмне забезпечення повинно бути уніфікованим, тобто процес створення програмного коду повинен бути зрозумілим і приведеним до однакової системи. Функціональний запас програмного забезпечення повинен включати в себе повний спектр можливостей налаштування, для найбільшої ефективності використання програмного забезпечення та його налаштування.

Розробка програмного забезпечення повинна проводитися на сучасному обладнанні при участі сучасних платформ для реалізації програмного коду.

Розробка програмного забезпечення бере свій початок з реалізації схем алгоритму основної програми та їх підпрограм. Схема алгоритму призначена для відображення принципу роботи певного алгоритму програми комп'ютерної системи. Вони описують дію алгоритму так, як його можна реалізувати на будь-якій мові програмування, в наслідку розробки схеми алгоритму створюється повне розуміння процесу, що відбувається.

Схема алгоритму основної програми:

На рисунку 4.1 зображена схема алгоритму, що описує роботу загальної структури програми, яка містить в собі функції «Консультанта абітурієнта» які виводяться на екран користувача.

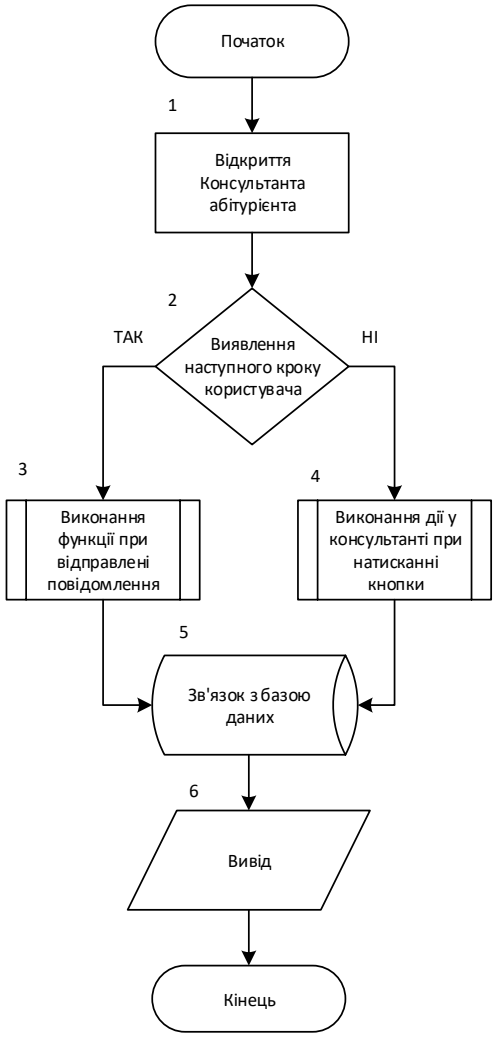


Рисунок 4.1 – Схема алгоритму основної програми

На рисунку 4.2 зображена схема алгоритму, що описує роботу виводу меню. В першому блоці схеми алгоритму відбувається вивід пропозиції початку роботи з «Консультантом абітурієнта» користувачеві, ця функція має два можливих початки перший – це виводиться кнопка «СТРАРТ» на яку потрібно натиснути, друга – це можливість вводу команди /Start для початку роботи, або вибір цієї команди у спеціальному меню команд. У другому блоці умови описується підтвердження початку роботи, якщо НІ то роботу з «Консультантом абітурієнта» неможливо буде розпочати, якщо ТАК то робота буде розпочата, тобто користування консультантом абітурієнта буде підтверджено. У третьому блоці відображена дія програми якщо підтверджується робота з консультантом абітурієнта та здійснюється вивід основного меню.

Рисунок 4.2 – Схема алгоритму виводу меню

Наступний етап – це вибір мови програмування за допомогою якої буде здійснюватися розробка програмного забезпечення «Консультанта абітурієнта» приймальної комісії НТУ «ДП». Для реалізації задачі програмування потрібно використовувати популярні мови програмування, наприклад: PHP, Python, Node is, Go. Передбачається, що для роботи над програмним забезпеченням потрібно використовувати найбільш відповідну мову програмування.

Після вибору мови програмування потрібно здійснити вибір у користь сучасної платформи, на якій буде написано програмне забезпечення. У випадку, якщо мова програмування вже відома можна вибрати серед тих платформ реалізації програмного коду, які мають змогу підтримувати написання коду саме вибраної мови програмування найкраще, а саме: Aptana Studio, Sublime Text, Atom, Webstorm, Visual Studio, Spket IDE. Платформа для реалізації коду повинна мати наступний мінімальний функціонал: вбудований відладчик, термінал, синхронізація по FTP, можливість командної розробки. Повинен включати в себе стандартні функції для додатків з цієї області (автодоповнення, перевірка помилок і т.п). Платформа реалізації коду повинна працювати на всіх основних операційних системах.

Створення і підтримка програмного забезпечення не повинно використовувати забагато технічних ресурсів у таблиці 4.1 наведені мінімальні вимоги до реалізації та підтримки «Консультанта абітурієнта» приймальної комісії

Таблиця 4.1 – Мінімальні вимоги

Модель процесора	Від 2.40GHz
Оперативна пам’ять	Від 1GB вільного місця
Накопичувачі	SSD накопичувач з об’ємом від 120 GB
Доступ до глобальної Мережеві	Швидкість від 100 Мбіт/с

4.3 Опис розробленої програми

4.3.1 Загальні відомості

Для розробки «Консультанта абітурієнта» приймальної комісії Національного технічного університету «Дніпровська політехніка» потрібні знання як влаштовано Telegram Bot API [10] програмування, яке програмне забезпечення потрібно для реалізації комп'ютерної системи та які мови програмування можна використовувати.

Реалізація комп'ютерної системи консультанта абітурієнта буде проводитись за допомогою програмної платформи Node.js [11], це платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Якщо раніше Javascript застосовувався для обробки даних в браузері користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їхнього виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників. Для її використання потрібно завантажити програмне забезпечення з офіційного сайту Node.js та встановити на персональний комп'ютер.

Платформа на якій буде виконуватися написання коду повинна відповідати критеріям простоти у використанні, якості та надійності, також платформа має бути сучасною. Це потрібно для того щоб код при написані та реалізації був без прихованих збоїв та помилок які часто виникають при інтеграції коду до кінцевого продукту. Зважаючи на всі перераховані критерії підбору платформи реалізації коду JavaScript для цієї задачі буде використана платформа Atom.

Atom [18] — розроблений компанією «GitHub» вільний текстовий редактор і редактор коду, який може використовуватися як самодостатнє рішення, так і у ролі технологічного стека для побудови різних спеціалізованих рішень. Зокрема, на платформі Atom побудовані середовище розробки «Visual Studio Code» від компанії «Microsoft» і «Nuclide» від «Facebook».

Atom надає засоби крос-платформового редагування коду, включає вбудований пакетний менеджер і інтерфейс навігації файловою системою, надає засоби для одночасної спільної роботи з кодом, має інтелектуальну систему автодоповнення вводу, надає режими сумісності з Vim і Emacs, підтримує API для розробки розширень. Кілька файлів можуть бути відкриті в різних вкладках і одночасно відображені з використанням вертикального або горизонтального розбиття панелей. Інтерфейс може налаштовуватися через теми оформлення, підтримуються вкладки, закладки, розумний контекстний пошук коду, схлопування блоків коду, одночасне використання декількох курсорів і областей виділення, наочна позначка змін, автодоповнення та перевірка коду для різних мов (Ruby, Python, SQL, PHP, Perl, Objective-C, C/C++, JavaScript, Java, Go тощо).

Після довгого вибору між мовами програмування та повного аналізу їх можливостей було прийнято рішення використовувати мову програмування яка відповідає критеріям надійності, якості виконання та можливості інтеграції на поширенні платформи реалізації коду. Тому в даній ситуації для розробки консультанта абітурієнта буде використана поширена мова для веб-програмування JavaScript.

JavaScript [12] — динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне

керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Мова JavaScript використовується для:

- написання сценаріїв веб-сторінок для надання їм інтерактивності;
- створення односторінкових веб-застосунків (ReactJS, AngularJS, Vue.js);
- програмування на стороні сервера (Node.js);
- стаціонарних застосунків (Electron, NW.js);
- мобільних застосунків (React Native, Cordova);
- сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter);
- всередині PDF-документів тощо.

Незважаючи на схожість назв, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов отриманий «у спадок» від мови C, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme.

4.3.2 Функціональне призначення

Розроблюване програмне забезпечення комп'ютерної системи «Консультанта абітурієнта» приймальної комісії Національного технічного університету «Дніпровська політехніка» призначається для реалізації функцій виконання команд, які використовують користувачі та адміністратори консультанта абітурієнта.

Програма виконує такі дії як:

- Функція кнопки, призначена для виводу кнопки на екран користувача, натискання кнопок, натискання на функціональні блоки у консультанті абітурієнта;
- Функцій вводу-виводу повідомлення, ця функція безпосередньо взаємодіє з усіма функціями консультанта абітурієнта, вона призначена для використання чату з консультантом абітурієнта у різних цілях: вводу

команд користувача по відношенню до консультанта абітурієнта, виводу повідомлень консультанта на екран користувача, підтримання зворотного зв’язку з представниками приймальної комісії;

- Функції глобального пошуку, функція взаємодіє з всією базою даних для видачі шуканої інформації у консультанті абітурієнта, визивається за допомогою функції вводу-виводу повідомлення;
- Функції локального пошуку, функція виконується у трьох різних розділах консультанта абітурієнта і шукає інформацію тільки в тому розділі де була ініційована дія пошуку, визивається за допомогою функцій вводу-виводу повідомлення та кнопки;
- Функції запису інформації до бази даних, функція виконується при створенні нового функціонального блоку, або його редагуванні у трьох різних розділах консультанта абітурієнта, визивається за допомогою функції кнопки;
- Функції редагування інформації у базі даних, призначена для редагування вже створених функціональних блоків, визивається за допомогою функції кнопки;
- функції видалення інформації з бази даних, призначена для видалення існуючої інформації з бази даних, визивається за допомогою функції кнопки;
- Створення функціональних блоків, створює блок у трьох різних розділах для подальшого заповнення інформації у нього, визивається за допомогою функції кнопки, може бути видалений за допомогою функції видалення інформації з бази даних;
- Створення зворотного зв’язку, створює можливість спілкування користувача з представниками приймальної комісії, визивається за допомогою функції кнопки, підтримується за допомогою функції вводу-виводу повідомлення.

Усі вище перечисленні функції реалізовані у консультанті абітурієнта у повному обсязі. Були проведені тести роботи програми для виявлення помилок і їх доопрацювання.

Користувач «Консультанта абітурієнта» може використовувати тільки функції кнопки, та функції вводу-виводу повідомлення, глобального та локального пошуку, це обумовлено його статусом, користувач немає прав на зміну інформаційного наповнення консультанта, видалення функціональних блоків або їх створення.

Адміністратор або представник приймальної комісії має права на створення інформаційного наповнення його редагування та видалення, створення функціональних блоків, використання глобального та локального пошуку.

Натомість використання зворотного зв’язку доступне як для користувачів так і адміністраторів та представників приймальної комісії.

4.3.3 Опис логічної структури

Схема алгоритму [19] зображає послідовність символів, з'єднаних між собою стрілками, які вказують послідовність виконання і зв'язок між символами. Всередині символів записується їх короткий зміст.

Схема - це спосіб представлення алгоритму в графічній формі, у вигляді геометричних фігур, сполучених між собою лініями (стрілками). Форма блока визначає тип дії, а текст всередині блоку дає детальне пояснення конкретної дії. Стрілки на лініях, що сполучають блоки схеми, вказують послідовність виконання команд, передбачених алгоритмом. Схеми, за рахунок наочності спрощують створення ефективних алгоритмів, розуміння роботи вже створених, а як наслідок і їх оптимізацію.

Алгоритми підпрограми виконання дії у консультанті:

На рисунку 4.3 зображена схема алгоритму виконання дії у консультанті при натисканні кнопки. В першому блоці проводиться оголошення констант та запис даних у зміну – це потрібно для подальшої роботи з кнопками у

«Консультанті абітурієнта». У другому блоці описується умова оновлення даних консультанта, якщо дані потребують оновлення умова виконується у третьому блоці, якщо немає потреби в оновлені даних відбувається перехід до наступної умови. У четвертому блоці зображена умова виклика функції пошуку спеціальностей, якщо користувач у меню консультанта натискає кнопку «пошук спеціальностей» то відбувається перехід до наступного алгоритму дії пошуку, а в інших випадках викликається функція визначення наступного кроку.

