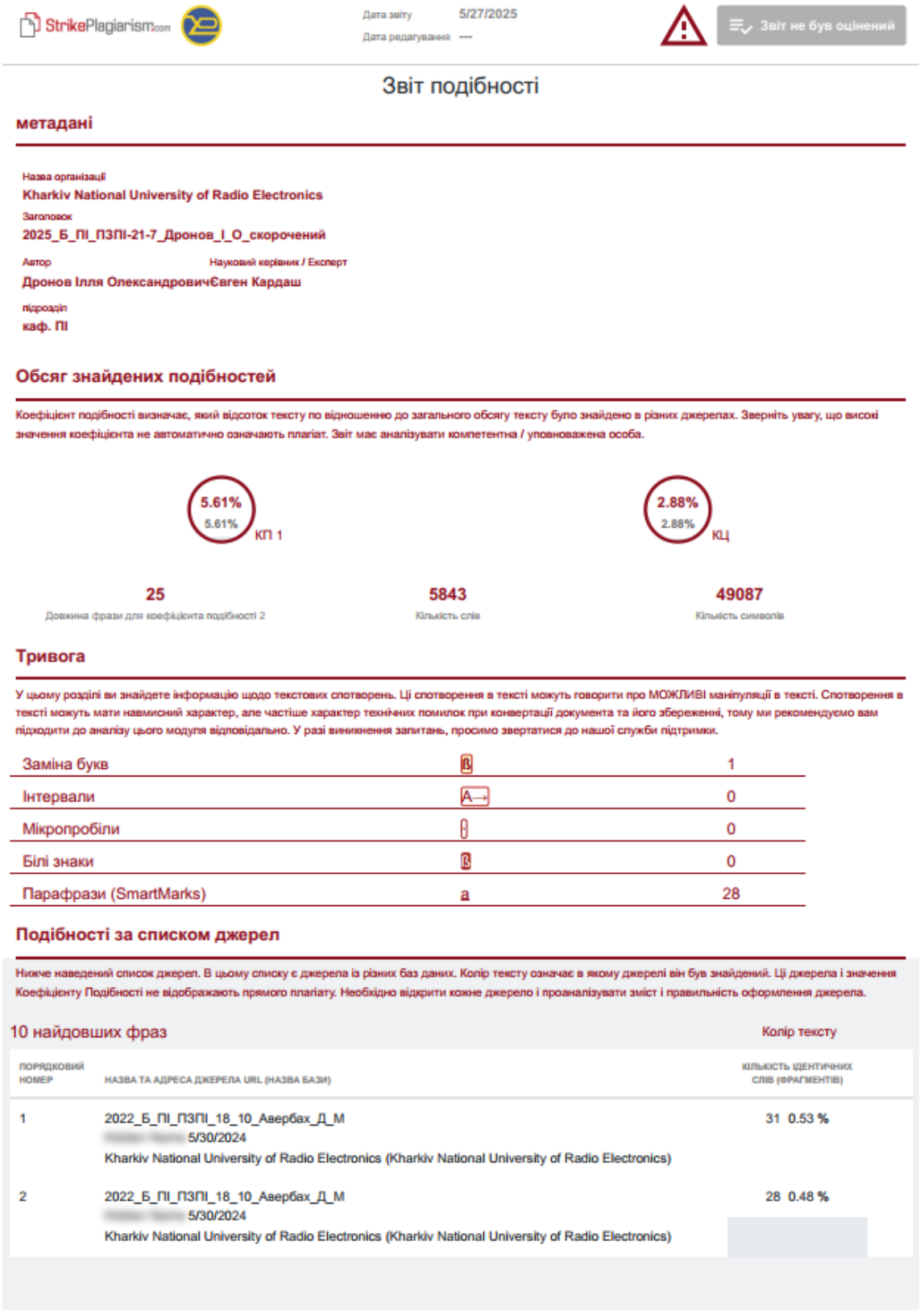
ДОДАТОК А

Звіт результатів перевірки на унікальність в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 5/27/2025

Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації

Kharkiv National University of Radio Electronics

Заголовок

2025_Б_ПІ_ПЗПІ-21-7_Дронов_І_О_скорочений

Автор **Дронов Ілля Олександрович**

Науковий керівник / Експерт **Євген Кардаш**

підрозділ

каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

5.61% 5.61% КП 1

2.88% 2.88% КЦ

25 Довжина фрази для коефіцієнта подібності 2

5843 Кількість слів

49087 Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	1
Інтервали	A⟷	0
Мікропробіли	¶	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	28

