ДОДАТОК А

Апробація

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(КІТАР)

МАТЕРІАЛИ

ІІ Всеукраїнської конференції
«Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки»
(Computer-integrated technologies, automation and robotics)
CITAR`25
16-17 травня 2025

[електронне видання]

Харків 2025

Рисунок А.1 – Титульна сторінка збірника CITAR

АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ ВЕБСАЙТУ-ПОМІЧНИКА АБІТУРІЄНТА: ЧОМУ NEXT.JS ТА GOOGLE ТАБЛИЦІ – ОПТИМАЛЬНЕ РІШЕННЯ

Д.В. Лукієнко, Д.В Гурін
Харківський національний університет радіоелектроніки
Україна, 61166, Харків, пр. Науки, 14
Email: dmytro.lukiienko@nure.ua, dmutro.gurin@nure.ua

Анотація. У тезах розглядається процес розробки вебсайту, спрямованого на абітурієнтів для обрання спеціальності, який вимагає тщательного вибору технологічного стеку. В роботі було обрано Next.js для фронтенду та Google Таблиці для зберігання та обробки даних тестування. Next.js вибрано через його високу продуктивність та SEO-оптимізацію, що робить сайт швидким та добре індексованим пошуковими системами. Використання Google Таблиць дозволяє ефективно керувати контентом без необхідності повноцінної бази даних, що є економічно вигідним рішенням на початкових етапах проєкту. Висновки підкреслюють, що такий підхід забезпечує швидкий доступ до актуальної інформації для користувачів та спрощує адміністрування сайту.

Ключові слова: SSG, Next.js, Google Таблиці, веб.

TECHNOLOGY ANALYSIS FOR A STUDENT ASSISTANT WEBSITE: WHY NEXT.JS AND GOOGLE SHEETS ARE THE OPTIMAL SOLUTION

D. Lukiienko, D.Gurin
Kharkiv National University of Radio Electronics
Ukraine, 61166, Kharkiv, Nauky av., 14
Email: dmytro.lukiienko@nure.ua, dmutro.gurin@nure.ua

Abstract. The abstracts consider the process of developing a website aimed at applicants for choosing a specialty, which requires careful selection of the technology stack. The work selected Next.js for the frontend and Google Sheets for storing and processing test data. Next.js was chosen due to its high performance and SEO optimization, which makes the site fast and well indexed by search engines. Using Google Sheets allows for effective content management without the need for a full-fledged database, which is a cost-effective solution at the initial stages of the project. The conclusions emphasize that this approach provides quick access to relevant information for users and simplifies site administration.

***Keywords*:** SSG, Next.js, Google Sheets, web.

Розробка вебсайту, що допомагатиме абітурієнтам у виборі спеціальності, потребує ретельно підібраного технологічного стека, який забезпечить швидку роботу, гнучкість у розширенні функціоналу та зручність в адмініструванні контенту. Враховуючи ці вимоги, було обрано Next.js як основний фреймворк для фронтенду, а для зберігання даних тестування – Google Таблиці. Таке рішення дозволяє створити продуктивний, легкий у підтримці та економічно вигідний проєкт, який відповідатиме всім вимогам як користувачів, так і розробників.

Основною причиною вибору Next.js є його висока продуктивність та SEO-оптимізація. Оскільки сайт буде орієнтований на абітурієнтів, важливо, щоб сторінки швидко завантажувалися і добре індексувалися пошуковими системами. Завдяки підтримці серверного рендерингу (SSR) та статичної генерації (SSG) Next.js дозволяє оптимізувати контент для пошукових алгоритмів, що підвищить його видимість у пошукових системах. Крім того, це позитивно вплине на досвід користувачів – сторінки відкриватимуться миттєво, без зайвих

Рисунок А.2 – Перша сторінка наукової статті

затримок, що особливо важливо для молодіжної аудиторії, яка звикла до швидкої взаємодії з веб-ресурсами.

Ще одним важливим фактором є мінімізація зовнішніх залежностей. Використовуючи Next.js, ми можемо уникнути перевантаження сайту зайвими бібліотеками, що позитивно вплине на швидкість завантаження сторінки. Легка вага бандлу дозволяє забезпечити стабільну роботу навіть на повільному інтернет-з'єднанні, що робить сайт доступним для всіх користувачів. Такий підхід допомагає не лише покращити продуктивність, а й оптимізувати витрати на хостинг і серверні ресурси.

Також важливою перевагою використання Next.js є його підтримка глобального кешування та оптимізація ресурсів. Це означає, що сторінки, які не змінюються часто, можуть бути збережені в кеші та віддаватися користувачам без додаткових запитів до сервера. Такий підхід значно зменшує навантаження на сервер і покращує загальну швидкодію сайту. Крім того, Next.js має вбудовані інструменти для автоматичного розбиття коду (code splitting), що допомагає завантажувати лише ті компоненти, які необхідні для конкретної сторінки, забезпечуючи швидке завантаження та плавну навігацію по сайту.

Для зберігання та обробки даних тестування було обрано Google Таблиці – це простий, швидкий та бюджетний варіант. Вони дозволяють зручно керувати тестовими питаннями, списками спеціальностей та результатами без необхідності розгортати повноцінну базу даних. Завдяки API Google Sheets можна легко інтегрувати таблиці з вебсайтом, що дозволяє в реальному часі оновлювати інформацію без внесення змін у код. Якщо університет змінює набір спеціальностей або коригує питання тесту, адміністратор може оновити дані в таблиці, і вони миттєво відобразяться на сайті.

Ще однією перевагою Google Таблиць є їх економічність. Використання цього інструменту дозволяє уникнути витрат на окремий сервер для бази даних, що особливо важливо на початкових етапах проєкту. Крім того, це знижує витрати на підтримку, оскільки адміністратори можуть самостійно керувати вмістом сайту без залучення розробників. Такий підхід дозволяє зосередитися на вдосконаленні основного функціоналу вебсайту без додаткових технічних складнощів.

ВИСНОВКИ. Поєднання Next.js та Google Таблиць створює ідеальний баланс між продуктивністю, зручністю та економічною ефективністю. Використання сучасного фронтенд-фреймворку разом із легким та доступним способом керування контентом дозволяє розробити платформу, яка буде не лише швидкою та зручною для користувачів, але й простою в адмініструванні. Такий підхід забезпечує абітурієнтам швидкий доступ до актуальної інформації, спрощуючи процес ознайомлення зі спеціальностями та можливостями навчання.

ЛІТЕРАТУРА

1. Next.js Documentation [Електронний ресурс] – Режим доступу: https://nextjs.org/docs. – Дата звернення: 29.03.2025.

2. Google Sheets API Documentation [Електронний ресурс] – Режим доступу: https://developers.google.com/sheets/api. – Дата звернення: 29.03.2025.

3. Веб-оптимізація та продуктивність: принципи та найкращі практики [Електронний ресурс] – Режим доступу: https://web.dev. – Дата звернення: 29.03.2025.

4. SEO та продуктивність сайтів [Електронний ресурс] – Режим доступу: https://moz.com/learn/seo/page-speed. – Дата звернення: 29.03.2025.

5. Web site reliability analysis using the python parsing method / D. Gurin, S. Maksymova, V. Yevsieiev, Ahmad Alkhalaileh // Journal of Universal Science Research. – 2024. – Vol. 2(5). – P. 113-126.

Рисунок А.3 – Друга сторінка наукової статті

6. Roca, M. D. L., Chan, M. M., Garcia-Cabot, A., Garcia-Lopez, E., & Amado-Salvatierra, H. (2024). The impact of a chatbot working as an assistant in a course for supporting student learning and engagement. *Computer Applications in Engineering Education*, *32*(5), e22750.

7. Yevsieiev V. The Solution to the Task of Analyzing the Reliability of Web Sites Using Python Parsing / V. Yevsieiev, S. Uskov // Information technologies and systems : proceedings of the VI International Scientific and Theoretical Conference, July 12, 2024. - Amsterdam, 2024. - Section 6. - P. 51-53

8. Web site reliability analysis using the python parsing method / D. Gurin, S. Maksymova, V. Yevsieiev, Ahmad Alkhalaileh // Journal of Universal Science Research. – 2024. – Vol. 2(5). – P. 113-126.

9. Yevsieiev V. A Program for Analyzing the Structure of a Web site Development Using the Parsing Method Based on the Python / V. Yevsieiev, S. Maksymova, Ahmad Alkhalaileh // Journal of Universal Science Research. – 2024. – Vol. 2(4). – P. 172-183.