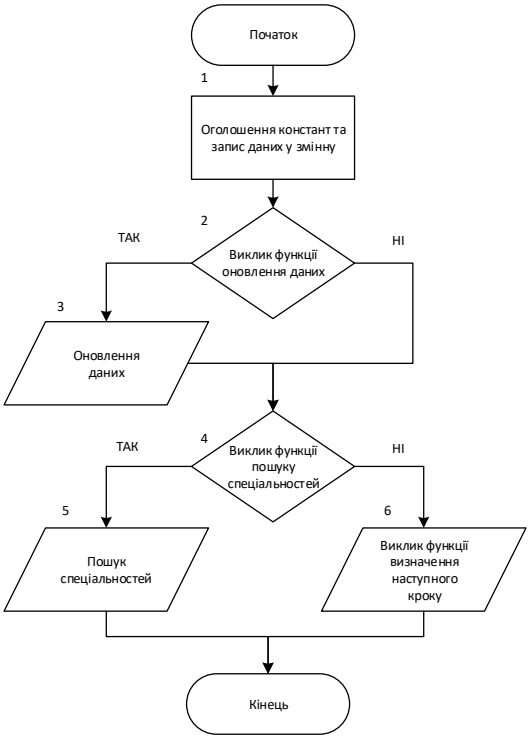


Рисунок 4.3 – Схема алгоритму виконання дії у консультанті при натисканні кнопки

Текст програми алгоритму виконання дії у консультанті при натисканні кнопки:

```
bot.on("callback_query", (query) => {
  // функція при натисканні кнопки
  const id = query.message.chat.id;
  const splitted = query.data.split("|");
  const search = splitted.length > 1 && splitted[0] == "search";
```

```
// Запис данних у змінну
if (splitted.length > 1 && splitted[5] == "1")
  return updateData(id, splitted, false);
// Виклик функції оновлення даних
if (query.data == "/search" || search)
  return searchFacultet(id, "", search ? splitted : false);
// Виклик функції пошуку факультету
processNextStep(query.data, id); }
// Виклик функції визначення наступного кроку
```

На рисунку 4.4 зображена схема алгоритму виконання функції оновлення даних. В першому блоці схеми алгоритму можна побачити умову виконання функції оновлення даних, якщо користувач написав текстове повідомлення до «Консультанта абітурієнта» то виконується процес оновлення даних, у випадку якщо тексту нема, буде виконуватися функція наступного кроку яка зображена у іншому алгоритмі. В третьому, четвертому та п’ятому блоках виконуються процеси оновлення бази даних консультанта, та можливої її зміни для своєчасної та правильної відповіді на запит користувача.

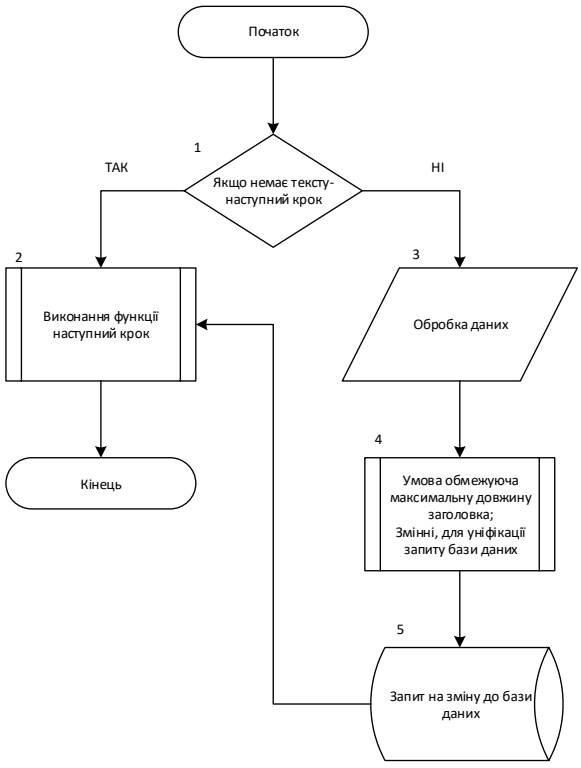


Рисунок 4.4 – Схема алгоритму виконання функції оновлення даних

Текст програми алгоритму виконання функції оновлення даних:

```
function updateData(id, _data, text) {
 // Функція оновлення даних
 if (!text)
  return processNextStep(
   `${_data[0]}|${_data[1]}|${_data[2]}|${
    _data[3] == "text" ? "2" : "text"
   }|${_data[4]}`,
   id
  );
 // Умова якщо немає тексту відобразити таступний крок
 const schema = {
  "0": "b_school",
  "1": "b_college",
  "2": "master",
 };
 // Відображення цифрових данних до меню
 let _item = _data.split("|");
 // Обробка даних
 if (_item[3] == "title" && text.length > 60) {
  bot.sendMessage(id, "Перевищено максимальну довжину заголовка!");
  editData(id, _item);
  return;
 }
 // Умова обмежуюча максимальну довжину заголовка
 let data = [text, _item[4]];
 let sql = `UPDATE ${schema[_item[2]]}
          SET ${_item[3]} = ?
          WHERE id = ?`;
 // Змінні, для уніфікації запиту бази даних
```

```
    db.run(sql, data, function (err) {
      if (err) return console.error(err.message);
      console.log(`updateData(f) Row(s) updated: ${this.changes}`);
    });
    // запит на зміну до бази даних
    processNextStep(
      `${_item[0]}|${_item[1]}|${_item[2]}|${_item[3] == "text" ? 2 : "text"}|${
        _item[4]
      }`,
      id
    );
    // Виявлення наступного кроку
  }
```

На рисунку 4.5 зображена схема алгоритму виконання функції пошуку спеціальностей. В першому блоці схеми алгоритму зображено процес виводу про розпочатий пошук, для того щоб розпочати пошук користувачеві потрібно або натиснути кнопку пошуку у місцях де ця можливість реалізована, або написати текст повідомлення для «Консультанта абітурієнта» з ключовими словами для пошуку. У другому блоці відбувається вивід даних навігації спеціальностей в залежності від того де саме був розпочатий пошук користувачем. У третьому блоці відбувається пошук у базі даних спеціальностей для подальшого виводу результату. У блоці чотири виводиться результат пошуку, з оглядом на те, у якому функціональному блоці, або глобально було оформлено запит на пошук, виводиться результат. Тобто якщо функцію пошуку було викликано у певному функціональному блоці то пошук буде реалізовано саме серед цього блоку, якщо пошук було викликано глобально буде виведено результати з усіх функціональних блоків, а самі результати будуть відсортовані.

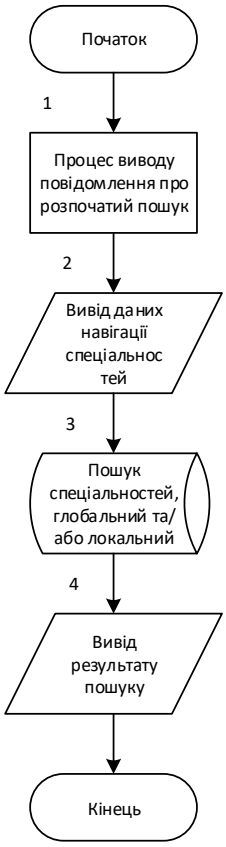


Рисунок 4.5 – Схема алгоритму виконання функції пошуку спеціальностей

Текст програми алгоритму виконання функції пошуку спеціальностей:

```
async function searchFacultet(id, text, status) {
 // Функція пошуку Факультету
 if (!status || (typeof status == "object" && status[2] == "skip")) {
  const record_data =
   typeof status == "object" ? `search|${status[1]}` : `search`;
  recordAction(id, record_data);
  bot.sendMessage(id, "Введіть номер спеціальності чи просто назву");
  // Умова виводу повідомлення про початий пошук та запис дії пошуку
  return;
 }
 const schema = {
  b_school: "∘ Вступ Бакалавра на базі ПЗСО ∘",
  b_college: "∘ Вступ Бакалавр на базі ОКР ∘",
```

```
  master: "∘ Вступ Магістра на базі Бакалавра ∘",
 };
 // Схема меню
 const tableSchema = {
  b_school: "0",
  b_college: "1",
  master: "2",
 };
 // Відношення схеми меню до цифрового значення
 let _status = typeof status == "object" && status[1];
 // Змінна поточного статусу пошуку
 bot.sendMessage(
  id,
  !_status
   ? "Назад до головного меню"
   : `Назад до спеіальностей ${schema[status[1]]}`,
  {
   reply_markup: {
    inline_keyboard: [
     [
      {
       text: !_status ? "Меню 🏠" : "🌧 Назад 🌧",
       callback_data: !_status
        ? "hello"
        : `facultets|${tableSchema[status[1]]}`,
},],],}, } );
```

// Вивід даних навігації спеціальностей у відношенні від глобального пошуку та локального

//Різниця між глобалним та локальним пошуком

```
setTimeout(() => {
```

```
// Затримка на 500 мл. сек.
for (let item of !_status
 ? ["b_school", "b_college", "master"]
 : [status[1]]) {
 db.all(`SELECT * FROM ${item}`, [], (err, rows) => {
  let output = {
   text: `▾ ${schema[item]} ▾`,
   data: [],
  };
  for (let el of rows) {
   if (el.title.includes(text))
    output.data.push([
     {
      text: el.title,
      callback_data: `facultets_info|${el.id}|${tableSchema[item]}`,
  }, ]); }
   if (output.data.length) {
    bot.sendMessage(id, output.text, {
     reply_markup: {
      inline_keyboard: output.data,
}, });} }); }
 // Пошук спеціальності, глобальний і локальний
}, 500); }
```

На рисунку 4.6 зображена схема алгоритму виконання функції наступного кроку, головна виконуюча функція у консультанті абітурієнта. В першому блоці оголошується змінна для подальшої роботи з функцією наступного кроку. У другому блоці виконується умова присутності даних, якщо їх немає виконується дія виводу повідомлення «Помилка», яка зображена у третьому блоці. У блоці чотири відбувається виклик основних функцій консультанта таких як:

– функції обробки даних;
– функції зворотного зв'язку;
– функції створення даних;
– функції відображення даних;
– функції відображення спеціальностей та основної інформації;
– функції видалення користувача;
– функції видалення даних;
– функції керування зворотним зв'язком.

Після виклику всіх функцій відбувається обробка навігації головного меню та його відображення яке зображено у блоках п’ять та шість.

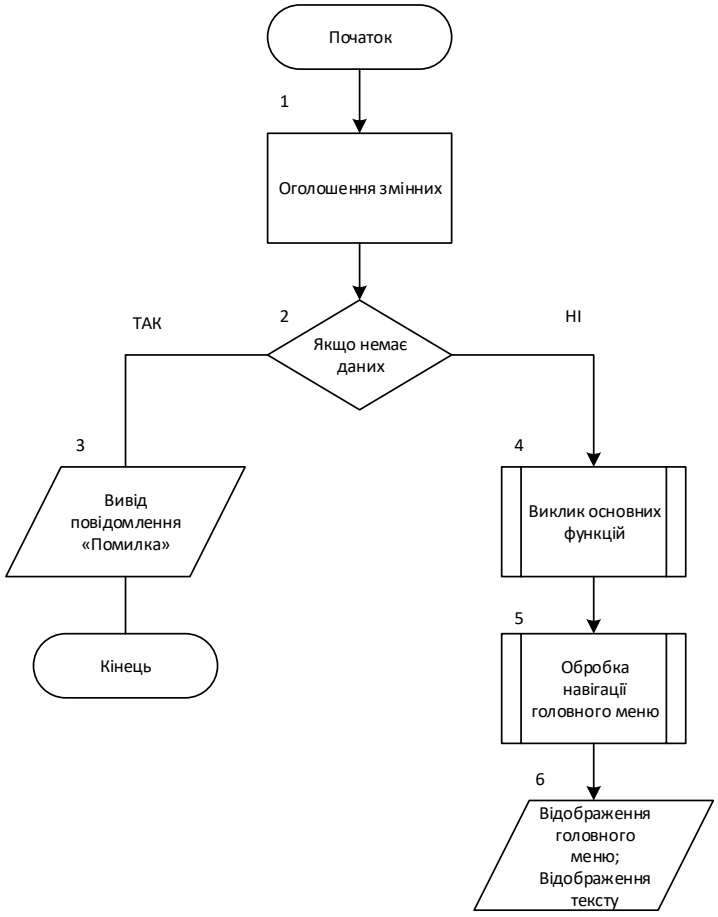


Рисунок 4.6– Схема алгоритму виконання функції наступного кроку

Текст програми алгоритму виконання функції наступного кроку:

```
function processNextStep(data, id) {
 // Функція виявлення наступного кроку
```

```
let _data = [];
let splitted = [];
if (data.includes("cancel")) return deleteAction(id, data);
// Оголошення змінних
if (!data) return bot.sendMessage(id, "✕ Помилка ✕");
// Умова відображає помилку
if (data.includes("|")) {
  // Умова виявляе наявність дії
  splitted = data.split("|");
  // Змінна розділення даних
  if (splitted[0] == "edit") {
    return editData(id, splitted); }
  if (splitted[0] == "photo_price") return addPhotoPrice(id, data, false);
  if (splitted[0] == "photo") return addPhoto(id, data, false);
  // Виклик Функції обробки данних
  if (splitted[0] == "user") return writeToUser(id, data);
  // Виклик Функції зворотного зв'язку
  if (splitted[0] == "create") return createData(id, splitted);
  // Виклик Функції створення даних
  if (splitted[0] == "facultets_info") return showData(id, splitted);
  // Виклик Функції відображення даних
  if (splitted[0] == "facultets" || splitted[0] == "main_info")
    return processDataShow(id, splitted);
  // Виклик Функції відображення спеціальностей та основної інформації
  if (splitted[0] == "back" && splitted[1] == "0") _data = stage_1;
  // Запис данних у змінну для повернення у головне меню
  if (splitted[0] == "delete" && splitted[1] == "user")
    return deleteUser(id, data, 0);
  // Виклик Функції видалення користувача
  if (splitted[0] == "delete") return handleDelete(id, splitted, "message");
```

```
    // Виклик Функції видалення данних
    if (splitted[0] == "chat") return processChat(id, data, false);
    // Виклик Функції керування зворотним зв'язком
  } else {
    switch (data) {
// Обробка навігації головного меню
      case "0":
        _data = stage_2;
        break;
      case "1":
        _data = stage_3;
        break;
      case "2":
        _data = stage_4;
        break;
      case "hello":
        recordAction(id, null);
              _data = stage_1;
        break;
      default:
  }}
  if (_data.data) {
    bot.sendMessage(id, _data.text, {
      reply_markup: {
        inline_keyboard: _data.data, },});
          return; }
    // Відображення головного меню
  bot.sendMessage(id, _data.text); }
  // Відображення тексту
```