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	2022_Б_ПІ_ПЗПІ_18_10_Авербах_Д_М [redacted] 5/30/2024 Kharkiv National University of Radio Electronics (Kharkiv National University of Radio Electronics)	31 0.53 %
2	2022_Б_ПІ_ПЗПІ_18_10_Авербах_Д_М [redacted] 5/30/2024 Kharkiv National University of Radio Electronics (Kharkiv National University of Radio Electronics)	28 0.48 %

3	2022_Б_ПІ_ПЗПІ_18_10_Авербах_Д_М [redacted] 5/30/2024 Kharkiv National University of Radio Electronics (Kharkiv National University of Radio Electronics)	25 0.43 %
4	https://toxigon.com/implementing-authentication-in-fastapi	19 0.33 %
5	2022_Б_ПІ_ПЗПІ_18_10_Авербах_Д_М [redacted] 5/30/2024 Kharkiv National University of Radio Electronics (Kharkiv National University of Radio Electronics)	17 0.29 %
6	2022_Б_ПІ_ПЗПІ_18_10_Авербах_Д_М [redacted] 5/30/2024 Kharkiv National University of Radio Electronics (Kharkiv National University of Radio Electronics)	15 0.26 %
7	https://toxigon.com/implementing-authentication-in-fastapi	13 0.22 %
8	2022_Б_ПІ_ПЗПІ_18_10_Авербах_Д_М [redacted] 5/30/2024 Kharkiv National University of Radio Electronics (Kharkiv National University of Radio Electronics)	12 0.21 %
9	https://github.com/monoerea/medi-byte/blob/master/src/app/(components)/ControlPage.jsx	11 0.19 %
10	https://ela.kpi.ua/bitstreams/c2bb3b40-c48c-4254-b7fa-034a63baa8bd/download	11 0.19 %

з бази даних RefBooks (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з домашньої бази даних (2.98 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	2022_Б_ПІ_ПЗПІ_18_10_Авербах_Д_М [redacted] 5/30/2024 **Kharkiv National University of Radio Electronics (Kharkiv National University of Radio Electronics)**	**148 (8) 2.53 %**
2	2024_Б_ШІ_ІТШІ_20_5_Падалкін_К_Д_записка [redacted] 12/15/2024 Kharkiv National University of Radio Electronics (Kharkiv National University of Radio Electronics)	26 (4) 0.44 %

з програми обміну базами даних (0.82 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	bitstream_3a6ef3b2-57fc-4281-a7d4-b8190441f036 [redacted] 12/7/2024 National Technical University "Kharkiv Polytechnic Institute" students papers (National Technical University "Kharkiv Polytechnic Institute" students papers)	20 (2) 0.34 %
2	Програмне забезпечення генерації та обслуговування паролів [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	15 (2) 0.26 %
3	Кваліфікаційна робота_Салякін [redacted] 12/8/2024 Zaporizhzhia National University (Кафедра програмної інженерії)	13 (2) 0.22 %

з Інтернету (1.81 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

1	https://toxigon.com/implementing-authentication-in-fastapi	64 (6) 1.10 %
2	https://www.programcreek.com/typescript/?api=@headlessui/react.Dialog	14 (2) 0.24 %
3	https://github.com/monoerea/medi-byte/blob/master/src/app/(components)/ControlPage.jsx	11 (1) 0.19 %
4	https://ela.kpi.ua/bitstreams/c2bb3b40-c48c-4254-b7fa-034a63baa8bd/download	11 (1) 0.19 %
5	https://templespark.com/build-a-secure-fastapi-application-with-api-key-authentication-user-management-jinja2-templates-material-design-and-automated-testing-amazing-python-package-showcase-7/	6 (1) 0.10 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