10. Yevsieiev, V., & et al. (2025). Development of a program for processing 3d models of objects in a collaborative robot workspace using an HD camera. ACUMEN: International journal of multidisciplinary research, 2(1), 194-210.

11. Gurin, D., & et al. (2024). Effect of Frame Processing Frequency on Object Identification Using MobileNetV2 Neural Network for a Mobile Robot. Multidisciplinary Journal of Science and Technology, 4(8), 36-44.

12. Attar, H., Abu-Jassar, A. T., Amer, A., Lyashenko, V., Yevsieiev, V., & Khosravi, M. R. (2022). Control system development and implementation of a CNC laser engraver for environmental use with remote imaging. Computational intelligence and neuroscience, 2022(1), 9140156.

13. Nevliudov, I., & et al.. (2020). Method of Algorithms for Cyber-Physical Production Systems Functioning Synthesis. International Journal of Emerging Trends in Engineering Research, 8(10), 7465-7473.

14. Abu-Jassar AT, Attar H, Amer A, et al. Remote Monitoring System of Patient Status in Social IoT Environments Using Amazon Web Services (AWS) Technologies and Smart Health Care. International Journal of Crowd Science, 2024

15. Abu-Jassar A. Building a Route for a Mobile Robot Based on the BRRT and A*(H-BRRT) Algorithms for the Effective Development of Technological Innovations / Amer Abu-Jassar, Hassan Al-Sukhni, Yasser Al-Sharo, S. Maksymova, V. Yevsieiev, V. Lyashenko // International Journal of Engineering Trends and Technology. – 2024. – V. 72(11). – P. 294-306.

16. Yevsieiev, V., Alkhalaileh, A., Maksymova, S., & Gurin, D. (2024). Research of Existing Methods of Representing a Collaborative Robot-Manipulator Environment within the Framework of Cyber-Physical Production Systems. *Multidisciplinary Journal of Science and Technology*, *4*(9), 112-120.

17. Vladyslav Yevsieiev, Nikolaj Starodubcev (2023). Development of a control algorithm for a small-sized mobile manipulation robot. Proceedings of the 2nd International Scientific and Practical Conference«Diversity and Inclusion in Scientific Area», Value 140, P.648-651

18. Yevsieiev, V., & Gurin, D. (2023). Comparative Analysis of the Basic Methods Used in Industry 4.0 and Industry 5.0. Collection of Scientific Papers «ΛΌГОΣ», (Bologna, Italy), 113–115. https://doi.org/10.36074/logos-29.09.2023.31