Алгоритми підпрограми виконання функції при відправлені повідомлення:

На рисунку 4.7 зображена схема алгоритму виконання функції при відправлені повідомлення, одна з двох головних функцій консультанта абітурієнта. В першому блоці можна побачити, що функцію буде визвано тільки при відправці повідомлення до консультанта абітурієнта, в другому блоці можна побачити процес запису даних унікального ключа користувача, це потрібно для ведення обліку користувачів та подальшої можливої модерації у консультанті абітурієнта. Третій блок умови допомагає зрозуміти які функції до яких користувачів відносяться, на приклад якщо ви є адміністратор консультанта абітурієнта то ви маєте доступ до всіх функцій які зображені в блоках від четвертого до тринадцятого, а якщо ви звичайний користувач вам будуть доступні функції які зображені у блоках чотири та сім і в блоках одинадцять та тринадцять. У блоках від четвертого до сьомого описується робота текстового пошуку, спочатку функція викликається звичайним текстовим повідомленням у консультанті абітурієнта, згодом виконується умова після якої текстове повідомлення обробляється у базі даних і звіряється з можливими результатами, останній крок, це вивід шуканої інформації на екран. У блоках з восьмого по десятий описуються функція доступ до яких має тільки адміністратор консультанта абітурієнта. Самі функції використовуються для пошуку користувачів по унікальному ключу, видалення даних про користувача, видалення даних та оновлення даних. У блоках від одинадцятого до тринадцятого описується функція зворотного зв’язку з представниками приймальної комісії. Функцією зворотного зв’язку можуть користуватись як адміністратори так і користувачі, але тільки звичайні користувачі можуть визивати цю функцію, інші категорії користувачів мають тільки можливість підтвердження або відміни запиту на зворотній зв'язок. Закінчити розмову можна у будь який момент, весь текст повідомлень записується і не видаляється.

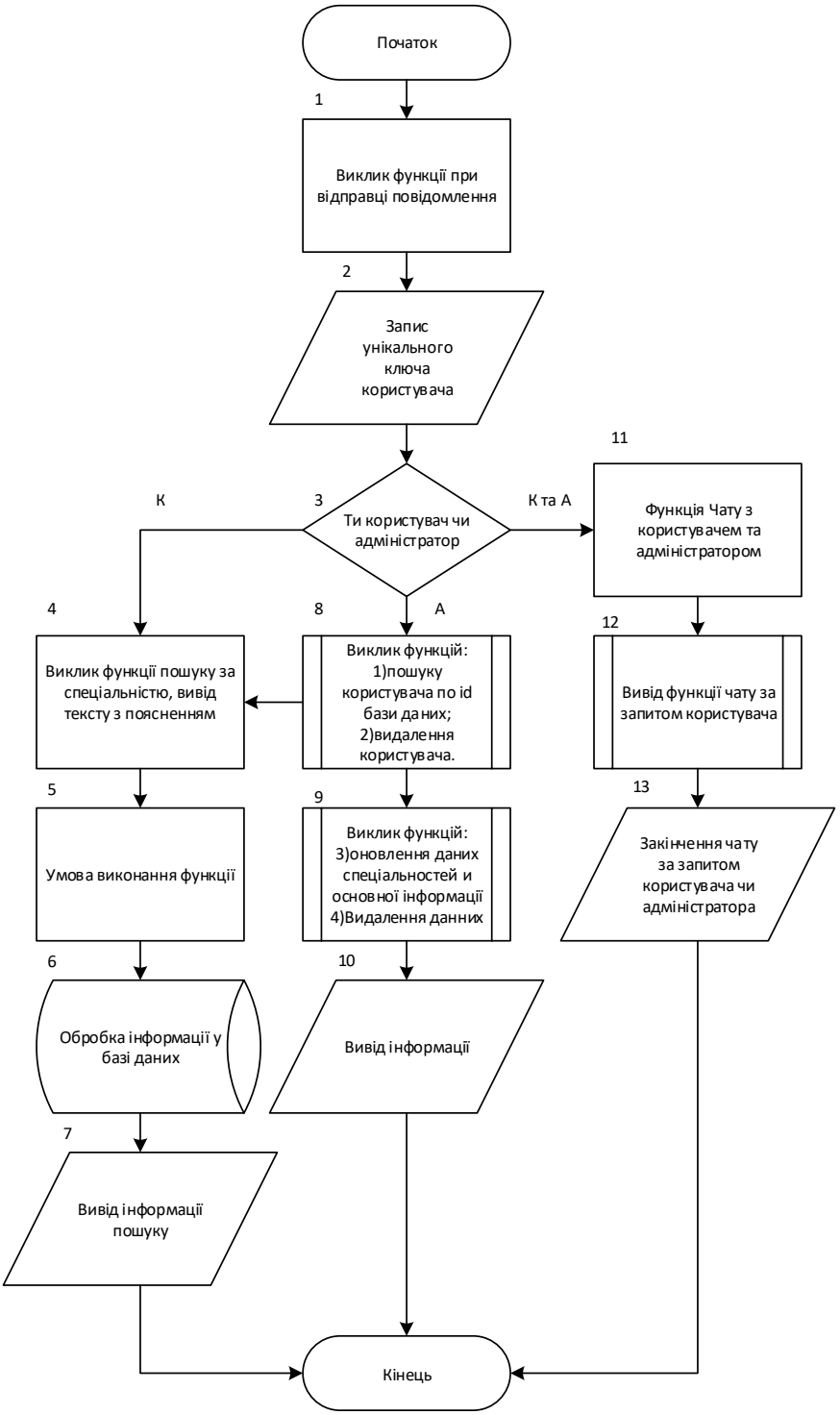


Рисунок 4.7 – Схема алгоритму виконання функції при відправлені повідомлення

Текст програми алгоритму виконання функції при відправлені повідомлення:

```
bot.on("message", async (res) => {
  // Виклик функції при відправці повідомлення
```

```
const id = res.chat.id;
// запис унікального ключа користувача
if (res.text == "/users") return showListOfPeople(id, "");
// Виклик функції відображення користувачів
if (res.text == "/search") return searchFacultet(id, res.text, false);
if (res.text == "/start" || res.text == "/menu")
  return processNextStep("hello", id);
// Виклик функції пошуку за спеціальністю, вивід тексту з поясненням
searchForUser(id, function (rows) {
  if (rows.length && rows[0].action !== null) {
    // Умова виконання текстових команд користувача
    const action = rows[0].action;
    const splitted = action.split("|");
    if (splitted[0] == "photo") return addPhoto(id, action, res);
    if (splitted[0] == "photo_price") return addPhotoPrice(id, action, res);
    // Обробка данних
    if (splitted.length > 1 && splitted[1] == "user")
      return deleteUser(id, rows[0].action, !!(res.text == "+"));
    // Виклик функції видалення користувача
    if (
      action == "search" ||
      (splitted.length > 1 && splitted[0] == "search"))
          return searchFacultet(
        id,
        res.text,
        splitted.length > 1 ? splitted : true);
// Виклик функції пошук за спеціальністю, пошук та вивід шуканої інформації
    if (splitted[0] == "chat") return processChat(id, action, res.text);
    // Функція чату з користувачем
```

```
            if (action !== null && action.split("|")[0] == "delete")
              return handleDelete(id, action, res.text == "+");
            // Видалення данних
            if (action !== null) return updateData(id, action, res.text);
            // Функція оновлення даних спеціальностей и основної інформації
            return;
          } else {
            if (!rows.length) {
              // Умова відсутності користувача бази даних
              db.run(
                `INSERT INTO users(first_name, last_name, chat_id, language) VALUES(?, ?, ?, ?)`,
                [ [res.chat.first_name],
                [res.chat.last_name],
                [id],
                [res.chat.language_code], ],
                function (err) {
                if (err) return console.log(err.message);
                processNextStep("hello", id);
              });
              // Додавання користувача у базу даних
              return; }
              if (res.text == "/start" || res.text == "/menu")
              return processNextStep("hello", id);
            // Вивід меню
            searchFacultet(id, res.text, true);
            // Пошук спеціальностей по тексту користувача
            // Виконається якщо не має інших поточних задач
        }});});
```

На рисунку 4.8 зображена схема алгоритму виконання функції зворотного зв’язку. У першому блоці виконується обробка даних для подальшої роботи системи зворотного зв’язку. У другому блоці виконується процес початку чату з адміністратором чи користувачем. У третьому блоці виконується умова відхилення чату, якщо чат відхиллено виводиться повідомлення про його відмін, що зображено у четвертому блоці. При прийомі чату з двох боків чат вважається доступний, листування почалося, для завершення чату необхідно або натиснути кнопку для завершення, або виконати іншу доступну дію.

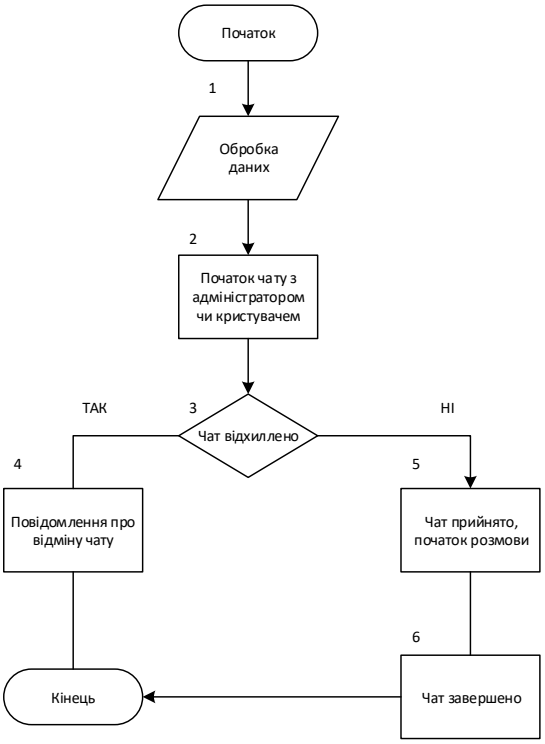


Рисунок 4.8 – Схема алгоритму виконання функції зворотного зв’язку

Текст програми алгоритму виконання функції зворотного зв’язку:

```
function processChat(id, data, message) {
  // Керування зворотним зв'язком
  let user = data.split("|");
  // Обробка данних
  if (message) {
    bot.sendMessage(
      +user[1],
```

```
            `${!user[2] ? "Адміністратор" : user[2]}: ${message}`
          ); return; }
        // Відправлення повідомлення користувачу на отримання повідомлення від адміністратора
          if (id == +user[1]) {
            bot.sendMessage(id, "Ви не можете почати чат з собою");
            recordAction(id, null);
            // Видалення поточної дії
            showListOfPeople(id, "");
            return;
          }
            // Умова обмежуюча початок чату з самим собою
           if (user[2] == "declined") {
            bot.sendMessage(id, "Чат відхилено.");
            bot.sendMessage(+user[1], "Чат відхилено користувачем.");
            recordAction(id, null);
            // Видалення поточної дії
            return; }
            // Умова відміни чату
           if (user[2] == "accept")
          {
            bot.sendMessage(id, "Чат прийнятий. Просто пишіть сюди свои питання.");
            bot.sendMessage(+user[1], "Чат прийнятий користувачем.");
            recordAction(id, `chat|${+user[1]}|${user[3]}`);
            // Запис дії для адміністратора
            recordAction(+user[1], `chat|${id}`);
            // Запис дії для користувача
            return; }
            // Умова підтвердження зворотного зв'язку
```

```
 if (!message)
{
   bot.sendMessage(id, "Запрос відправлено");
   bot.sendMessage(
     +user[1],
     "Адміністатору є що Вам сказати. Нажміть кнопку для початку чата",
     { reply_markup:
{
        inline_keyboard: [[{
            text: "Почати чат",
            callback_data: `chat|${id}|accept|${user[2]}`, },
            {text: "Відхилити чат",
            callback_data: `chat|${id}|declined`,
         },],],},} ); }
 // Умова відправлення запиту на зворотний зв'язок
 recordAction(id, data); }
 // Запис дії для роботи з зворотним зв'язком
```

4.3.4 Використовувані технічні засоби

Для того щоб розроблюваний консультант абітурієнта працював безпосередньо на платформі мессенджера Telegram [5] необхідно передбачити можливість установки його на комп’ютерний сервер приймальної комісії.