ДОДАТОК Б

Код сторінки створення нового поста

```
'use client';

import { ActivityCategory, Post } from '@/app/types';
import { Button } from '@/components/ui/button';
import {
  Form,
  FormControl,
  FormField,
  FormItem,
  FormLabel,
  FormMessage,
} from '@/components/ui/form';
import { Input } from '@/components/ui/input';
import { MultiSelect } from '@/components/ui/multi-select';
import {
  Select,
  SelectContent,
  SelectItem,
  SelectTrigger,
  SelectValue,
} from '@/components/ui/select';
import { Textarea } from '@/components/ui/textarea';
import { toast } from '@/hooks/useToast';
import { axiosInstance } from '@/lib/services/axiosConfig';
import { zodResolver } from '@hookform/resolvers/zod';
import { AxiosError } from 'axios';
import { useSession } from 'next-auth/react';
import { useParams, useRouter } from 'next/navigation';
import { useState } from 'react';
import { useForm } from 'react-hook-form';
import * as z from 'zod';

interface PostFormProps {
  activityCategories: ActivityCategory[];
  post?: Post;
}

const formSchema = z.object({
  title: z.string().min(1, 'Title is required'),
  description: z.string().min(1, 'Description is required'),
  service_price: z.preprocess(
    (val) => (val === '' ? undefined : Number(val)),
    z.number({ invalid_type_error: 'Price must be a number'
}).positive('Price must be positive'),
  ),
  service_type: z.enum(['S', 'P']),
  category_ids: z.array(z.string()).min(1, 'At least one category is
required'),
});
type FormValues = z.input<typeof formSchema>;

const serviceTypeOptions = [
  { value: 'S', label: 'Seeking Mentor' },
  { value: 'P', label: 'Offering Mentorship' },
```

```
];

export default function PostForm({ activityCategories, post }:
PostFormProps) {
  const { data: session } = useSession();
  const lng = useParams().lng;
  const router = useRouter();
  const [isLoading, setIsLoading] = useState(false);

  const form = useForm<FormValues>({
    resolver: zodResolver(formSchema),
    defaultValues: {
      title: post?.title || '',
      description: post?.description || '',
      service_price: post?.service_price ?? 0,
      service_type: post?.service_type || 'S',
      category_ids: post?.categories.map((category) => category.id) || [],
    },
  });
  const onSubmit = async (values: FormValues) => {
    if (!session?.accessToken) return;

    setIsLoading(true);
    try {
      if (post) {
        await axiosInstance.put(
          `/posts/${post.id}`,
          {
            ...values,
            service_price: Number(values.service_price),
          },
          {
            headers: {
              Authorization: `Bearer ${session.accessToken}`,
            },
          },
        );

        toast({
          title: 'Success',
          description: 'Post updated successfully',
        });
      } else {
        await axiosInstance.post(
          '/posts/',
          {
            ...values,
            service_price: Number(values.service_price),
          },
          {
            headers: {
              Authorization: `Bearer ${session.accessToken}`,
            },
          },
        );

        toast({
          title: 'Success',
```

```
          description: 'Post created successfully',
        });
      }

      router.push(`/${lng}/my-posts`);
    } catch (error) {
      if (error instanceof AxiosError) {
        toast({
          title: 'Error',
          description: error.response?.data.message,
          variant: 'destructive',
        });
      } else {
        toast({
          title: 'Error',
          description: post ? 'Failed to update post' : 'Failed to create
post',
          variant: 'destructive',
        });
      }
    } finally {
      setIsLoading(false);
    }
  };

  return (
    <Form {...form}>
      <form onSubmit={form.handleSubmit(onSubmit)} className="space-y-6">
        <FormField
          control={form.control}
          name="title"
          render={({ field }) => (
            <FormItem>
              <FormLabel>Title</FormLabel>
              <FormControl>
                <Input placeholder="Enter post title" {...field} />
              </FormControl>
              <FormMessage />
            </FormItem>
          )}
        />

        <FormField
          control={form.control}
          name="description"
          render={({ field }) => (
            <FormItem>
              <FormLabel>Description</FormLabel>
              <FormControl>
                <Textarea
                  placeholder="Enter post description"
                  className="min-h-[100px]"
                  {...field}
                />
              </FormControl>
              <FormMessage />
            </FormItem>
          )}
```

```
        />

        <FormField
          control={form.control}
          name="service_price"
          render={({ field }) => (
            <FormItem>
              <FormLabel>Price</FormLabel>
              <FormControl>
                <Input
                  type="number"
                  placeholder="Enter price"
                  {...field}
                  value={field.value?.toString() || ''}
                />
              </FormControl>
              <FormMessage />
            </FormItem>
          )}
        />

        <FormField
          control={form.control}
          name="service_type"
          render={({ field }) => (
            <FormItem>
              <FormLabel>Service Type</FormLabel>
              <Select onValueChange={field.onChange}
defaultValue={field.value}>
                <FormControl>
                  <SelectTrigger>
                    <SelectValue placeholder="Select service type" />
                  </SelectTrigger>
                </FormControl>
                <SelectContent className="bg-white">
                  {serviceTypeOptions.map((option) => (
                    <SelectItem key={option.value} value={option.value}>
                      {option.label}
                    </SelectItem>
                  ))}
                </SelectContent>
              </Select>
              <FormMessage />
            </FormItem>
          )}
        />

        <FormField
          control={form.control}
          name="category_ids"
          render={({ field }) => (
            <FormItem>
              <FormLabel>Categories</FormLabel>
              <FormControl>
                <MultiSelect
                  options={activityCategories.map((category) => ({
                    label: category.title,
                    value: category.id,
```

```
                  }))}
                  value={field.value}
                  onChange={field.onChange}
                  placeholder="Select categories"
                  variant="inverted"
                  animation={2}
                  maxCount={3}
                />
              </FormControl>
              <FormMessage />
            </FormItem>
          )}
        />

        <div className="flex justify-end gap-4">
          <Button
            type="button"
            variant="secondary"
            className="bg-gray-200 hover:bg-gray-300 text-gray-800 font-
semibold"
            onClick={() => router.push(`/${lng}/my-posts`)}>
            Cancel
          </Button>
          <Button
            type="submit"
            disabled={isLoading}
            className="bg-blue-brand hover:bg-blue-brand/90 text-white
font-semibold">
            {isLoading ? 'Saving...' : post ? 'Update Post' : 'Create
Post'}
          </Button>
        </div>
      </form>
    </Form>
  );
}
```