Рисунок А.4 – Третя сторінка наукової статті

ДОДАТОК Б

Код проекту

```
# ChatWidget.tsx
const ChatWidget = () => {
  const { open, openChat, quizOpen } = useChatWidget();


  return (
    <div className="pointer-events-none">
      <AnimatePresence mode="wait">
        {open && (
          <motion.div
            key="chat"
            initial={{ opacity: 0, scale: 0.9, y: 40 }}
            animate={{ opacity: 1, scale: 1, y: 0 }}
            exit={{ opacity: 0, scale: 0.9, y: 40 }}
            transition={{ type: 'spring', stiffness: 260, damping: 24 }}
            layout
            className={clsx(
              'fixed z-50 pointer-events-auto',
              'bg-card bg-background overflow-hidden',
              'w-full h-full right-0 bottom-0',
              quizOpen ? 'md:w-180 md:h-145' : 'md:w-80 md:h-115',
              'md:bottom-6 md:right-6 md:shadow-xl md:rounded-2xl md:max-w-[90vw] md:max-h-[90vh]'
            )}
          >
            {<Chat />}
          </motion.div>
```

```
        )}
      </AnimatePresence>

      {!open && <ChatOpenButton openChat={openChat} />}
    </div>
  );
};

export default ChatWidget;

# Chat.tsx
const Chat: FC<IChatProps> = ({ withoutHeader }) => {
  const t = useTranslations('ChatWidget');

  const {
    handleSendMessage,
    loading,
    inputVisible,
    conversation,
    context,
    setContext,
    ref,
    scrollToBottom,
    closeConversation,
    setInputVisible,
    openQuiz,
    quizOpen,
    quizConversation,
    askNextQuizQuestion,
```

```
  selectQuizAnswer,
} = useChatWidget();

return (
  <div className="w-full h-full flex flex-col overflow-hidden bg-transparent">
    {/* Header */}
    {!withoutHeader && (
      <div
        className={clsx(
          'bg-secondary text-fg-secondary',
          'flex items-center justify-between p-4'
        )}
      >
        <h4 className="text-base font-medium">{t('widgetTitle')}</h4>

        <button onClick={closeConversation} className="cursor-pointer">
          <X className="w-5 h-5" />
        </button>
      </div>
    )}

    {/* Messages */}
    <div
      ref={ref}
      className="flex-1 overflow-y-auto px-4 py-3 space-y-4 text-sm"
    >
      {quizOpen
        ? quizConversation.map((msg, index) => (
            <QuizMessageDisplay
```

```
          key={'test' + index}
          {...msg}
          scrollToBottom={scrollToBottom}
          setInputVisible={setInputVisible}
          callback={
            quizConversation.length > 1 || index !== 0
              ? undefined
              : () => askNextQuizQuestion()
          }
        />
      ))
    : conversation.map((msg, index) => (
        <ChatMessageDisplay
          key={'chat' + index}
          {...msg}
          setContext={setContext}
          scrollToBottom={scrollToBottom}
          setInputVisible={setInputVisible}
          openTest={openQuiz}
        />
      ))}

  {loading && (
    <div className="ml-1">
      <ChatLoader />
    </div>
  )}
</div>
```

```
      <AnimatePresence>
        {inputVisible ? (
          quizOpen ? (
            <QuizChoices
              answers={quizConversation[quizConversation.length - 1].answers}
              selectAnswer={selectQuizAnswer}
              multiSelect={
                quizConversation[quizConversation.length - 1].multiselect
              }
            />
          ) : (
            <ChatInput
              maxHeight={120}
              handleSendMessage={handleSendMessage}
              context={context}
              onClearContext={() => setContext(null)}
            />
          )
        ) : null}
      </AnimatePresence>
    </div>
  );
};

export default Chat;
# ChatWidgetContext.tsx
const ChatWidgetContext = createContext<ChatWidgetContextType | undefined>(
  undefined
);
```

```
export const ChatWidgetProvider = ({ children }: { children: ReactNode }) => {
  const t = useTranslations('ChatWidget');

  const startMessage = {
    role: 'assistant',
    text: t('startMessage'),
  } as IChatMessage;

  const quizStartMessage = {
    role: 'assistant',
    text: t('quiz.startMessage'),
  } as IChatMessage;

  const [open, setOpen] = useState(false);

  const openChatWidget = () => setOpen(true);
  const closeChatWidget = () => setOpen(false);

  const [loading, setLoading] = useState(false);
  const [inputVisible, setInputVisible] = useState(false);
  const [quizOpen, setQuizOpen] = useState(false);
  const [conversation, setConversation] = useState<IChatMessage[]>([
    startMessage,
  ]);
  const [quizConversation, setQuizConversation] = useState<IQuizMessage[]>([
    quizStartMessage,
  ]);
  const [context, setContext] = useState<string | null>(null);
```

```
const [quizInterests, setQuizInterests] = useState<InterestsRecord>({});
const [dislikedInterests, setDislikedInterests] = useState<string[]>([]);
const { ref, scrollToBottom } = useScrollToBottom<HTMLDivElement>();

const closeConversation = () => {
  closeChatWidget?.();
  setConversation((prev) => prev.map((msg) => ({ ...msg, read: true })));
  setQuizConversation((prev) =>
    prev.length === 1
      ? [quizStartMessage]
      : prev.map((msg) => ({ ...msg, read: true }))
  );
  setContext(null);
  setInputVisible(true);
  setLoading(false);
};

const handleSendMessage = async (message: string) => {
  try {
    setInputVisible(false);
    setConversation((prev) => [
      ...prev,
      { role: 'user', text: message, read: true },
    ]);
    setLoading(true);
    scrollToBottom({ force: true, smooth: true, delayMs: 100 });

    const response = await axios.post('/api/chat/ask', {
      question: context ? `${message} ${context}` : message,
```

```
  result_limit: context ? 2 : 1,
});

if (!response.data) return;

let data = response.data[0];

if (context) {
  const previousAssistantMessage = conversation
    .filter((q) => q.role === 'assistant')
    .map((q) => q.text);
  data = response.data.filter(
    (q: IQuestion) => !previousAssistantMessage.includes(q.answer)
  )[0];
}

if (data.metadata?.clarifyingQuestion && data.metadata?.context) {
  setContext(data.metadata?.context);
}

setConversation((prev) => [
  ...prev,
  {
    role: 'assistant',
    text: data.answer,
    metadata: data.metadata,
    similarity: data.similarity,
  },
]);
```

```
  } catch {
    setInputVisible(true);
  } finally {
    setLoading(false);
  }
};

const openQuiz = () => {
  setQuizOpen(true);
  openChatWidget();
};

const openChat = () => {
  setQuizOpen(false);
  openChatWidget();
  if (
    quizConversation.length > 1 &&
    conversation[conversation.length - 1].text !==
      t('quiz.resume_prompt_started')
  ) {
    setConversation((prev) => [
      ...prev,
      {
        text: t('quiz.resume_prompt_started'),
        role: 'assistant',
        metadata: { testTrigger: true },
        quizButtonLabel: t('quiz.resume_button'),
      },
    ]);
```

```
    }
  };

  const askChat = async (message: string) => {
    if (conversation.length === 1 && conversation[0].role === 'assistant') {
      setConversation([]);
    }
    openChat();
    if (loading) return;
    setContext(null);
    handleSendMessage(message);
  };

  const sendQuizResultMessage = async (message: string) => {
    setLoading(true);
    scrollToBottom({ force: true, smooth: true, delayMs: 100 });
    await new Promise((resolve) => setTimeout(resolve, 1000));
    setLoading(false);
    setConversation((prev) => [...prev, { role: 'assistant', text: message }]);
    scrollToBottom({ force: true, smooth: true, delayMs: 100 });
    await new Promise((resolve) =>
      setTimeout(resolve, CHAT_MESSAGE_DISPLAY_DELAY_MS *
message.length + 500)
    );
  };

  function clearResumeButtonMessages() {
    const filtered = conversation.filter(
      (msg) => msg.quizButtonLabel !== t('quiz.resume_button')
```

```
  );

  if (
    filtered.length === 1 &&
    filtered[0].role === 'assistant' &&
    filtered[0].text === startMessage.text
  ) {
    return [];
  }
  return filtered;
}

const showQuizResults = async () => {
  setLoading(false);
  setConversation(clearResumeButtonMessages());

  setQuizOpen(false);
  setQuizConversation([]);

  await sendQuizResultMessage(t('quiz.suggestion_main'));

  const abilitiesEducationPrograms = EducationPrograms.map((program) => {
    const hasAllTopics = program.topics.every(
      (topic) => quizInterests[topic]
    );

    const score = program.topics.reduce((acc, topic) => {
      return acc + (quizInterests[topic] ?? 0);
    }, 0);
```

```
  return {
    ...program,
    score,
    hasAllTopics,
  };
})
  .filter((program) => program.score)
  .sort((a, b) => {
    if (a.hasAllTopics !== b.hasAllTopics) {
      return a.hasAllTopics ? -1 : 1;
    }

    return b.score - a.score;
  });

console.log('abilitiesEducationPrograms', abilitiesEducationPrograms);

const suitableEducationPrograms = abilitiesEducationPrograms.filter(
  (program) =>
    !program.topics.some((topic) => dislikedInterests.includes(topic))
);

console.log('suitableEducationPrograms', suitableEducationPrograms);

await sendQuizResultMessage(
  t('quiz.suggestion_match', {
    programs:
      '\n' +
```

```
      suitableEducationPrograms
        .slice(0, 5)
        .map((program, index) => `${index + 1}. ${program.name}`)
        .join('\n'),
    })
);

const filteredAbilitiesEducationPrograms =
  abilitiesEducationPrograms.filter(
    (program) =>
      !suitableEducationPrograms.includes(program) && program.score > 2
  );

if (filteredAbilitiesEducationPrograms.length) {
  await sendQuizResultMessage(
    t('quiz.suggestion_extra', {
      programs:
        '\n' +
        filteredAbilitiesEducationPrograms
          .slice(0, 5)
          .map((program, index) => `${index + 1}. ${program.name}`)
          .join('\n'),
    })
  );
}

setInputVisible(true);
setQuizConversation([quizStartMessage]);
setQuizInterests({});
```

```
  setDislikedInterests([]);
};

const askNextQuizQuestion = async (interests?: InterestsRecord) => {
  const assistantAnswers = quizConversation.filter(
    (msg) => msg.role === 'assistant' && msg?.answers && msg.id
  );

  if (!interests) {
    interests = quizInterests;
  }

  if (assistantAnswers.length > 10) {
    showQuizResults();
    return;
  }

  const assistantAnswersIds = assistantAnswers.map((msg) => msg.id);

  const currentInterestsKeys = Object.keys(interests).filter(
    (key) => interests[key] > 0 && !dislikedInterests.includes(key)
  );

  const filteredQuizQuestions = QUIZ_QUESTIONS.filter(
    (q) =>
      !assistantAnswersIds.includes(q.question) &&
      (!q.requiredInterests ||
        q.requiredInterests.every((r) => currentInterestsKeys.includes(r)))
  );
```

```
const getInterestScore = (required?: string[]) => {
  if (!required || required.length === 0) return 0;
  return required.reduce((score, key) => score + (interests[key] || 0), 0);
};

let nextQuestion;

if (assistantAnswers.length > 2) {
  nextQuestion = filteredQuizQuestions
    .map((q) => ({
      ...q,
      score: getInterestScore(q.requiredInterests),
    }))
    .filter((q) => q.score > 0)
    .sort((a, b) => b.score - a.score)[0];
}

if (!nextQuestion) {
  nextQuestion = filteredQuizQuestions[0];
}

setQuizConversation((prev) => [
  ...prev,
  {
    id: nextQuestion.question,
    role: 'assistant',
    text: nextQuestion.question,
    answers: nextQuestion.answers,
```

```
      multiselect: nextQuestion.multiselect || false,
    },
  ]);

  scrollToBottom({ force: true, smooth: true, delayMs: 100 });
  setLoading(false);
  setInputVisible(true);
};

const selectQuizAnswer = async (
  selected: QuizAnswerType | QuizAnswerType[]
) => {
  setInputVisible(false);

  const answers = Array.isArray(selected) ? selected : [selected];

  setQuizConversation((prev) => [
    ...prev,
    {
      role: 'user',
      text: answers.map((r) => r.answer).join(', '),
      read: true,
    },
  ]);

  const newInterests = { ...quizInterests };

  const addCount: Record<string, number> = {};
  const removeCount: Record<string, number> = {};
```

```
answers.forEach(({ addInterests, removeInterests }) => {
  addInterests?.forEach((topic: string) => {
    addCount[topic] = (addCount[topic] || 0) + 1;
  });
  removeInterests?.forEach((topic: string) => {
    removeCount[topic] = (removeCount[topic] || 0) + 1;
  });
});

Object.entries(addCount).forEach(([topic, count]) => {
  newInterests[topic] = (newInterests[topic] || 0) + count;
});

Object.entries(removeCount).forEach(([topic, count]) => {
  newInterests[topic] = (newInterests[topic] || 0) - count;
  if (newInterests[topic] <= 0) {
    delete newInterests[topic];
  }
});

setQuizInterests(newInterests);

const newDislikedInterests = [
  ...dislikedInterests,
  ...answers.flatMap((a) => a.dislikeInterests ?? []).filter(Boolean),
];

setDislikedInterests(newDislikedInterests);
```

```
  const filteredInterests = { ...newInterests };

  newDislikedInterests.forEach((topic) => {
    if (filteredInterests[topic]) {
      delete filteredInterests[topic];
    }
  });

  setLoading(true);
  scrollToBottom({ force: true, smooth: true, delayMs: 100 });
  await new Promise((resolve) => setTimeout(resolve, 1000));
  scrollToBottom({ force: true, smooth: true, delayMs: 100 });
  askNextQuizQuestion(filteredInterests);
};

return (
  <ChatWidgetContext.Provider
    value={{
      open,
      handleSendMessage,
      loading,
      inputVisible,
      conversation,
      context,
      setContext,
      ref,
      scrollToBottom,
      closeConversation,
```

```
        setInputVisible,
        askChat,
        openQuiz,
        quizOpen,
        quizConversation,
        openChat,
        askNextQuizQuestion,
        selectQuizAnswer,
      }}
    >
      {children}
    </ChatWidgetContext.Provider>
  );
};

export const useChatWidget = () => {
  const ctx = useContext(ChatWidgetContext);
  if (!ctx)
    throw new Error('useChatWidget must be used within ChatWidgetProvider');
  return ctx;
};
```