Подальша розробка комп'ютерної системи консультанта абітурієнта включає в себе реалізацію рішень по максимально ефективному впровадженню його в потужність сервера, який встановлений безпосередньо головному кабінеті приймальної комісії НТУ «ДП». Інтеграція онлайн консультанта не повинна обумовлюватися ніякими труднощами як з боку самого розроблюваного програмного коду так і з боку безпосередньо інтеграції коду програми на сервер. Так само при розробці програми буде враховуватися

надійність стороннього зв'язку з серверами, які будуть служити базою даних для одного з пунктів розроблюваного консультанта абітурієнта.

Окрім тих чинників що були описані вище так само в Telegram Bot API [10] програмі передбачено взаємодію користувача з чат ресурсом зворотного зв’язку.

На рисунку 4.9 зображено діаграму взаємодії комп’ютера з сервером. На ньому чітко зображено як користувач буде взаємодіяти з основною інформацією яка представлена в програмі на сервері.

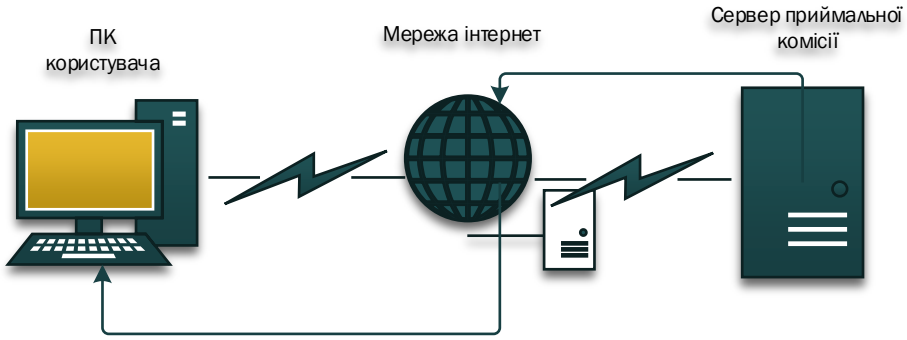


Рисунок 4.9 – Діаграма взаємодії комп’ютера з сервером приймальної комісії

На рисунку 4.10 зображено діаграму взаємодії смартфона користувача з сервером приймальної комісії. Персональний комп’ютер користувача через мережу інтернет взаємодіє з сервером приймальної комісії для взаємодії з «Консультантом абітурієнта».

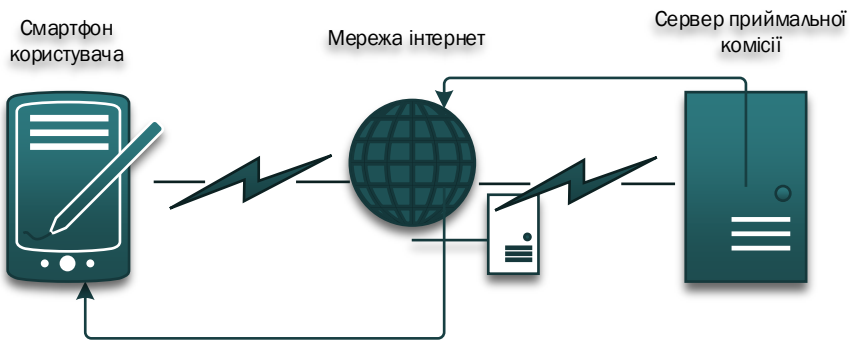


Рисунок 4.10 – Діаграма взаємодії смартфона з сервером приймальної комісії

На рисунку 4.11 зображено діаграму взаємодії персонального комп’ютера користувача та/або смартфона з чат ресурсом зворотного зв’язку. Користувач через мережу інтернет взаємодіє з сервером приймальної комісії, який в свою чергу формує та відправляє запит до адміністратора консультанта абітурієнта для подальшої консультації

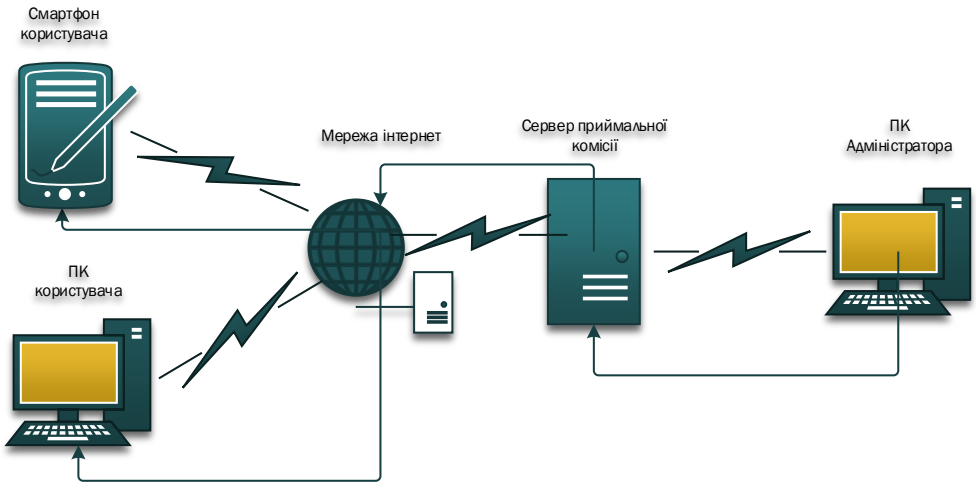


Рисунок 4.11 – Діаграма взаємодії комп’ютера з почтовим ресурсом

4.4 Очікувані техніко-економічні показники

Вибір саме такої реалізації програмного забезпечення обумовлений, відносно меншими економічними витратами в розробці і підтримці комп'ютерної системи. У порівнянні з іншими методами реалізації подібних програмних забезпечень комп’ютерних систем, які мають у своїй основі регулярні невизначені економічні вкладення в сервіси по створенню подібних систем і програмного забезпечення, а так само їх довгострокову підтримку, розроблена комп’ютерна система має значну економічну перевагу, яка полягає в одноразовій витраті певного бюджету на створення комп'ютерної системи, програмного забезпечення та підтримки системи.

З технічної точки зору розроблена комп'ютерна система і програмне забезпечення до неї має необмежений термін актуальності і експлуатації, так як система розроблялася на надійному обладнанні, при використанні сучасної мови програмування, що розвивається, при головних вимога багатофункціональності та адаптивності до будь-яких можливих зміни.

4.5 Висновки по розділу

«Консультант абітурієнта» приймальної комісії був розроблений на сучасній платформі реалізації програмного коду використовуючи поширену мову програмування. При розробці враховувалися всі сучасні тенденції Telegram Bot API програмування. Спираючись на дані факти можна стверджувати, що розроблене програмне забезпечення в своїй мірі досить сучасне і актуальне на сьогоднішній час. Корисність програмного забезпечення оцінюється його функціональністю і гнучкістю в налаштуванні, а так само безпосередньо у застосуванні.

5 ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ

5.1 Формулювання завдання та обґрунтування методики

Мета експерименту: виявлення ефективності обслуговування користувачів комп’ютерної системи консультанта абітурієнта на основі розробленої моделі СМО. Дослідити ефективність роботи впроваджуваної комп’ютерної системи. Скласти вивід на основі отриманих даних ефективності впроваджуваної комп’ютерної системи.

Задача експерименту: Методом безпосередньої оцінки [21] виявити найкращі значення модельованої ситуації обслуговування користувачів консультанта абітурієнта у часі та виявити ефективність впроваджуваної комп’ютерної системи «Консультант абітурієнта».

Метод безпосередньої оцінки [21] надає значення вимірюваної величини безпосередньо за відліковим пристроєм вимірювального приладу прямої дії. Наприклад, вимірювання тиску пружинного манометра, маси на циферблатних вагах, сили електричного струму амперметром і т.п. Точність вимірювання за допомогою цього методу стає обмеженою, але швидкість процедури вимірювання робить його незамінним для практичного застосування. Найбільш численною групою засобів вимірювання, що застосовуються для вимірювання цим методом є в тому числі і стрілочні прилади. Вимірювання за допомогою інтегруючого вимірювального приладу-лічильника також є методом безпосередньої оцінки.

Методика проведення експерименту: Експеремент проводиться за допомогою створеної у розділі 2 моделі системи масового обслуговування, яка розроблена для комп'ютерної системи "консультант абітурієнта", що повинна розраховувати в процентному еквіваленті кількість користувачів які залишилися в системі обслуговування за певний проміжок часу, тобто являються ще не обслуженими з певних причин.. На рисунку 5.1 зображена структурна схема системи масового обслуговування комп’ютерної системи «Консультанта абітурієнта».

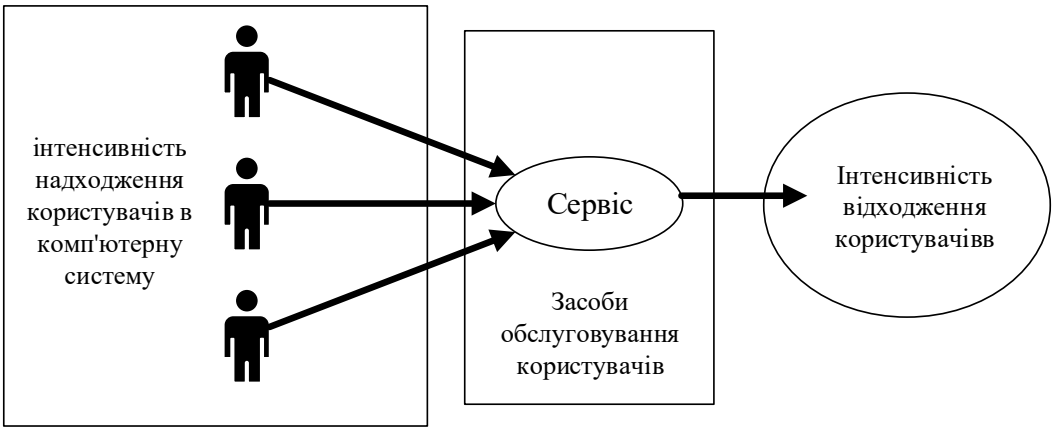


Рисунок 5.1 – Структурна схема системи масового обслуговування

Запуск моделі та проведення експерименту здійснюється у середовищі MATLAB [14] за допомогою інтерактивного інструменту для моделювання, імітації та аналізу динамічних систем – Simulink [15]. Розроблене компанією The MathWorks. Дає можливість будувати графічні блок-діаграми, імітувати динамічні системи, досліджувати працездатність систем і вдосконалювати проекти. Simulink повністю інтегрований з MATLAB, що забезпечує швидкий доступ до широкого спектра інструментів аналізу і проектування. На рисунку 5.2 зображено модель СМО комп’ютерної системи.

Опис моделі:

блок n – константа, визначається у ручному режимі, потрібна для визначення кількості користувачів в комп’ютерній системі;

блок Subsystem – підсистема моделі СМО, зображена на рисунку 2.4;

блок Display – виведення результату.

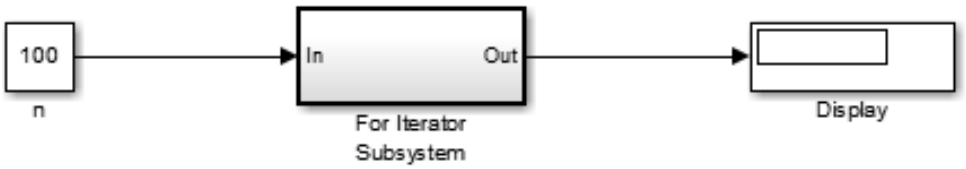


Рисунок 5.2 – Модель СМО комп’ютерної системи

На рисунку 5.3 зображено підсистему моделі СМО комп’ютерної системи

Опис підсистеми моделі:

блок In – вхід підсистеми;

блок Data Store Memory (A,B) – зберігає всі дані під час моделювання системи;

блок Data Store Write (A,B) – записує всі дані під час моделювання системи;

Блок Data Store Read (A,B) – зчитує всі дані під час моделювання системи;

блок Uniform Random Number – генерація випадкового сигналу в діапазоні від 0 до 100;

блок Data Type Conversion (int8) – використовується для перетворення нецілих чисел в дійсні числа;

блок Product – використовується для множення або ділення чисел, які приходять на вхід блоку;

блок Constant - застосовується для використання статичного числа;

блок Scope – Осцилограф, який застосовується для виведення графіку на екран;

блок Out – вихід підсистеми.