ДОДАТОК В

Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Веб-платформа для надання та отримання послуг

Дронов Ілля Олександрович ПЗПІ-21-7

Керівник: доц. Русакова Н.Є.

10 червня 2025

Мета роботи

Створення зручної, безпечної та масштабованої веб-платформи, яка надає змогу користувачам як продавати свої знання та навички, так і отримувати нові. Один користувач може бути одночасно як ментором, так і учнем. Проєкт орієнтований на підвищення доступності якісних освітніх та консультаційних послуг.

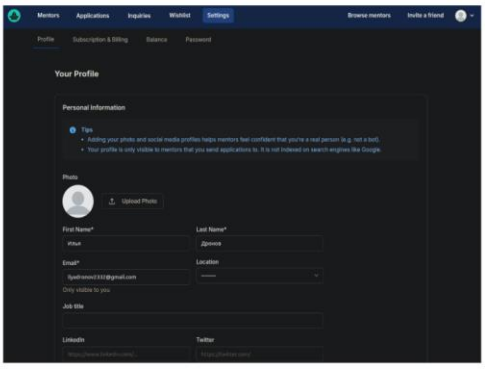
Попит на індивідуальне менторство стрімко зростає — люди шукають швидкі й практичні способи отримання знань, персональної підтримки та розвитку кар'єри. У таких умовах виникає необхідність у платформі, яка б забезпечувала повноцінну взаємодію між ментором та учнем: від реєстрації до проведення сесій та здійснення оплати.

Аналіз існуючих рішень

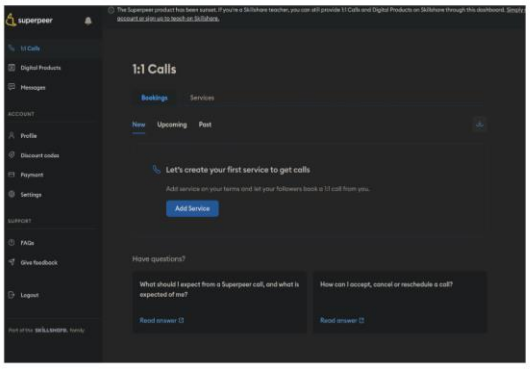
Сучасні аналоги частково покривають потреби, проте мають обмеження:

- **MentorCruise**
  + широкий вибір менторів з технічних сфер
  – обмежена функціональність для учнів (відсутність внутрішнього чату, обмежені типи сесій)
- **Superpeer**
  + інтеграція з відеозв'язком та календарем
  – орієнтація більше на одноразові консультації, ніж на тривале менторство

Користувацькі інтерфейси аналогів



MentorCruise



Superpeer

Постановка задачі та опис системи

- виконати проектування архітектури веб-додатку;
- здійснити вибір засобів реалізації завдання;
- організувати структуру бази даних;
- реалізувати серверну та клієнтську частини веб-додатку;
- організувати пошук, сортування, фільтрацію даних;
- провести функціональне та нефункціональне тестування веб-додатку.