Повний лістинг коду представленно у git репозиторії за посиланням: https://github.com/Belphin/knure-career-helper

ДОДАТОК В

Демонстраційний матеріал

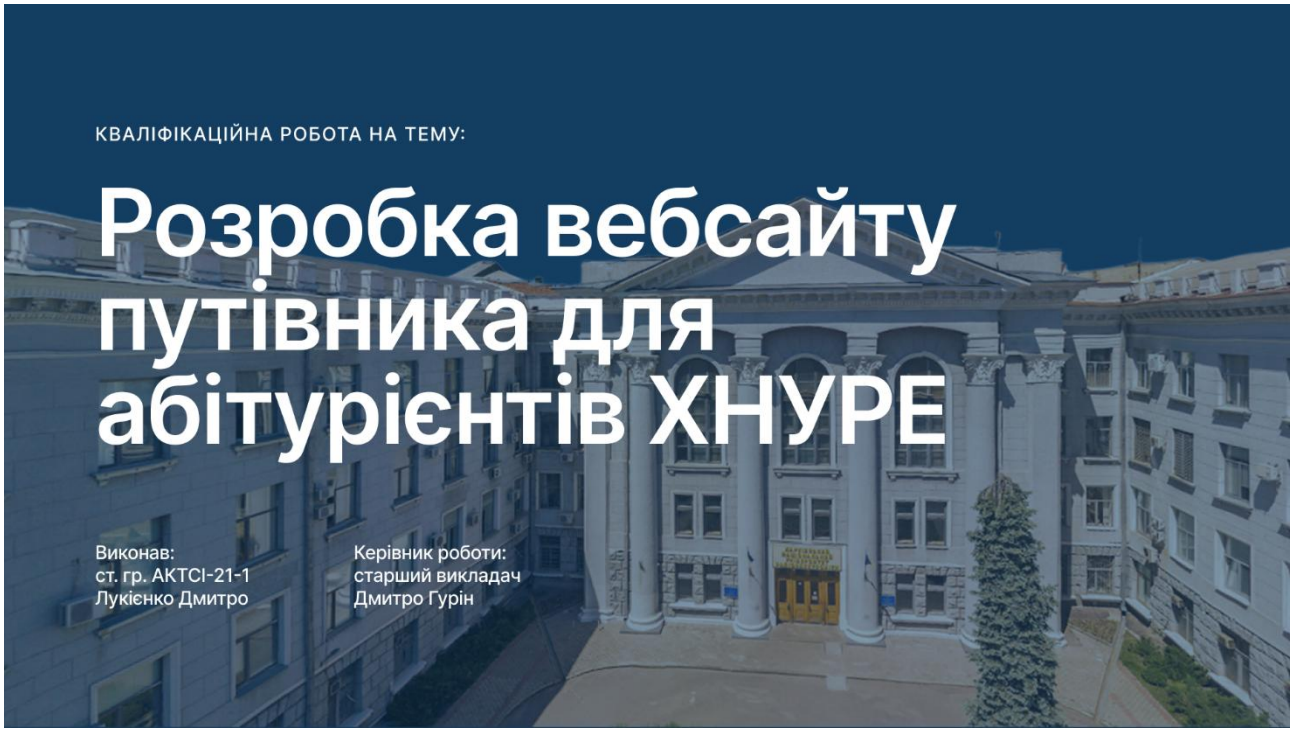

Рисунок В.1 – Титульний слайд

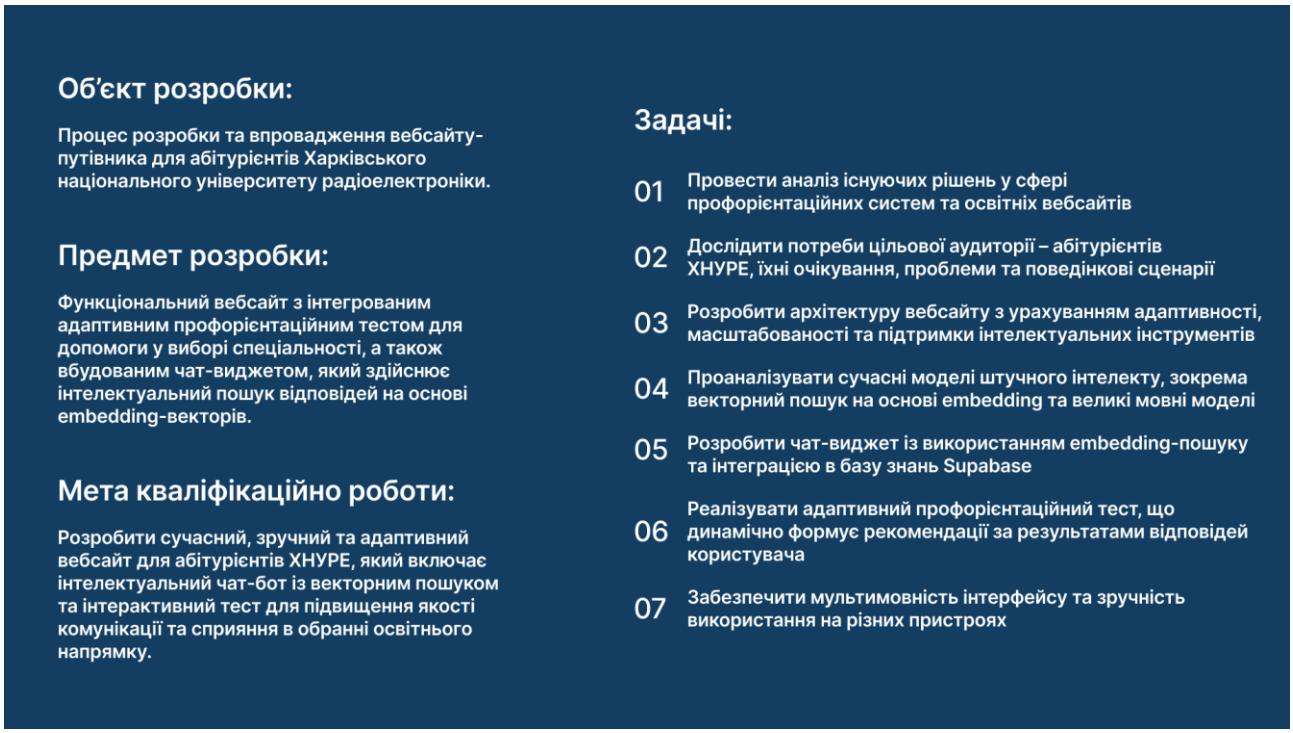

Рисунок В.2 – Задачі кваліфікаційної роботи

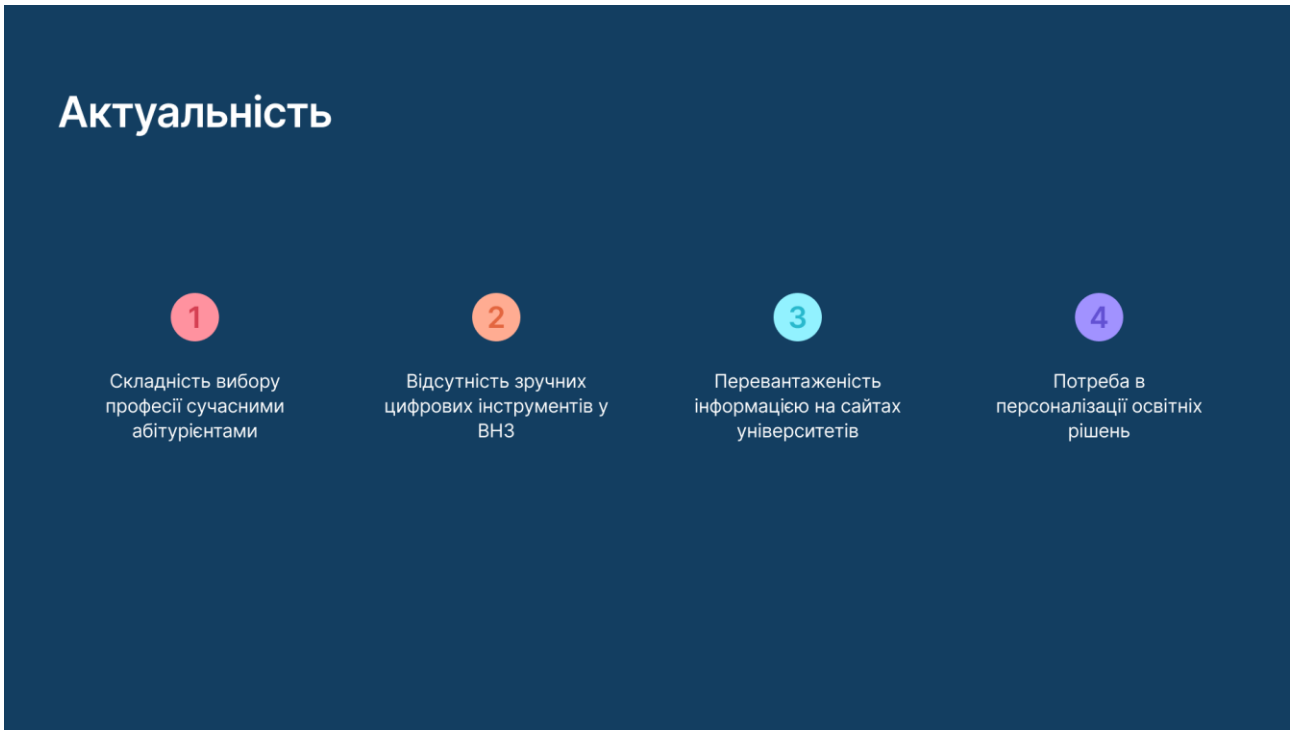