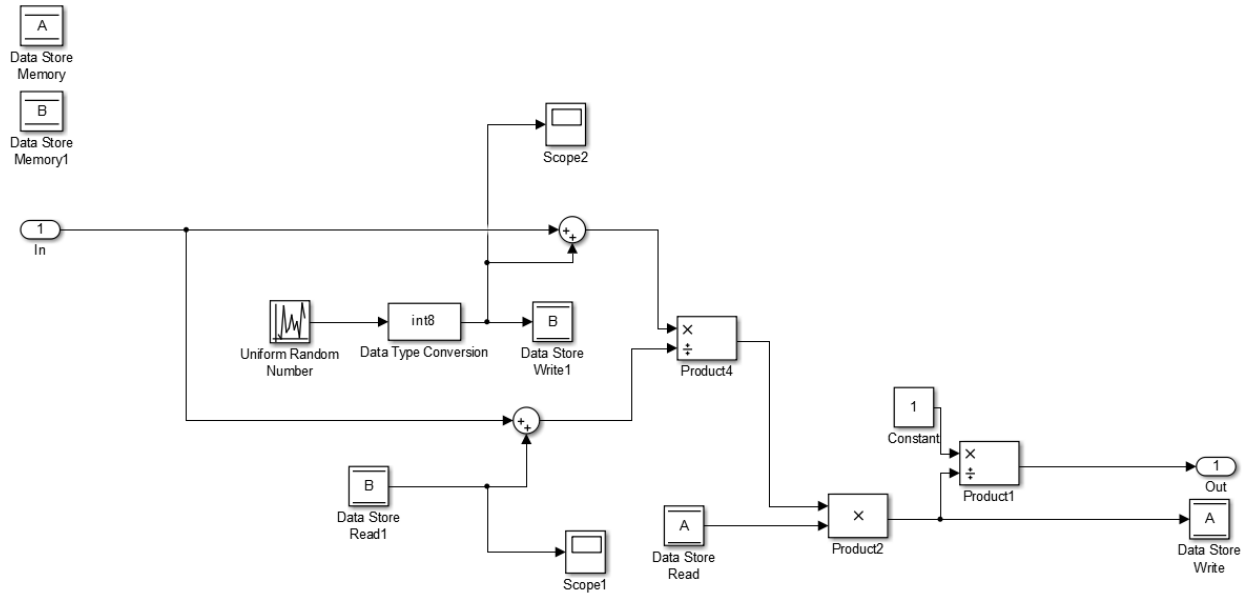


Рисунок 5.3 Підсистема моделі СМО комп’ютерної системи

5.2 Вимоги до експерименту

Для подальшого проведення експерименту слід ввести наступні вимоги:

- Використання створеної моделі системи масового обслуговування комп’ютерної системи «Консультанта абітурієнта»;
- Використовувати персональний комп’ютер з наступними мінімальними характеристиками які зображено в таблиці 5.1, для запуску моделі СМО;
- Вести запис даних моделювання СМО у таблицю;
- Провести розрахунок ефективності обслуговування користувачів комп’ютерної системи «Консультанта абітурієнта».

Таблиця 5.1 – Мінімальні характеристики персонального комп’ютера

Тип	Найменування
Процесор	Intel Pentium N4200
Оперативна пам’ять	4 ГБ
Жорсткий диск	500 ГБ
Відео адаптер	Intel HD Graphics 505
Операційна система	Windows 7/8/10

5.3 Результати експерименту

5.3.1 Сутність експерименту

Експеримент проводиться методом безпосередньої оцінки, для виявлення найліпшого показника ефективності моделі СМО. У ході експерименту буде проведено 10 моделювань системи масового обслуговування з поступальним значенням кількості користувачів у системі від 10 до 100. Експеримент проводиться за певний проміжок часу, 3 хвилини. Результатом одного моделювання являється процентне відношення користувачів які находяться в системі на протязі часу, серед всіх користувачів які надійшли до системи масового обслуговування. Чим менший процентний показник тим ліпше працює система. Після проведення моделювань усі параметри будуть занесені до таблиці, для подальшого розрахунку оптимального варіанту кількості

користувачів, що обслуговуються комп’ютерною системою «Консультант абітурієнта» у певний проміжок часу.

Передбачуваним результатом експерименту являється вирахуваний найменший показник решти користувачів у системі по відношенню до користувачів яких вже обслужили, цей показник означає, що система працює стабільно, без перебою та обслуговує більше користувачів за певний проміжок часу ніж залишається у системі.

При проведені експерименту очікується отримати оптимальний варіант показника обслуговування користувачів для комп'ютерної системи «Консультанта абітурієнта».

5.3.2 Результат експерименту у фактах

Результат проведення експерименту зображено на графіку 5.1 та у таблиці 5.2. При проведені експерименту, були виконані певні розрахунки для отримання більш детального результату експерименту.

За формулою 5.1 було виконано розрахунки кількості обслужених користувачів, розрахунки виконувалися для визначення подальшого параметру ефективності комп’ютерної системи.

$$\mu_n = \mathrm{n} - \mathrm{P}_n \tag{5.1}$$

За формулою 5.2 було виконано розрахунки кількості користувачів які залишилися у системі, розрахунки виконувався для визначення подальшого параметру ефективності комп’ютерної системи.

$$\lambda_n = \mathrm{n} - \mu_n \tag{5.2}$$

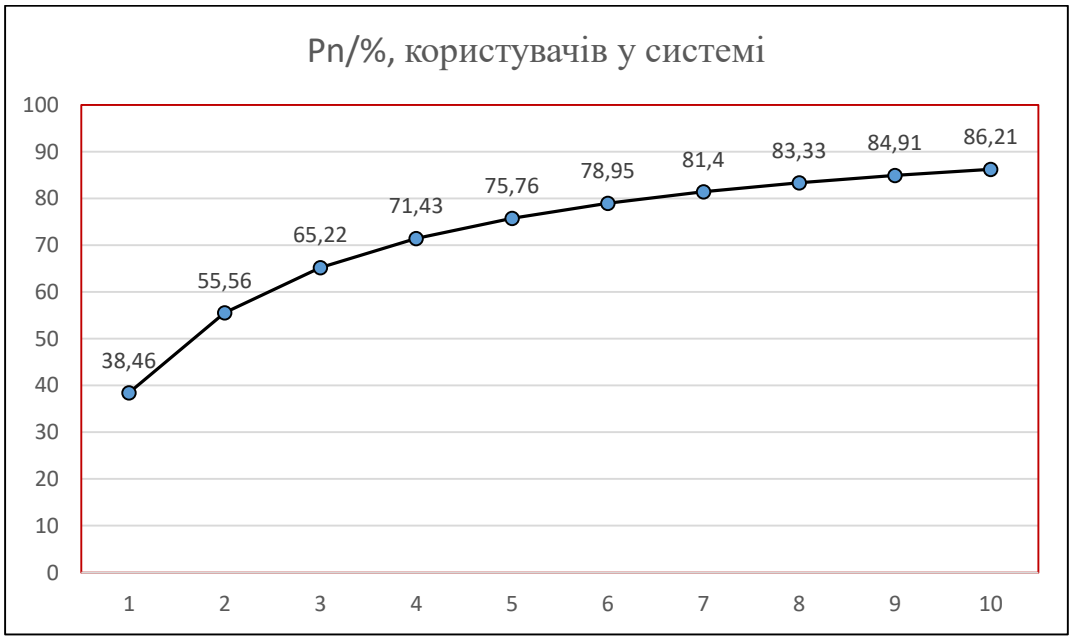
Фактичним результатом експерименту являється розрахунок Δ_n, за формулою 5.3, що являється дією ділення користувачів які залишились у системі на обслужених користувачів. Якщо результат ділення менше або дорівнює 1 цей результат являється ефективним показником обслуговування користувачів у комп’ютерній системи, в інших випадках результат являється неефективним. Результат розрахунку Δ_n, представлено у таблиці 5.2 та на графіку 5.2.

$$\Delta_n = \frac{\lambda_n}{\mu_n} \tag{5.3}$$

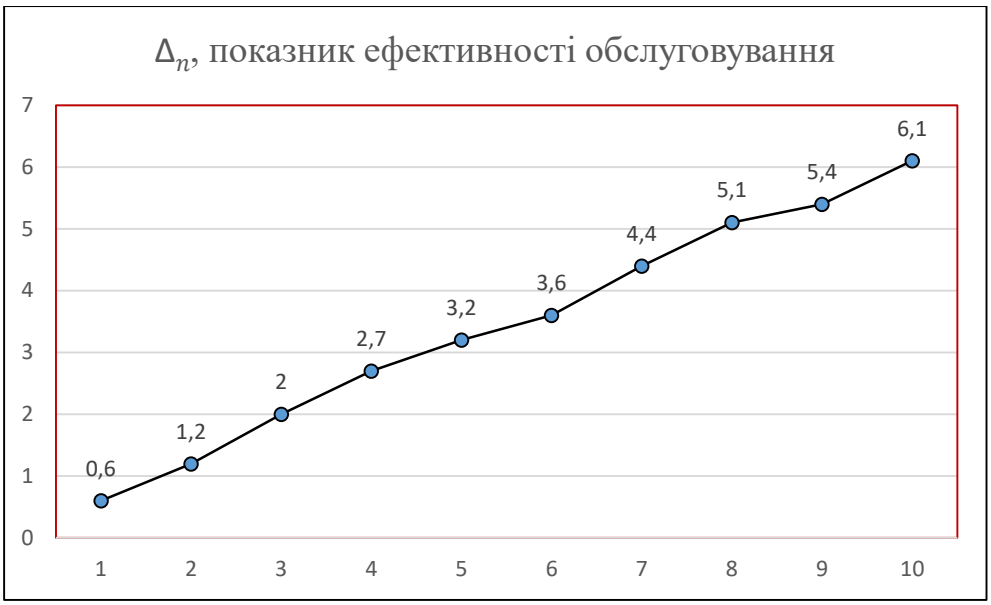
Всі дані що заносилися у таблицю 5.2 округляються у більшу сторону так, як вони мали цілі та десяткові значення при виконанні розрахунків.

Приблизна похибка у вимірюваннях допускається у діапазоні від 0.1% до 2.5%, це обумовлено відхиленнями у розрахунках в моделі системи масового обслуговування та при наявності вірогідності різних умов поведінки користувачів комп’ютерної системи.

У таблиці 5.2 наведені результати експерименту та результати розрахунку. З огляду на поставлену задачу експерименту, методом безпосередньої оцінки та за допомогою використання зазначених критеріїв успішної роботи комп’ютерної системи, було вирахувано оптимальний показник кількості користувачів, що обслуговуються комп’ютерною системою протягом трьох хвилин. Комп’ютерна система консультанта абітурієнта буде ефективно працювати якщо 10 або менше користувачів одночасно будуть нею обслуговуватися



Графік 5.1 – Результат моделювання експерименту



Графік 5.1 – Результат розрахунку експерименту

Таблиця 5.2 – Результат проведення експерименту

№/М	n, кількість користувачів	P_n/%, користувачі в у системі	μ_n, кількість обслужених користувачів	λ_n, кількість користувачів які залишились у системі	Δ_n
1	10	38,46	6	4	0,6
2	20	55,56	9	11	1,2
3	30	65,22	10	20	2
4	40	71,43	11	30	2,7
5	50	75,76	12	38	3,2
6	60	78,95	13	47	3,6
7	70	81,4	13	57	4,4
8	80	83,33	13	67	5,1
9	90	84,91	14	76	5,4
10	100	86,21	14	86	6,1

5.3.3 Аналіз відповідності досліджень

При проведенні теоретичних досліджень, параметри налаштування і функціонування створеної моделі системи масового обслуговування комп’ютерної системи «Консультант абітурієнта» були налаштовані для проведення подальшого експерименту. Проведення експерименту довело, що модель системи масового обслуговування була створена і налаштована коректна і його виконанню нічого не заважало.

Експеримент пройшов успішно, було виявлено оптимальний варіант кількості клієнтів які комп'ютерна система «Консультанта абітурієнта» може обслужити за 3 хвилини. Даний параметр в значній мірі перевершує звичайну консультативну діяльність прийомної комісії в декілька разів, так як для обслуговування абітурієнта очним методом взаємодії з ним, буде потрібно від 10 до 15 хвилин часу, коли розроблена система при різних умовах зможе обслужити від 6 до 14 абітурієнтів за 3 хвилини.

5.3.4 Характеристика новизни результатів

Наукове значення експерименту, що провівся з використанням розробленої моделі системи масового обслуговування комп'ютерної системи «Консультанта абітурієнта» полягає у обґрунтуванні доцільності математичної моделі та її використанні для проведення експерименту.

Практичне значення результатів експерименту полягає у значущості досліджень для практики та можливі шляхи використання результатів у майбутніх наукових працях.

5.4 Висновки по розділу

При початку проведення експерименту було встановлено його мету, завдання та обрано методику проведення експерименту. Експеримент проводився за заздалегідь сформульованими та визначеними вимогами, які відповідають суті експерименту. Проведений експеримент був вдалим, а дії які виконувалися у впродовж експерименту були докладно та чітко описаними з кінцевим результатом. Результатом експерименту вважається виведена методом безпосередньої оцінки за допомогою моделі системи масового обслуговування та подальшими розрахунками величина з урахуванням часу, яка характеризує ефективну роботу впроваджуваної комп'ютерної системи «Консультант абітурієнта»

ВИСНОВКИ

Кваліфікаційна робота є завершеною науковою роботою, в якій вирішена науково-практична задача створення комп’ютерної системи «Консультант абітурієнта» приймальної комісії Національного технічного університету «Дніпровська політехніка», шляхом застосування розробленої моделі та методів реалізації системи підтримки функціонування, вибору та обґрунтування обладнання для розробки і підтримки, розробки програмного забезпечення. Основні висновки і результати роботи полягають у наступному:

1. Розроблено модель системи масового обслуговування комп’ютерної системи «Консультант абітурієнта». Описується і обґрунтовується доцільність наукового завдання по створенню моделі майбутньої комп’ютерної системи. Проводиться вибір системи за допомогою якої можна буде провести моделювання створюваної комп’ютерної системи, а саме створення моделі системи масового обслуговування для комп’ютерної системи. Було описано і обґрунтовано вибір моделі системи, а кінцевим етапом являється створення моделі системи масового обслуговування для комп’ютерної системи «Консультант абітурієнта».