Вибір технологій розробки

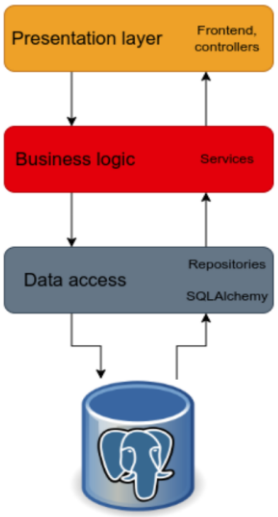







Архітектура створенного програмного забезпечення

- **Presentation layer** — обробляє запити користувачів, взаємодіє з фронтендом та контролерами.
- **Business logic** — реалізує основні правила системи, обробляє запити через сервіси.
- **Data access layer** — відповідає за взаємодію з базою даних через репозиторії та ORM (SQLAlchemy).
- **Database** — PostgreSQL використовується для надійного зберігання структурованих даних.



Переваги архітектури:

- спрощення супроводу та масштабування;
- легкість в тестуванні кожного шару окремо;
- висока гнучкість при розширенні функціоналу.

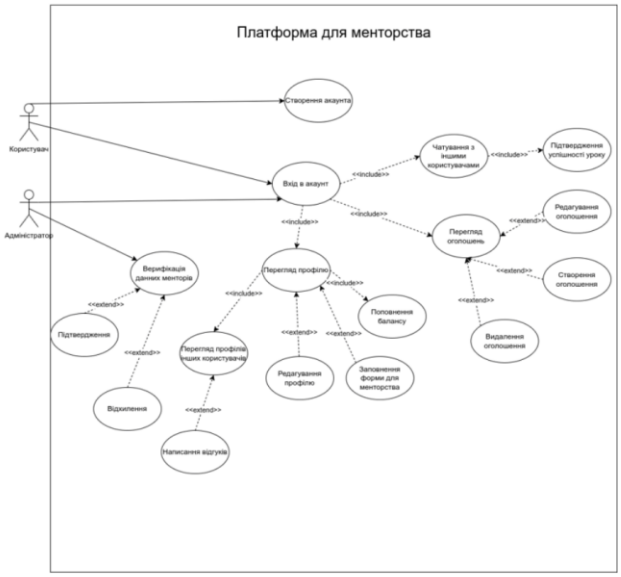
Діаграма прецедентів

На діаграмі зображено основні сценарії взаємодії користувачів із платформою.
Користувач може:

- зареєструватися та увійти в систему;
- переглядати та редагувати профіль;
- заповнювати менторську анкету;
- створювати, редагувати та видаляти оголошення;
- спілкуватися з іншими користувачами;
- підтверджувати успішність уроку;
- поповнювати баланс.

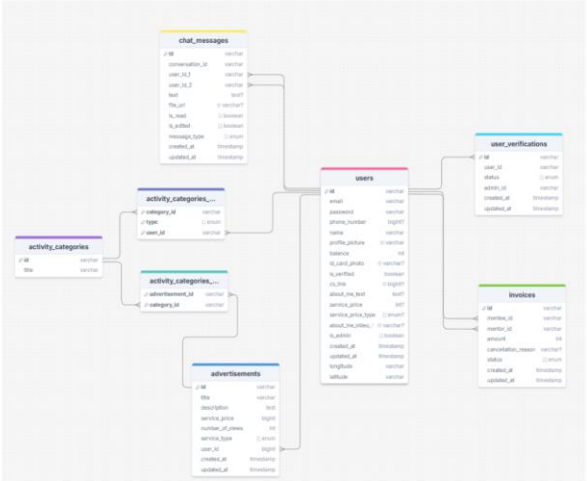
Адміністратор має розширені можливості:

- верифікація менторів (підтвердження/відхилення);