Рисунок В.3 – Актуальність кваліфікаційної роботи

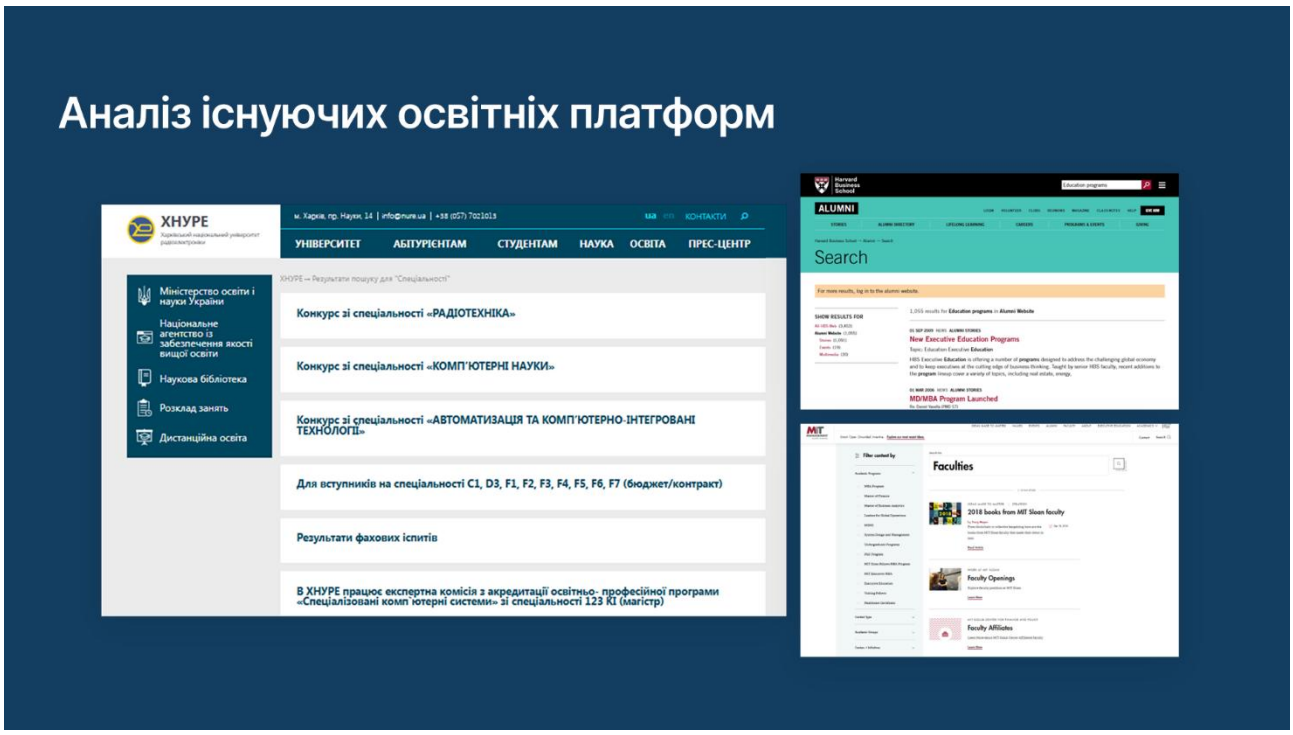

Рисунок В.4 – Аналіз існуючих освітніх платформ

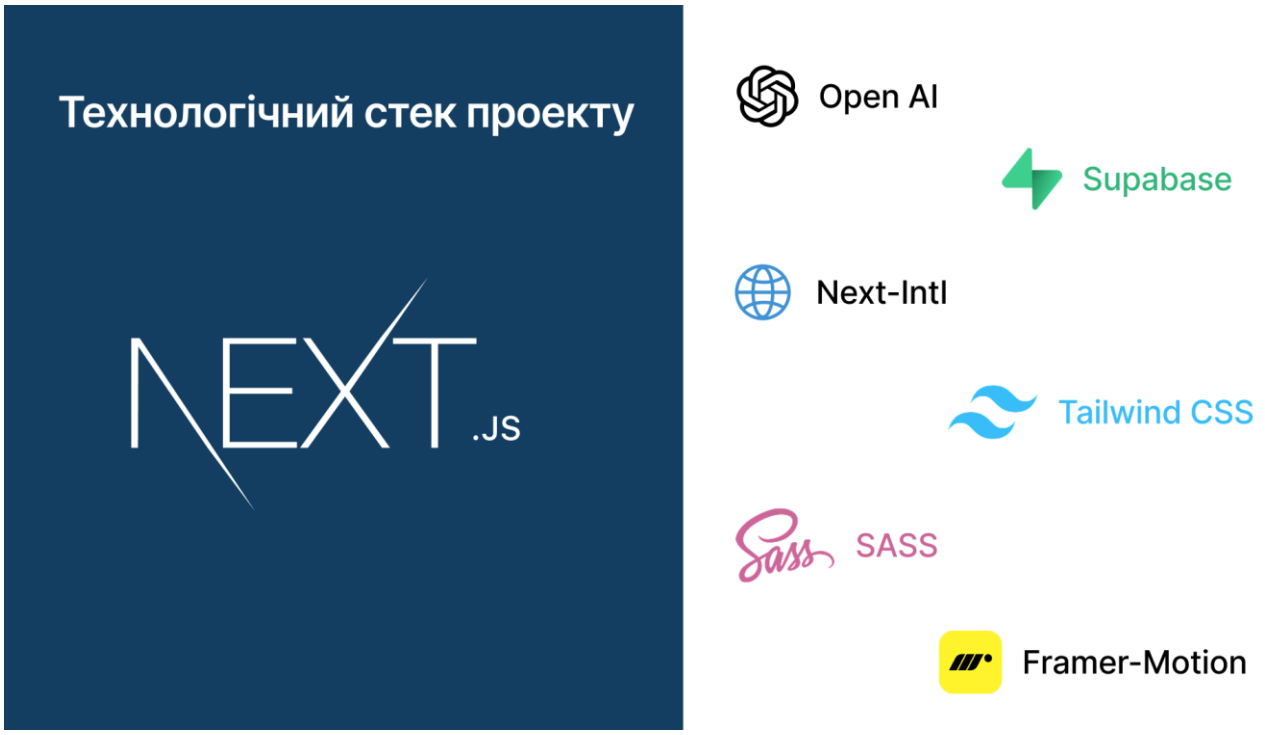

Рисунок В.5 – Технологічний стек проекту

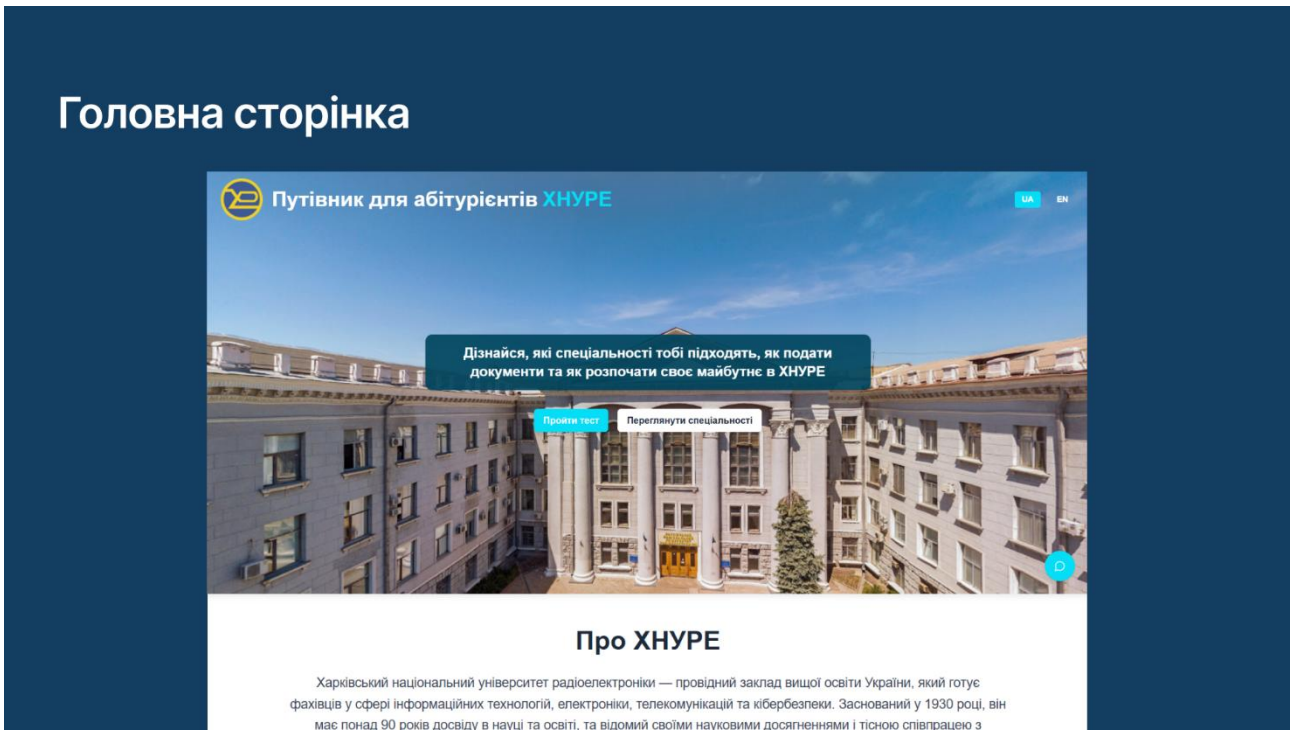

Рисунок В.6 – Головна сторінка

Рисунок В.7 – Вибір Embeddings