2. При створенні системи контролю в розробленій комп'ютерній системі "Консультант абітурієнта" були враховані всі поставлені вимоги до реалізації системи контролю. Створення системи контролю включало в себе вибір оптимальних, сучасних існуючих рішень апаратних частини системи контролю для розробки програмного забезпечення комп'ютерної системи і її підтримки в майбутньому. Створення системи контролю комп'ютерної системи було розпочато з висування пропозиції апаратного рішення з описом його докладних характеристик.

3. Для підтримки комп'ютерної системи вибір був зроблений на користь реалізації трьох серверів з серверними технологіями бази даних, протоколу динамічної конфігурації IP-адрес і протоколу передачі файлів. Кінцевим етапом даного розробки системи контролю є створення структурної схеми системи

контролю на якій показані засоби управління комп'ютерною системою, а так само їх розташування.

4. Розробка програмного забезпечення приводилась за допомогою сучасної платформі реалізації програмного коду використовуючи поширену мову програмування. При розробці враховувалися всі сучасні тенденції Telegram Bot API програмування. Спираючись на дані факти можна стверджувати, що розроблене програмне забезпечення в своїй мірі досить сучасне і актуальне на сьогоднішній час.

5. Корисність програмного забезпечення оцінюється його функціональністю і гнучкістю в налаштуванні, а так само безпосередньо у застосуванні.

6. При проведення експерименту було встановлено мету, завдання та обрано методику проведення експерименту. Експеримент проводився за сформульованими та визначеними вимогами, які відповідають суті експерименту. Проведений експеримент був вдалим, а дії які виконувалися у впродовж експерименту були докладно та чітко описаними з кінцевим результатом.

7. Результатом експерименту вважається виведена методом безпосередньої оцінки за допомогою моделі системи масового обслуговування та подальшими розрахунками величина з урахуванням часу, яка характеризує ефективну роботу впроваджуваної комп’ютерної системи «Консультант абітурієнта».

ПЕРЕЛІК ПОСИЛАНЬ

1. Приймальна комісія закладу вищої освіти, wikipedia.org, [Електронний ресурс, текст]. URL: https://uk.Wikipedia.org/wiki/Приймальна_комісія _закладу_вищої_освіти (дата звернення: 14.09.2020);
2. Положення про приймальну комісію Державного вищого навчального закладу «Національний гірничий університет» від 30 травня 2017 року №10, PDF;
3. Офіційна сторінка НТУ ДП у соціальній мережі Facebook, [Електронний ресурс]. URL: https://www.facebook.com/ntudp/ (дата звернення: 12.09.20);
4. Офіційна сторінка НТУ ДП у соціальній мережі Instagram, [Електронний ресурс]. URL: https://www.instagram.com/dniprotech/ (дата звернення: 13.09.20);
5. Офіційний сайт Telegtam, [Електронний ресурс]. URL: https://web.telegram.org (дата звернення: 10.10.20);
6. Офіційний сайт Flow XO, [Електронний ресурс]. URL: https://flowxo.com (дата звернення: 10.10.20);
7. Офіційний сайт Botmother, [Електронний ресурс]. URL: https://botmother.com (дата звернення: 10.10.20);
8. Офіційний сайт Gupshup, [Електронний ресурс]. URL: https://www.gupshup.io/developer/home (дата звернення: 10.10.20);
9. Стеценко, І.В. Моделювання систем: навч. посіб. [Електронний ресурс, текст] / І.В. Стеценко; М-во освіти і науки України, Черкас. держ. технол. ун-т. – Черкаси : ЧДТУ, 2010. – 399 с. ISBN 978-966-402-073-9 (дата звернення: 18.10.20);
10. Офіційний сайт Telegram Bot API, [Електронний ресурс]. URL: https://core.telegram.org/api (дата звернення: 20.10.20);
11. Офіційний сайт Node.js, [Електронний ресурс]. URL: https://nodejs.org/uk/ (дата звернення: 20.10.20);
12. Сучасний посібник JavaScript, [Електронний ресурс]. URL: https://learn.javascript.ru (дата звернення: 23.10.20);

13. Хемди А. Таха. Глава 17. Системы массового обслуживания // Введение в исследование операций. — 7-е изд. — М.: «Вильямс», 2007. — С. 629—697. (дата звернення: 24.10.20);
14. Офіційний сайт MATLAB [Електронний ресурс]. URL: https://www.mathworks.com/products/matlab.html (дата звернення: 26.10.20);
15. Simulink, wikipedia.org, [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Simulink (дата звернення: 26.10.20);
16. Комп'ютер ARTLINE Business B26v14, [Електронний ресурс]. URL: https://artline.ua/product/kompyuter-artline-business-b26v14win (дата звернення: 10.11.20);
17. Сервер DELL R730XD (24x2.5) SFF Електронний ресурс]. URL: https://server-shop.ua/server-dell-r730xd-24x25-sff-srv62.html (дата звернення: 10.11.20);
18. Офіційний сайт Atom, [Електронний ресурс]. URL: https://atom.io (дата звернення: 28.10.20);
19. Інформатика. Основи алгоритмізації: Схеми алгоритму, [Електронний ресурс]. https://yevshan.com.ua/info/006/content/content3.html (дата звернення: 03.11.20);
20. Експеримент, wikipedia.org, [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Експеримент (дата звернення: 20.11.20);
21. Методы измерений, [Електронний ресурс]. URL: https://metrcons.ru/info/articles/ (дата звернення: 21.11.20).

ДОДАТОК А

Текст програми комп’ютерної системи «Консультанта абітурієнта»

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

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ «КОНСУЛЬТАНТА АБІТУРІЄНТА»

Текст программы

804.02070743.20021-01 12 01

Листів 26

2020

АНОТАЦІЯ

Даний документ містить вихідний код програми комп’ютерної системи «Консультанта абітурієнта».

Текст програми реалізований на мові JavaScript, Node.js.

Середовище розробки та налагодження − Atom.