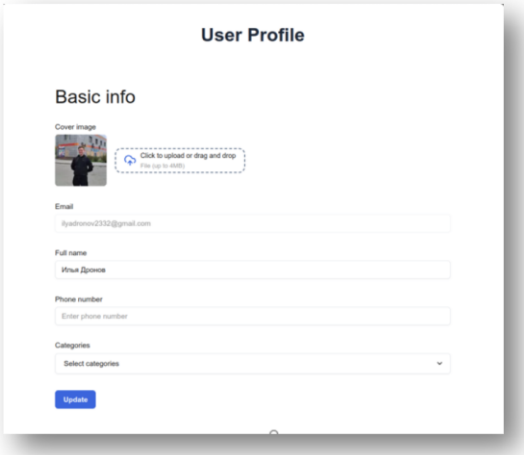


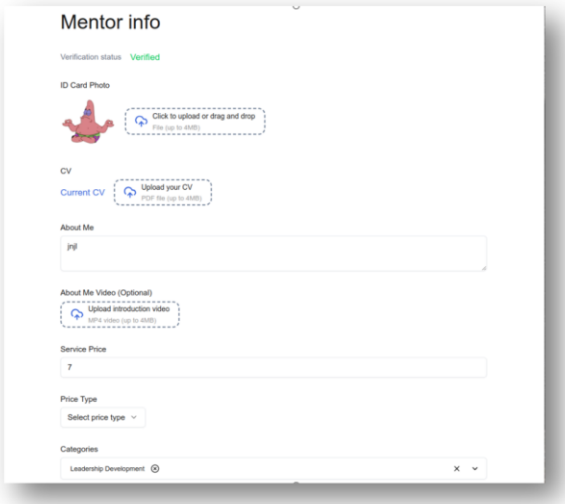
Схеми бази даних

Схема складається з 8 таблиць, які відповідають за верифікацію користувачів, спілкування між користувачами в чаті, інформацію про оголошення, зберігання інформації про інвойси



Інтерфейс профілю користувача





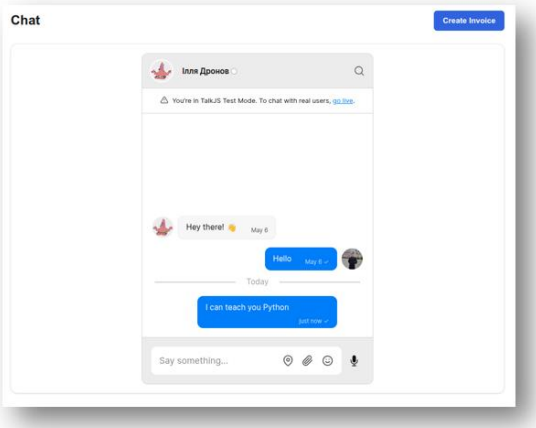
Приклад реалізації оновлення профілю

На рисунку присутня така логіка:

- Валідація користувача, що відправив запит.
- Обробка та конвертація одного типу даних в інший
- Створення дочірніх сутностей (категорій)
- Завантаження наявних картинок в Amazon S3
- Повернення оновлених даних

```python
try:
    logger.info(f'Updating user profile of the user "{current_user}"')

    if current_user.id != user_id:
        raise HTTPException(status.HTTP_403_FORBIDDEN, detail="Forbidden")

    if data.activity_categories:
        activity_categories_ids = json.loads(data.activity_categories[0])
        await self.user_repository.clean_activity_categories(current_user.id)

        new_activity_categories = [
            ActivityCategoryUser(
                user_id=current_user.id, category_id=category_id
            )
            for category_id in activity_categories_ids
        ]

        await self.user_repository.save_many(new_activity_categories)
        data.activity_categories = None

    # Upload files to S3
    upload_tasks = [
        ("profile_picture", "profile_picture.jpg"),
        ("id_card_photo", "id_card_photo.jpg"),
        ("about_me_video_link", "about_me_video_link.mp4"),
        ("cv_link", "cv_link.pdf"),
    ]

    await self._upload_files_to_s3(data, upload_tasks)

    await self.user_repository.update_user(current_user.id, data)
    updated_user = await self.user_repository.get_user_by_id(current_user.id)
```