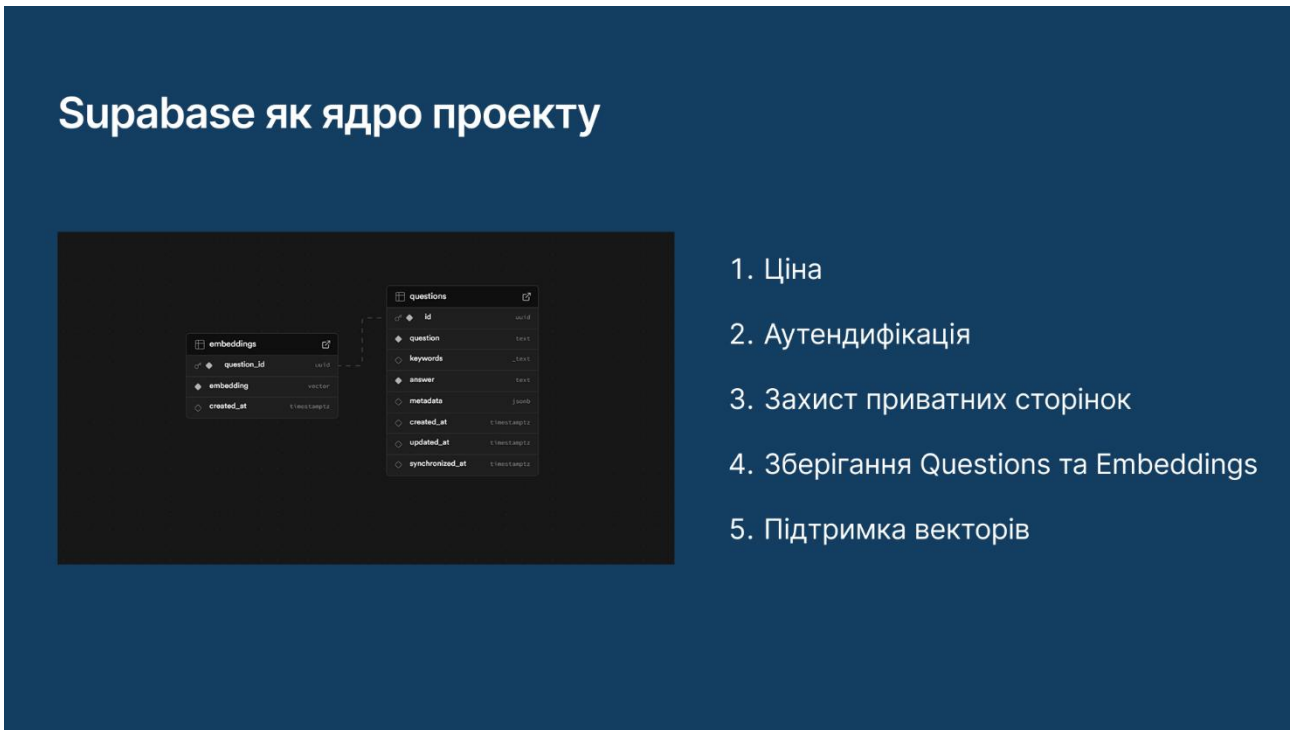

Рисунок В.8 – Вибір Supabase

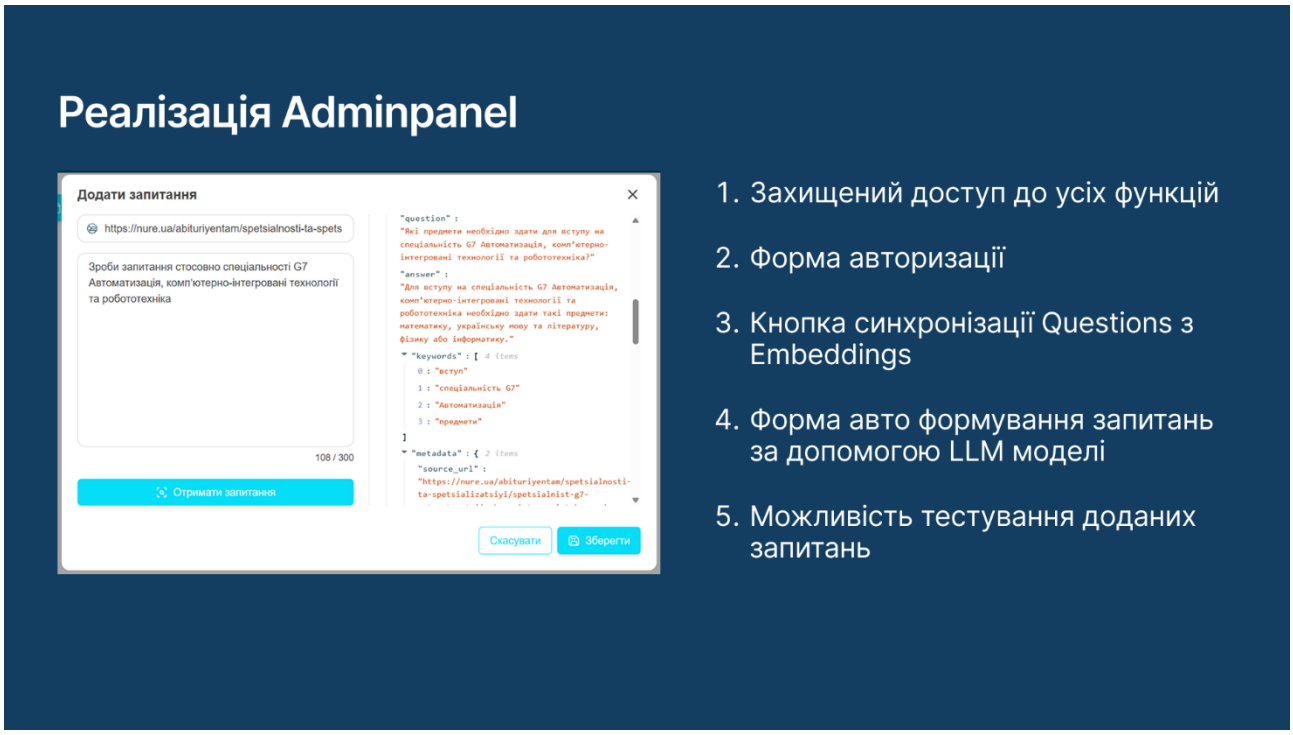

Рисунок В.9 – Реалізація Adminpanel

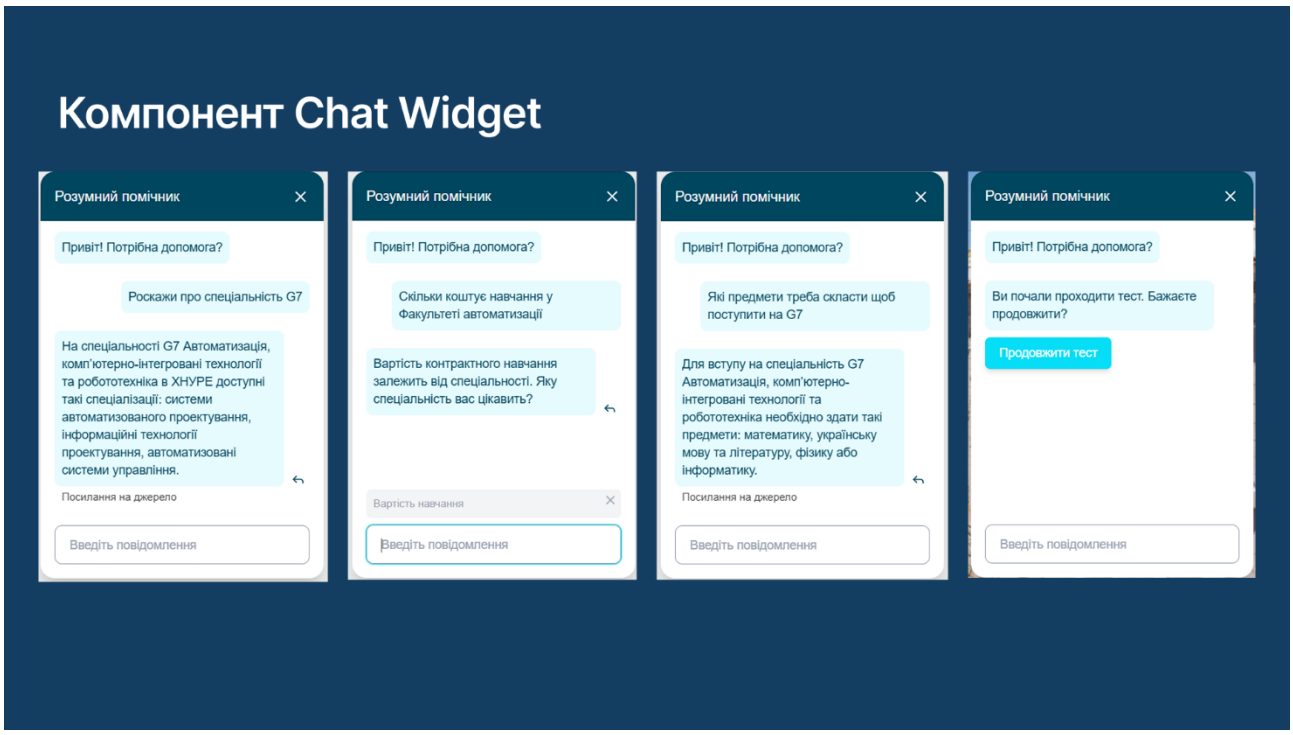

Рисунок В.10 – Компонент Chat Widget

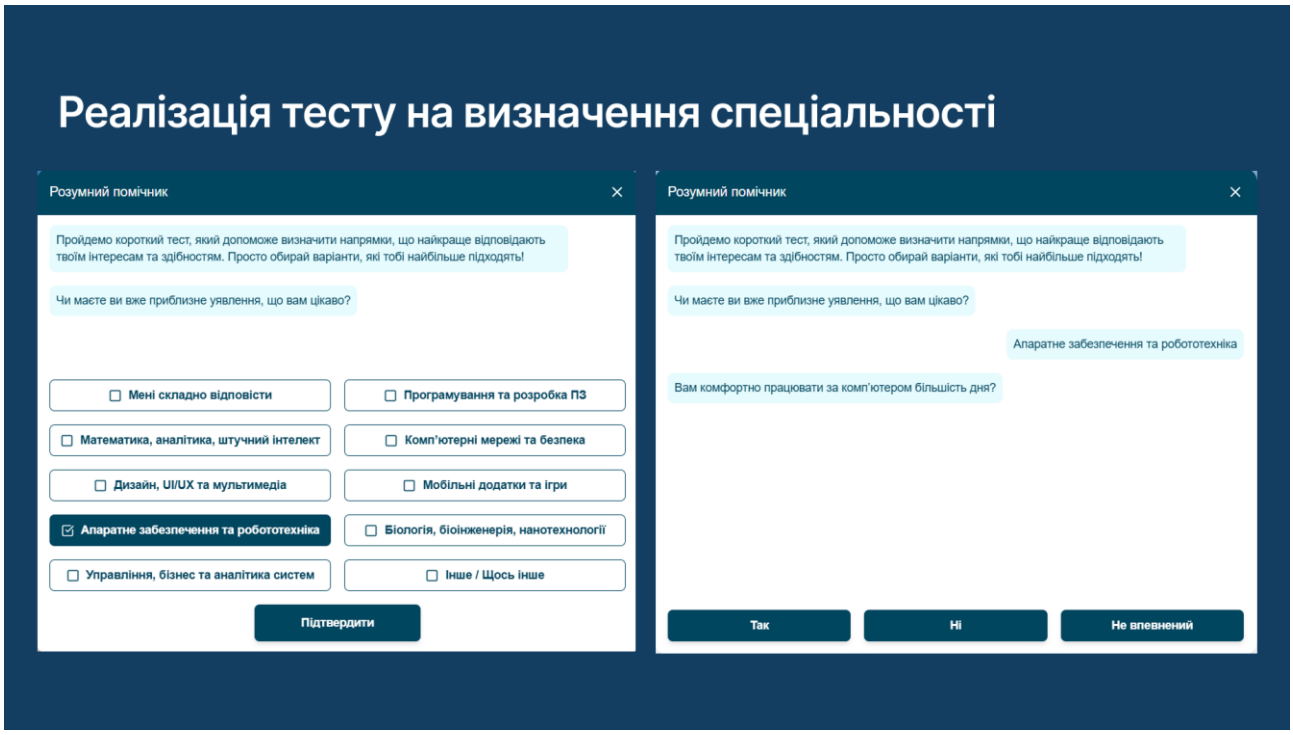

Рисунок В.11 – Компонент Chat Widget

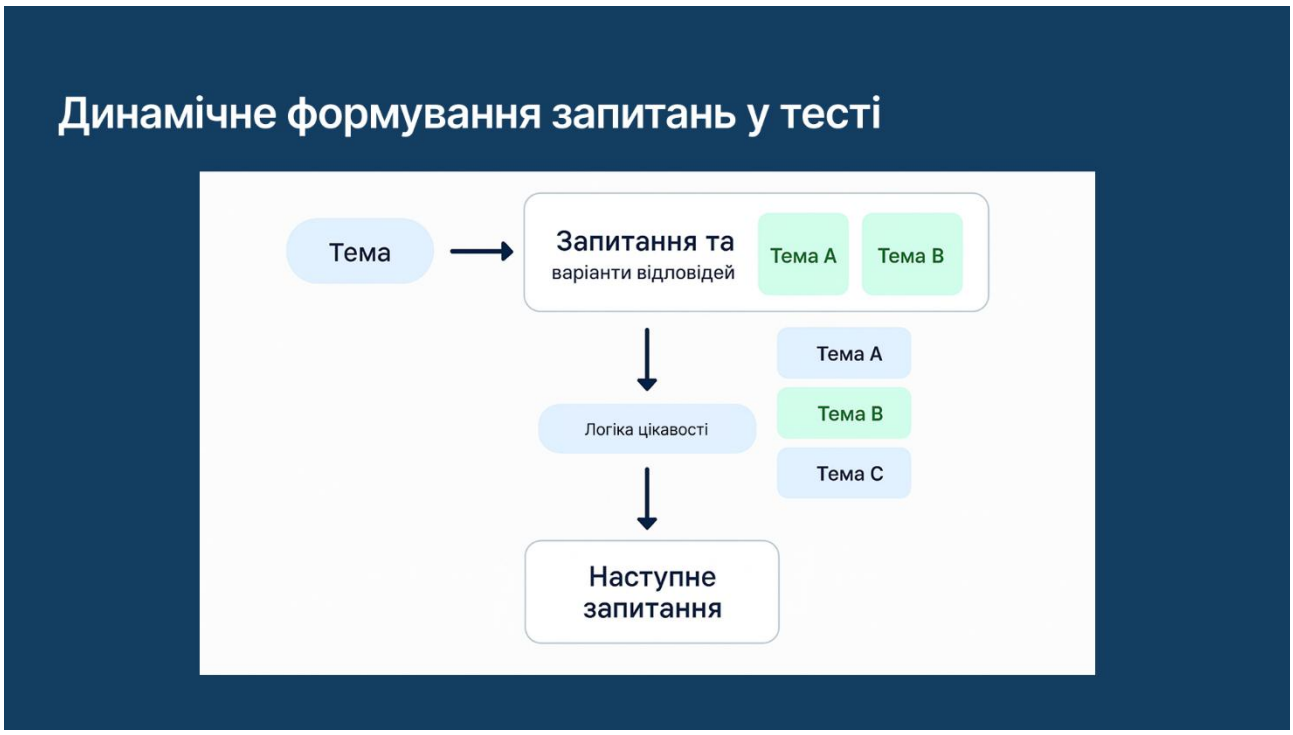