ЗМІСТ

Index.js

const TelegramBot 4

const bot = new TelegramBot 4

const sqlite3 4

const stage_1 4

const stage_2 4

const stage_3 5

const stage_4 5

function checkForEmptyRows 6

async function processDataShow 6

function showData 9

function recordAction 10

function editData 11

function createData 12

function handleDelete 13

function processChat 14

function addPhoto 17

function processNextStep 18

function searchForUser 19

async function searchFacultet 21

bot.on("message" 22

bot.on("callback_query" 24

Package.json 25

```
const TelegramBot = require("node-telegram-bot-api");
const bot = new TelegramBot("1329068567:AAHZAND-tgd5lDpvliw1g0Z4joYNlx9Zr6s", {
  polling: true,
});
const sqlite3 = require("sqlite3").verbose();
let db = new sqlite3.Database(
  "./db/nmu_bot.db",
  sqlite3.OPEN_READWRITE,
  (err) => {
    if (err) return console.error(err.message);
    console.log("Connected to the nmu database.");
  }
);
//Підключення до бази даних
const stage_1 = {
  text:
    "Привіт 😇 \n \n⬇ Виберіть необхідну Вам ступінь навчання ⬇ \n \n Так само можна просто відправити мені номер спеціальності яка Вас цікавить і я спробую її знайти 👀",
  data: [
    [
      {
        text: "∘ Вступ Бакалавра на базі ПЗСО ∘",
        callback_data: "0",
      },
    ],
    [
      {
        text: "∘ Вступ Бакалавра на базі ОКР ∘",
        callback_data: "1",
      },
    ],
    [
      {
        text: "∘ Вступ Магістра на базі Бакалавра ∘",
        callback_data: "2",
      },
    ],
    [
      {
        text: "👀 Пошук за спеціальностями 👀",
        callback_data: "/search",
      },
    ],
  ],
};
const stage_2 = {
  text: "\n\n∘ Вступ Бакалавра на базі ПЗСО ∘",
  data: [
    [
      {
        text: "🥏 Назад 🥏",
        callback_data: "back|0",
      },
```

```
    ],
    [
      {
        text: "Опис спеціальностей",
        callback_data: "facultets|0",
      },
    ],
    [
      {
        text: "Основна Інформація",
        callback_data: "main_info|0",
      },
    ],
  ],
};
const stage_3 = {
  text: "∘ Вступ Бакалавра на базі ОКР ∘",
  data: [
    [
      {
        text: "🔙 Назад 🔙",
        callback_data: "back|0",
      },
    ],
    [
      {
        text: "Опис спеціальностей",
        callback_data: "facultets|1",
      },
    ],
    [
      {
        text: "Основна Інформація",
        callback_data: "main_info|1",
      },
    ],
  ],
};
const stage_4 = {
  text: "\n∘ Вступ Магістра на базі Бакалавра ∘",
  data: [
    [
      {
        text: "🔙 Назад 🔙",
        callback_data: "back|0",
      },
    ],
    [
      {
        text: "Опис спеціальностей",
        callback_data: "facultets|2",
      },
    ],
```

```
    [
      {
        text: "Основна інформація",
        callback_data: "main_info|2",
      },
    ],
  ],
};
function randomInteger(min, max) {
  // Функція для генерації випадкового числа
  let rand = min - 0.5 + Math.random() * (max - min + 1);
  return Math.round(rand);
}
function checkForEmptyRows(data, table) {
  // Функція для видалення порожніх даних
  for (let item of data) {
    // Цикл для перевірки даних
    if (!item.title) {
      // Умова виконується якщо заголовок пустий
      db.run(`DELETE FROM ${table} WHERE rowid=?`, item.id, function (err) {
        // Запрос бази даних
        if (err) return console.error(err.message);
        console.log(`Paracite Row(s) deleted ${this.changes}`);
      });
    }
  }
}

async function processDataShow(id, data) {
  // Функція для відображення спеціальностей
  const schema = {
    "0": "b_school",
    "1": "b_college",
    "2": "master",
  };
  // Схема ма меню
  const back_schema = {
    0: "∘ Вступ Бакалавра на базі ПЗСО ∘",
    1: "∘ Вступ Бакалавр на базі ОКР ∘",
    2: "∘ Вступ Магістра на базі Бакалавра ∘",
  };
  // Відношення схеми меню до цифрового значення
  const m_info = data[0] == "main_info";
  // Змінна для визначення зміни основної інформації
  db.all(
    `SELECT * FROM ${schema[data[1]]} WHERE content = "${data[0]}|${data[1]}"`,
    [],
    (err, rows) => {
      checkForEmptyRows(rows, schema[data[1]]);
      // Функція видалення порожніх даних
      if (err) throw err;
      const m_info = data[0] == "main_info";
      // Змінна для визначення зміни основної інформації
```

```
    if (m_info && rows.length) {
      let main_buttons = [
        [
          {
            text: "◈ Подивитись ціни ◈",
            callback_data: `photo_price|${data[0]}|${data[1]}|text|${rows[0].id}|0`,
          },
        ],
        [
          {
            text: "◈ Назад ◈",
            callback_data: `${data[1]}`,
          },
        ],
      ];
      if (id == 360954967 || id == 192689699)
        main_buttons.push(
          [
            {
              text: "◈ Редагувати ◈",
              callback_data: `edit|${data[0]}|${data[1]}|text|${rows[0].id}|0`,
            },
          ],
          [
            {
              text: "◈ Додати фото ◈",
              callback_data: `photo|${data[0]}|${data[1]}|text|${rows[0].id}|0`,
            },
          ]
        );

      if (rows[0].image && JSON.parse(rows[0].image).length) {
        bot.sendPhoto(id, JSON.parse(rows[0].image)[0][0]);
      }
      setTimeout(() => {
        bot.sendMessage(id, rows[0].text, {
          reply_markup: {
            inline_keyboard: main_buttons,
          },
        });
      }, 100);
      return;
    }
    // Умова для відображення кнопок дії основної інформації
    if (rows.length) {
      // Умова для відображення інформації якщо вона є
      let inline = {
        text:
          "Опис спеціальностей\n \n❗Виводиться тільки 10 випадкових спеціальностей.
Скористайтеся пошуком для пошуку потрібної Вам спеціальності",
        data: [
          [
```

```
        {
          text: "🔙 Назад 🔙",
          callback_data: `${data[1]}`,
        },
        {
          text: "Пошук 🔎",
          callback_data: `search|${schema[data[1]]}|skip`,
        },
      ],
    ],
  };
  if (id == 360954967 || id == 192689699) {
    inline.data.push([
      {
        text: "🔜 Створити 🔜",
        callback_data: `create|${data[0]}|${data[1]}`,
      },
    ]);
  }
  // Умова для відображення кнопки адміністратора
  let numbers = [];
  if (rows.length > 10) {
    for (let i = 0; i < 10; i++) {
      const index = randomInteger(0, rows.length - 1);
      const err = numbers.find((el) => el == index);
      if(err == index) {
        i += - 1;
        continue
      }
      numbers.push(index)
      // При усові якщо число не повторюється
    }
  }
  // Умова для виконання циклу якщо кількість елементів більше 10
  for (let _item of rows.length > 10 ? numbers : rows) {
    let item = rows.length > 10 ? rows[_item] : _item;
    inline.data.push([
      {
        text: item.title,
        callback_data: `facultets_info|${item.id}|${data[1]}`,
      },
    ]);
  }
  // Цикл для запису даних до змінної
  bot.sendMessage(id, inline.text, {
    reply_markup: { inline_keyboard: inline.data },
  });
} else {
  // Якщо немає даних
  if (id == 360954967 || id == 192689699) {
    // Умова для виводу кнопки адміністратора
    bot.sendMessage(id, "Немає даних", {
      reply_markup: {
```

```
            inline_keyboard: [
              [
                {
                  text: "✏ Створити ✏",
                  callback_data: `create|${data[0]}|${data[1]}`,
                },
              ],
            ],
          },
        });
      } else {
        bot.sendMessage(id, "❗ Немає даних ❗");
        // Вивід повідомлення якщо ти не адміністратор
      }
    }
  }
 );
}
function showData(id, data) {
 // Функція відображення конкретної спеціальності та інформації
 let _data = [];
 const m_info = data[0] == "main_info";
 // Змінна для визначення зміни основної інформації
 const schema = {
  "0": "b_school",
  "1": "b_college",
  "2": "master",
 };
 // Відображення цифрових данних до меню
 db.all(
  `SELECT * FROM ${m_info ? schema[data[1]] : schema[data[2]]} WHERE ${
   m_info ? "content" : "id"
  } = ${m_info ? `${data[0]}|${data[1]}` : data[1]}`,
  [],
  (err, rows) => {
   if (err) throw err;
   if (rows.length) {
    // Умова наявності спеціальності
    let data_line = [
     [
      {
       text: "✏ Назад ✏",
       callback_data: `facultets|${data[2]}`,
      },
     ],
    ];
    // Змінна для кнопки "Назад"
    if (id == 360954967 || id == 192689699) {
     // Умова для відображення кнопок адміністратора
     data_line.push(
      [
       {
        text: "✍ Змінтити ✍",
```

```
            callback_data: `edit|facultets|${data[2]}|title|${data[1]}|0`,
          },
        ],
        [
          {
            text: "🖼 Додати фото 🖼",
            callback_data: `photo|facultets|${data[2]}|title|${data[1]}|0`,
          },
        ],
        [
          {
            text: "❌ Видалити ❌",
            callback_data: `delete|facultets|${data[2]}|${data[1]}`,
          },
        ]
      );
    }
    if (rows[0].image && JSON.parse(rows[0].image).length) {
      bot.sendPhoto(id, JSON.parse(rows[0].image)[0][0]);
    }
    setTimeout(() => {
      bot.sendMessage(id, rows[0].text || "❗ Немає тексту ❗", {
        reply_markup: {
          inline_keyboard: data_line,
        },
      });
    }, 200);
    // Бот відправляє повідомлення "Немає тексту", якщо немає даних
  } else {
    bot.sendMessage(id, "❗ Немає даних ❗");
    // У інших випадках бот відправляє повідомлення "Немає даних"
  }
 }
);
}
function recordAction(id, action) {
 // Функція для запису дії користувача
 let data = [action, id];
 let sql = `UPDATE users
        SET action = ?
        WHERE chat_id = ?`;
 db.run(sql, data, function (err) {
  if (err) return console.error(err.message);
  console.log(`recordAction(f) Row(s) updated: ${this.changes}`);
 });
 // Функція запиту до бази даних
}
function editData(id, splitted) {
 // Функція для редагування даних
 if (splitted[3] == "title") {
  bot.sendMessage(id, "Введіть назву блока", {
    reply_markup: {
      inline_keyboard: [
```

```
          [
            {
              text: "Пропустити",
              callback_data: `edit|${splitted[1]}|${splitted[2]}|title|${splitted[4]}|1`,
            },
          ],
        ],
      },
    });
    return recordAction(
      id,
      `edit|${splitted[1]}|${splitted[2]}|title|${splitted[4]}`
    );
    // Запис дії про редагування даних
  }
  // Умова для виводу повідомлення про зміну заголовку
  if (splitted[3] == "text") {
    bot.sendMessage(id, "Введіть новий текст", {
      reply_markup: {
        inline_keyboard: [
          [
            {
              text: "Пропустити",
              callback_data: `edit|${splitted[1]}|${splitted[2]}|text|${splitted[4]}|1`,
            },
          ],
        ],
      },
    });
    return recordAction(
      id,
      `edit|${splitted[1]}|${splitted[2]}|text|${splitted[4]}`
    );
    // Запис дії про редагування даних
  }
  // Умова для виводу повідомлення про зміну тексту
  if (splitted[3] == "2") {
    bot.sendMessage(id, "Успішно змінено ✓");
    recordAction(id, null);
    // Видалення дії редагування даних
    processDataShow(id, `${splitted[1]}|${splitted[2]}`.split("|"));
    // Відображення поточних спеціальностей
  }
}
bot.on("polling_error", console.log);
function createData(id, data) {
  // Функція створення даних
  const schema = {
    "0": "b_school",
    "1": "b_college",
    "2": "master",
  };
  // Відображення цифрових данних до меню
```

```
  if (data[3] == "text") {
    bot.sendMessage(id, "Введіть текст");
    return recordAction(id, `${data[0]}|${data[1]}|${data[2]}|text|${data[4]}`);
    // Запис дії про створення даних
  }
  // Умова виводу повідомлення про змінюваний тип даних
  if (data[3] == "2") {
    bot.sendMessage(id, "Успішно створено ✓");
    recordAction(id, null);
    // Видалення дії створення даних
    processDataShow(id, `${data[1]}|${data[2]}`.split("|"));
    // Відображення поточних спеціальностей
    return;
  }
  // Умова для відображення повідомлення по успішне створення даних
  if (data.length !== 5) {
    const m_info = data[0] == "main_info";
    // Змінна для визначення зміни основної інформації
    db.run(
      `INSERT INTO ${
        schema[data[data[0] == "main_info" ? 1 : 2]]
      }(content, text, title) VALUES(?, ?, ?)`,
      [[`${data[m_info ? 0 : 1]}|${data[m_info ? 1 : 2]}`], [""], [""]],
      function (err) {
        if (err) return console.log(err.message);
        bot.sendMessage(id, m_info ? "Введіть текст" : "Введіть заголовок");
        return recordAction(
          id,
          `create|${data[m_info ? 0 : 1]}|${data[m_info ? 1 : 2]}|${
            m_info ? "text" : "title"
          }|${this.lastID}`
        );
        // Запис дії для створення даних
        console.log(`A row has been inserted with rowid ${this.lastID}`);
      }
    );
    // Запит до бази даних
    return;
  }
  // Умова для запису бази даних
}
function handleDelete(id, data, status) {
  //Функція видалення даних
  if (typeof data == "string") data = data.split("|");
  const schema = {
    "0": "b_school",
    "1": "b_college",
    "2": "master",
  };
  // Відображення цифрових данних до меню
  if (!status) {
    bot.sendMessage(id, `Операція відмінена ✕`);
    recordAction(id, null);
```

```
    return processDataShow(id, `${data[1]}|${data[2]}`.split("|"));
    // Відобразити поточні спеціальності
  }
  // Умова для відміни дії
  if (status == "message") {
    bot.sendMessage(
      id,
      `Ви впевнені, що хочете видалити цю спеціальність?\n \nНадішліть + для підтвердження і
будь-який інший символ для скасування`
    );
    return recordAction(
      id,
      `${data[0]}|${data[1]}|${data[2]}|${data[3]}|${data[4]}`
    );
    // Запис дії видалення
  }
  // Умова для підтвердження видалення
  if (typeof status == "boolean" && status) {
    db.run(`DELETE FROM ${schema[data[2]]} WHERE rowid=?`, data[3], function (
      err
    ) {
      if (err) return console.error(err.message);
      recordAction(id, null);
      // Видалення дії
      bot.sendMessage(id, `Операція успішна ✓`);
      return processDataShow(id, `${data[1]}|${data[2]}`.split("|"));
      // Відображення поточних спеціальностей
      console.log(`Row(s) deleted ${this.changes}`);
    });
  }
  // Умова запиту до бази даних для видалення
}
function writeToUser(id, user) {
  // Функція виводу дії для користувача
  const _user = user.split("|");
  bot.sendMessage(id, `Дія для юзера ${_user[2]}`, {
    reply_markup: {
      inline_keyboard: [
        [
          {
            text: "Почати чат",
            callback_data: `chat|${_user[1]}|${_user[2]}`,
          },
          {
            text: "Видалити",
            callback_data: `delete|user|${_user[1]}`,
          },
        ],
      ],
    },
  });
  // відправлення повідомлення з кнопками дії
}
```

```
function deleteUser(id, data, status) {
  // Функція видалення користувача
  if (typeof status == "number" && status == 0) {
    bot.sendMessage(
      id,
      `Ви впевнені, що хочете видалити ${
        data.split("|")[2]
      }?\n \nНадішліть + для підтвердження і будь-який інший символ для скасування`
    );
    recordAction(id, data);
    // Запис дії для видалення
    return;
  }
  // Умова виводу повідомлення про видалення
  if (typeof status == "boolean" && !status) {
    bot.sendMessage(id, "Дія відмінена");
    recordAction(id, null);
    // Видалення дії
    return;
  }
  // Умова для виводу повідомлення про відміну дії
  db.run(`DELETE FROM users WHERE chat_id=?`, data.split("|")[2], function (
    err
  ) {
    if (err) return console.error(err.message);
    bot.sendMessage(id, "Успішно видалено");
    // Повідомлення про видалення
    console.log(`Paracite Row(s) deleted ${this.changes}`);
  });
  // Запит у базу даних для видалення користувача
}
function processChat(id, data, message) {
  // Керування зворотним зв'язком
  let user = data.split("|");
  // Обробка данних
  if (message) {
    bot.sendMessage(
      +user[1],
      `${!user[2] ? "Адміністратор" : user[2]}: ${message}`
    );
    return;
  }
  // Відправлення повідомлення користувачу на отримання повідомлення від адміністратора
  if (id == +user[1]) {
    bot.sendMessage(id, "Ви не можете почати чат з собою");
    recordAction(id, null);
    // Видалення поточної дії
    showListOfPeople(id, "");
    return;
  }
  // Умова обмежуюча початок чату з самим собою
  if (user[2] == "declined") {
    bot.sendMessage(id, "Чат відхилено.");
```

```
    bot.sendMessage(+user[1], "Чат відхилено користувачем.");
    recordAction(id, null);
    // Видалення поточної дії
    return;
  }
  // Умова відміни чату
  if (user[2] == "accept") {
    bot.sendMessage(id, "Чат прийнятий. Просто пишіть сюди свои питання.");
    bot.sendMessage(+user[1], "Чат прийнятий користувачем.");
    recordAction(id, `chat|${+user[1]}|${user[3]}`);
    // Запис дії для адміністратора
    recordAction(+user[1], `chat|${id}`);
    // Запис дії для користувача
    return;
  }
  // Умова підтвердження зворотного зв'язку
  if (!message) {
    bot.sendMessage(id, "Запрос відправлено");
    bot.sendMessage(
      +user[1],
      "Адміністатору є що Вам сказати. Нажміть кнопку для початку чата",
      {
        reply_markup: {
          inline_keyboard: [
            [
              {
                text: "Почати чат",
                callback_data: `chat|${id}|accept|${user[2]}`,
              },
              {
                text: "Відхилити чат",
                callback_data: `chat|${id}|declined`,
              },
            ],
          ],
        },
      }
    );
  }
  // Умова відправлення запиту на зворотний зв'язок
  recordAction(id, data);
  // Запис дії для роботи з зворотним зв'язком
}
function addPhotoPrice(id, data, status) {
  const splitted = data.split("|");
  const schema = {
    "0": "b_school",
    "1": "b_college",
    "2": "master",
  };
  db.all(
    `SELECT * FROM ${schema[splitted[2]]} WHERE id = "${splitted[4]}"`,
    [],
```

```
function (err, rows) {
  if (err) console.log(err);
  const price_callback = [
    {
      text: "Назад",
      callback_data: `main_info|${splitted[2]}`,
    },
  ];
  if (id == 360954967 || id == 192689699)
    price_callback.push({
      text: "Поменять фото",
      callback_data: `photo_price|${splitted[1]}|${splitted[2]}|change|${splitted[4]}|0`,
    });
  if (
    rows[0].image != null &&
    JSON.parse(rows[0].image).length == 2 &&
    JSON.parse(rows[0].image)[1].length &&
    JSON.parse(rows[0].image)[1][0].length &&
    splitted[3] !== "change"
  ) {
    bot.sendPhoto(id, JSON.parse(rows[0].image)[1][0], {
      reply_markup: {
        inline_keyboard: [price_callback],
      },
    });
  } else {
    if (!status || !status.photo) {
      bot.sendMessage(
        id,
        "Відправте мені фото для його подальшого збереження",
        {
          reply_markup: {
            inline_keyboard: [
              [
                {
                  text: "Скасувати операцію",
                  callback_data: `cancel?${data}`,
                },
              ],
            ],
          },
        }
      );
      return recordAction(id, data);
    }
    const first_image =
      rows[0].image != null ? JSON.parse(rows[0].image)[0][0] : "";
    db.run(
      `UPDATE ${schema[splitted[2]]}
            SET image = ?
            WHERE id = ?`,
      [
        [JSON.stringify([[first_image], [status.photo[0].file_id]])],
```

```
          [splitted[4]],
        ],
        function (err) {
          if (err) console.log(err);
          bot.sendMessage(id, "Фото успішно додано.");
          recordAction(id, null);
          return processDataShow(
            id,
            `${splitted[1]}|${splitted[2]}`.split("|")
          );
          console.log(`added photo to id ${splitted[4]}`);
        }
      );
    }
  }
 );
}
function addPhoto(id, data, status) {
 if (!status || !status.photo) {
  bot.sendMessage(id, "Відправте мені фото для його подальшого збереження", {
    reply_markup: {
      inline_keyboard: [
        [
          {
            text: "Скасувати операцію",
            callback_data: `cancel?${data}`,
          },
        ],
      ],
    },
  });
  return recordAction(id, data);
 }
 const schema = {
  "0": "b_school",
  "1": "b_college",
  "2": "master",
 };
 const splitted = data.split("|");
 db.all(
  `SELECT * FROM ${schema[splitted[2]]} WHERE id = "${splitted[4]}"`,
  [],
  (err, rows) => {
    console.log(rows);
    const price =
      rows[0].image != null && JSON.parse(rows[0].image)[1][0].length
        ? JSON.parse(rows[0].image)[1]
        : [];
    // console.log(rows[0].image);
    const image =
      splitted[1] == "main_info"
        ? [[status.photo[0].file_id], price]
        : [[status.photo[0].file_id]];
```

```
      db.run(
        `UPDATE ${schema[splitted[2]]}
              SET image = ?
              WHERE id = ?`,
        [[JSON.stringify(image)], [splitted[4]]],
        function (err) {
          if (err) console.log(err);
          bot.sendMessage(id, "Фото успішно додано.");
          recordAction(id, null);
          return processDataShow(
            id,
            `${splitted[1]}|${splitted[2]}`.split("|")
          );
          console.log(`added photo to id ${splitted[4]}`);
        }
      );
    }
  );
  // console.log(data, status);
}

function processNextStep(data, id) {
  // Функція виявлення наступного кроку
  let _data = [];
  let splitted = [];
  if (data.includes("cancel")) return deleteAction(id, data);
  // Оголошення змінних
  if (!data) return bot.sendMessage(id, "✕ Помилка ✕");
  // Умова відображає помилку
  if (data.includes("|")) {
    // Умова виявляє наявність дії
    splitted = data.split("|");
    // Змінна розділення даних
    if (splitted[0] == "edit") {
      return editData(id, splitted);
    }
    if (splitted[0] == "photo_price") return addPhotoPrice(id, data, false);

    if (splitted[0] == "photo") return addPhoto(id, data, false);
    // Виклик Функції обробки данних
    if (splitted[0] == "user") return writeToUser(id, data);
    // Виклик Функції зворотного зв'язку
    if (splitted[0] == "create") return createData(id, splitted);
    // Виклик Функції створення даних
    if (splitted[0] == "facultets_info") return showData(id, splitted);
    // Виклик Функції відображення даних
    if (splitted[0] == "facultets" || splitted[0] == "main_info")
      return processDataShow(id, splitted);
    // Виклик Функції відображення спеціальностей та основної інформації
    if (splitted[0] == "back" && splitted[1] == "0") _data = stage_1;
    // Запис данних у змінну для повернення у головне меню
    if (splitted[0] == "delete" && splitted[1] == "user")
      return deleteUser(id, data, 0);
```

```
    // Виклик Функції видалення користувача
    if (splitted[0] == "delete") return handleDelete(id, splitted, "message");
    // Виклик Функції видалення данних
    if (splitted[0] == "chat") return processChat(id, data, false);
    // Виклик Функції керування зворотним зв'язком
  } else {
    switch (data) {
      case "0":
        _data = stage_2;
        break;
      case "1":
        _data = stage_3;
        break;
      case "2":
        _data = stage_4;
        break;
      case "hello":
        recordAction(id, null);
        // bot.sendPhoto(id, "./img/logo_m.png");
        _data = stage_1;
        break;
      default:
    }
    // Обробка навігації головного меню
  }
  if (_data.data) {
    bot.sendMessage(id, _data.text, {
      reply_markup: {
        inline_keyboard: _data.data,
      },
    });
    return;
  }
  // Відображення головного меню
  bot.sendMessage(id, _data.text);
  // Відображення тексту
}
function searchForUser(id, callback) {
  // Пошук користувача
  let chat_id = id
    ? `SELECT * FROM users WHERE chat_id = ${id}`
    : "SELECT * FROM users";
  db.all(`${chat_id}`, [], (err, rows) => {
    if (err) throw err;
    callback(rows);
  });
  // запит до бази данних
}
function updateData(id, _data, text) {
  // Функція оновлення даних
  if (!text)
    return processNextStep(
      `${_data[0]}|${_data[1]}|${_data[2]}|${
```

```
        _data[3] == "text" ? "2" : "text"
      }|${_data[4]}`,
      id
    );
  // Умова якщо немає тексту відобразити таступний крок
  const schema = {
    "0": "b_school",
    "1": "b_college",
    "2": "master",
  };
  // Відображення цифрових данних до меню
  let _item = _data.split("|");
  // Обробка даних
  if (_item[3] == "title" && text.length > 60) {
    bot.sendMessage(id, "Перевищено максимальну довжину заголовка!");
    editData(id, _item);
    return;
  }
  // Умова обмежуюча максимальну довжину заголовка
  let data = [text, _item[4]];
  let sql = `UPDATE ${schema[_item[2]]}
            SET ${_item[3]} = ?
            WHERE id = ?`;
  // Змінні, для уніфікації запиту бази даних
  db.run(sql, data, function (err) {
    if (err) return console.error(err.message);
    console.log(`updateData(f) Row(s) updated: ${this.changes}`);
  });
  // запит на зміну до бази даних
  processNextStep(
    `${_item[0]}|${_item[1]}|${_item[2]}|${_item[3] == "text" ? 2 : "text"}|${
      _item[4]
    }`,
    id
  );
  // Виявлення наступного кроку
}
function showListOfPeople(id, search) {
  // Функція відображення користувачів
  if (id !== 360954967)
    return bot.sendMessage(
      id,
      "Необхідно бути адміністратором щоб виконати цю команду!"
    );
  // Умова що відображає повідомлення
  searchForUser(search, function (rows) {
    // Функція пошуку користувачів
    let inline = {
      text: "Список юзерів бота",
      data: [
        [
          {
            text: "🥷 До головного меню 🥷",
```

```
            callback_data: `hello`,
          },
        ],
      ],
    };
    for (let _item of rows) {
      inline.data.push([
        {
          text: `${_item.first_name} ${_item.last_name}`,
          callback_data: `user|${_item.chat_id}|${_item.first_name} ${_item.last_name}`,
        },
      ]);
    }
    // Відношення користувачів до змanної
    bot.sendMessage(id, inline.text, {
      reply_markup: {
        inline_keyboard: inline.data,
      },
    });
    // відображення користувачів
  });
}
function deleteAction(id, data) {
  recordAction(id, null);
  bot.sendMessage(id, "Операція скасована!");
  // processNextStep(data.split("?")[1], id);
}
async function searchFacultet(id, text, status) {
  // Функція пошуку Факультету
  if (!status || (typeof status == "object" && status[2] == "skip")) {
    const record_data =
      typeof status == "object" ? `search|${status[1]}` : `search`;
    recordAction(id, record_data);
    bot.sendMessage(id, "Введіть номер спеціальності чи просто назву");
    // Умова виводу повідомлення про початий пошук та запис дії пошуку
    return;
  }
  const schema = {
    b_school: "∘ Вступ Бакалавра на базі ПЗСО ∘",
    b_college: "∘ Вступ Бакалавр на базі ОКР ∘",
    master: "∘ Вступ Магістра на базі Бакалавра ∘",
  };
  // Схема меню
  const tableSchema = {
    b_school: "0",
    b_college: "1",
    master: "2",
  };
  // Відношення схеми меню до цифрового значення
  let _status = typeof status == "object" && status[1];
  // Змінна поточного статусу пошуку
  bot.sendMessage(
    id,
```

```
    !_status
      ? "Назад до головного меню"
      : `Назад до спеіальностей ${schema[status[1]]}`,
    {
      reply_markup: {
        inline_keyboard: [
          [
            {
              text: !_status ? "Меню 🏠" : "🌧 Назад 🌧",
              callback_data: !_status
                ? "hello"
                : `facultets|${tableSchema[status[1]]}`,
            },
          ],
        ],
      },
    }
  );
  // Вивід даних навігації спеціальностей у відношенні від глобального пошуку та локального
  //Різниця між глобалним та локальним пошуком
  setTimeout(() => {
    // Затримка на 500 мл. сек.
    for (let item of !_status
      ? ["b_school", "b_college", "master"]
      : [status[1]]) {
      db.all(`SELECT * FROM ${item}`, [], (err, rows) => {
        let output = {
          text: `🔻 ${schema[item]} 🔻`,
          data: [],
        };

        for (let el of rows) {
          if (el.title.includes(text))
            output.data.push([
              {
                text: el.title,
                callback_data: `facultets_info|${el.id}|${tableSchema[item]}`,
              },
            ]);
        }
        if (output.data.length) {
          bot.sendMessage(id, output.text, {
            reply_markup: {
              inline_keyboard: output.data,
            },
          });
        }
      });
    }
    // Пошук спеціальності, глобальний і локальний
  }, 500);
}
bot.on("message", async (res) => {
```

```
// Виклик функції при відправці повідомлення
const id = res.chat.id;
// запис унікального ключа користувача
if (res.text == "/users") return showListOfPeople(id, "");
// Виклик функції відображення користувачів
if (res.text == "/search") return searchFacultet(id, res.text, false);
if (res.text == "/start" || res.text == "/menu")
  return processNextStep("hello", id);
// Виклик функції пошуку за спеціальністю, вивід тексту з поясненням
searchForUser(id, function (rows) {
  if (rows.length && rows[0].action !== null) {
    // Умова виконання текстових команд користувача
    const action = rows[0].action;
    const splitted = action.split("|");
    if (splitted[0] == "photo") return addPhoto(id, action, res);
    if (splitted[0] == "photo_price") return addPhotoPrice(id, action, res);
    // Обробка данних
    if (splitted.length > 1 && splitted[1] == "user")
      return deleteUser(id, rows[0].action, !!(res.text == "+"));
    // Виклик функції видалення користувача
    if (
      action == "search" ||
      (splitted.length > 1 && splitted[0] == "search")
    )
      return searchFacultet(
        id,
        res.text,
        splitted.length > 1 ? splitted : true
      );
    // Виклик функції пошук за спеціальністю, пошук та вивід шуканої інформації
    if (splitted[0] == "chat") return processChat(id, action, res.text);
    // Функція чату з користувачем
    if (action !== null && action.split("|")[0] == "delete")
      return handleDelete(id, action, res.text == "+");
    // Видалення данних
    if (action !== null) return updateData(id, action, res.text);
    // Функція оновлення даних спеціальностей и основної інформації
    return;
  } else {
    if (!rows.length) {
      // Умова відсутності користувача бази даних
      db.run(
        `INSERT INTO users(first_name, last_name, chat_id, language) VALUES(?, ?, ?, ?)`,
        [
          [res.chat.first_name],
          [res.chat.last_name],
          [id],
          [res.chat.language_code],
        ],
        function (err) {
          if (err) return console.log(err.message);
          processNextStep("hello", id);
        }
```

```
        );
        // Додавання користувача у базу даних
        return;
      }
      if (res.text == "/start" || res.text == "/menu")
        return processNextStep("hello", id);
      // Вивід меню
      searchFacultet(id, res.text, true);
      // Пошук спеціальностей по тексту користувача
      // Виконається якщо не має інших поточних задач
    }
  });
});
bot.on("callback_query", (query) => {
  // функція при натисканні кнопки
  const id = query.message.chat.id;
  const splitted = query.data.split("|");
  const search = splitted.length > 1 && splitted[0] == "search";
  // Запис данних у змінну
  if (splitted.length > 1 && splitted[5] == "1")
    return updateData(id, splitted, false);
  // Виклик функції оновлення даних
  if (query.data == "/search" || search)
    return searchFacultet(id, "", search ? splitted : false);
  // Виклик функції пошуку факультету
  processNextStep(query.data, id);
  // Виклик функції визначення наступного кроку
});
```

```
{
  "name": "NmuHTBot",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "mysql": "^2.18.1",
    "node-telegram-bot-api": "^0.30.0",
    "sqlite3": "^5.0.0"
  }
}
```