Рисунок В.12 – Динамічне формування запитань у тесті

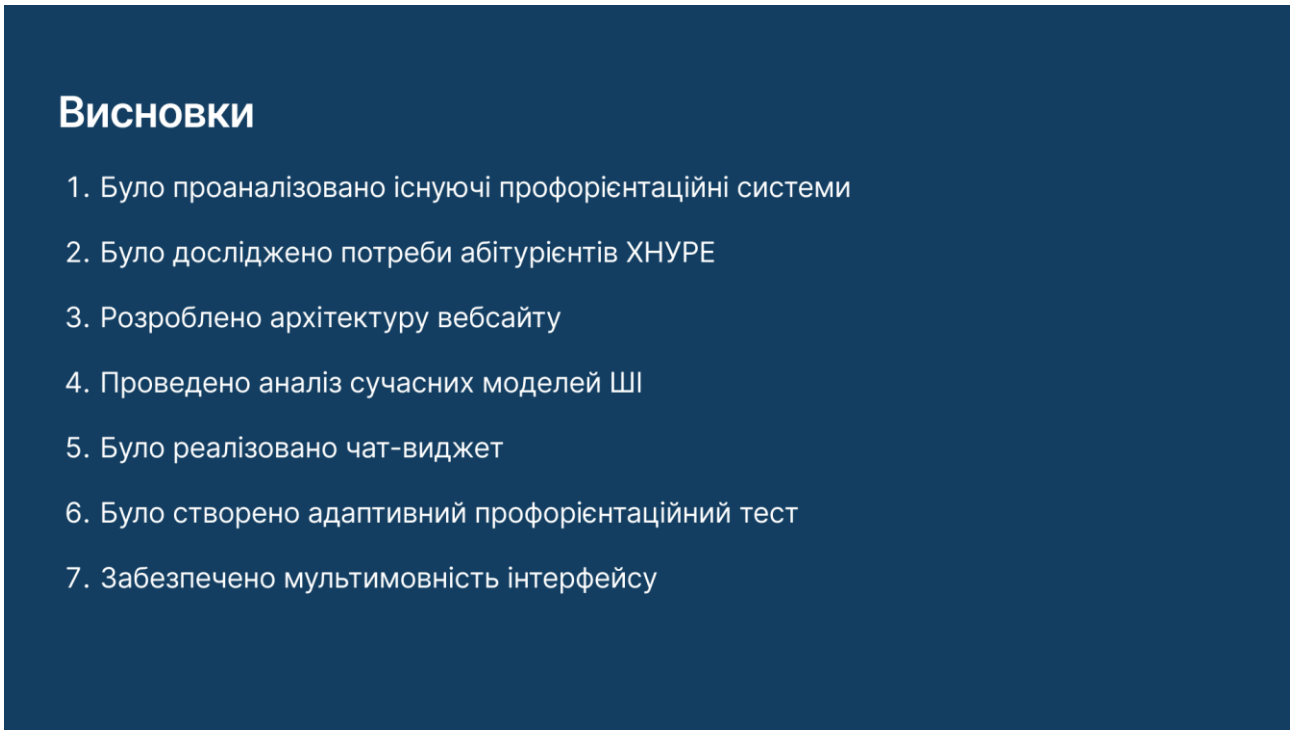

Рисунок В.13 – Висновки

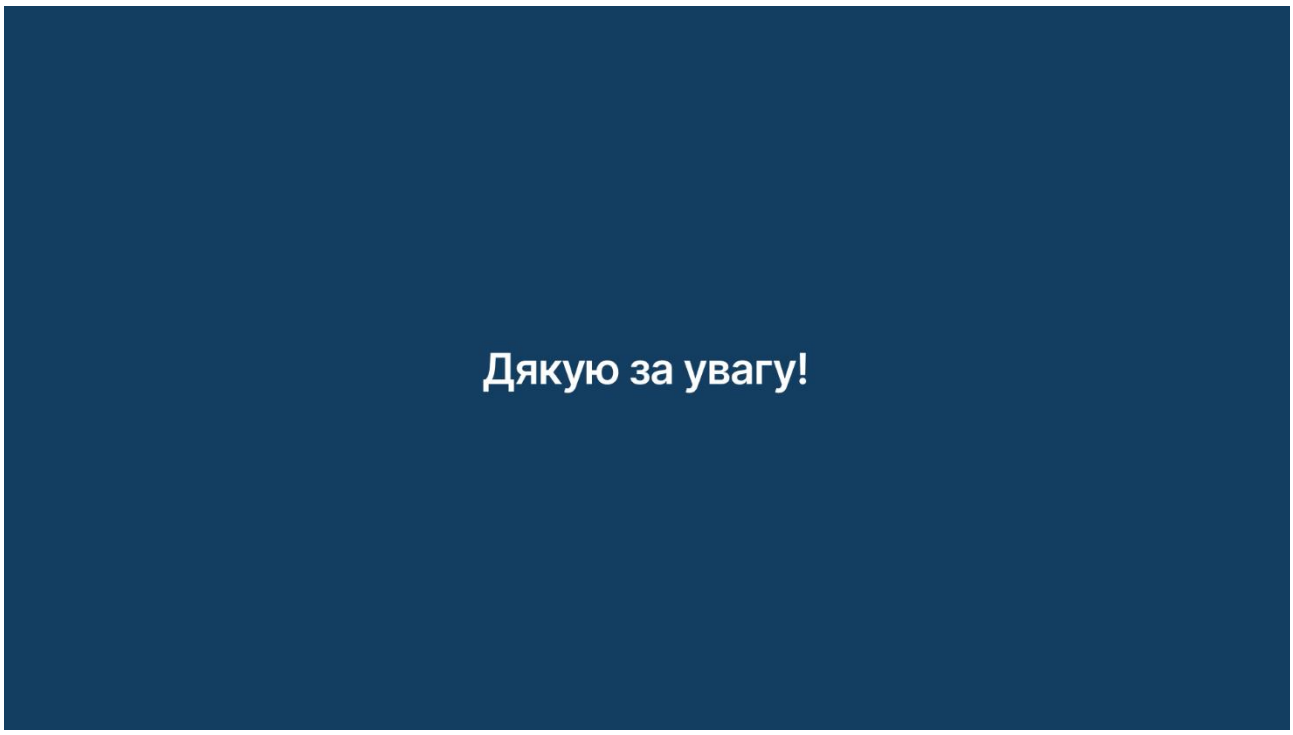

Рисунок В.14 – Фінальний слайд

№ документа	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК. 502900.005 ПЗ	Пояснювальна записка	А4, 58 с.
		Додаткові матеріали	
2		Апробіація	А4, 4 с.
3		Код проекту	А4, 20 с.
4		Демонстраційний матеріал	А4, 7 с.

					ГЮИК. 502900.005 ВД			
Змін.	Арк.	Номер докум.	Підпис	Дата				
Розроб.		Лукієнко Д. В.			Розробка вебсайту путівника для абітурієнтів ХНУРЕ	Літера	Аркуш	Аркушів
Перевір.		Гурін Д. В.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.контр.		Стародубцев М. Г.						
Затв.		Невлюдов І. Ш.						