Інтерфейс чату користувачів

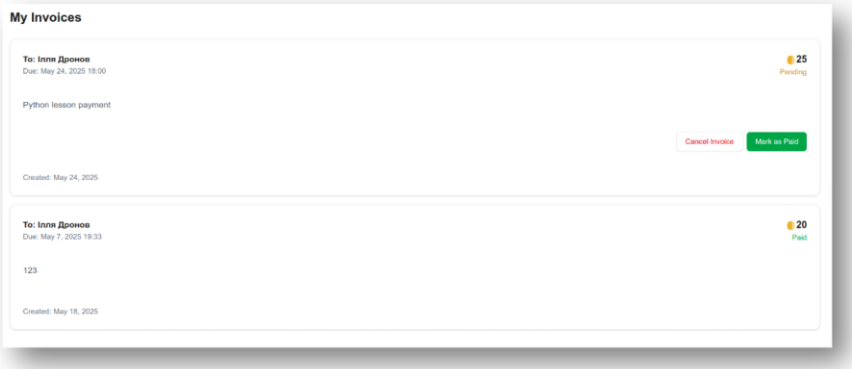


Приклад реалізації чату

Основна бізнес логіка знаходиться на стороні клієнта, де відбуватеься запит до 3rd-party (стороннього) сервіса, який надає зручний API та UI для швидкого впровадження функціоналу чатування в своєму проекті

```
useEffect(() => {
  const fetchConversationData = async () => {
    const response = await axios.get(
      `https://api.talkjs.com/v1/${process.env.NEXT_PUBLIC_TALKJS_APP_ID}/conversations/${id}`,
      {
        headers: {
          Authorization: `Bearer ${process.env.NEXT_PUBLIC_TALKJS_SECRET_KEY}`,
        },
      },
    );
    setConversationData(response.data);
  };
  fetchConversationData();
}, [id]);
```


Інтерфейс списку інфойсів ментора



Приклад реалізації створення інвойса

На рисунку присутня така логіка:

- Перевірка існування користувачів за наданими в запиті ID
- Валідація того, що учень має достатній баланс для проведення операції
- Зняття потрібної суми з балансу учня та передача її до сутності інвойсу.
- Створення інвойсу

```python
async def create_invoice(self, invoice_data: InvoiceCreate) -> LessonInvoice:
    mentor_user = await self.user_repository.get_user_by_id(invoice_data.mentor_id)
    if not mentor_user:
        raise HTTPException(status_code=404, detail="Mentor not found")

    mentee_user = await self.user_repository.get_user_by_id(invoice_data.mentee_id)
    if not mentee_user:
        raise HTTPException(status_code=404, detail="Mentee not found")

    if mentee_user.balance < invoice_data.amount:
        raise HTTPException(
            status_code=400, detail="Mentee does not have enough balance"
        )

    mentee_user.balance -= invoice_data.amount
    await self.user_repository.save(mentee_user)

    await self.invoice_repository.create_invoice(invoice_data)
```

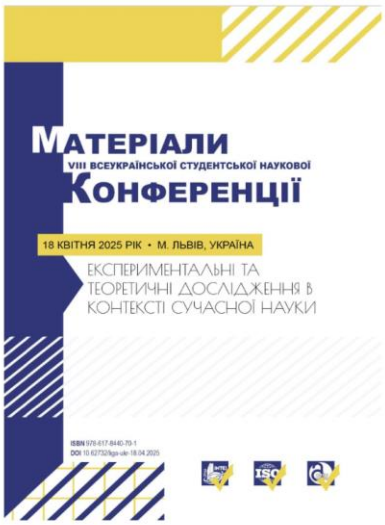

Тестування

У процесі розробки платформи було проведено декілька рівнів тестування для перевірки стабільності та коректності роботи функціональності:

- **Інтеграційне тестування бекенду** — реалізовано з використанням **pytest**. Покриває ключові API-ендпоінти: авторизація, реєстрація, створення постів, виставлення інвойсів.
- Системне тестування, на базі випадків використання

```
============================= test session starts ==============================
platform linux -- Python 3.10.12, pytest-8.2.0, pluggy-1.5.0 -- /usr/bin/python3
cachedir: .pytest_cache
rootdir: /home/user/Desktop
plugins: anyio-4.9.0, asyncio-0.26.0
asyncio: mode=strict, asyncio_default_fixture_loop_scope=None, asyncio_default_test_loop_scope=function
collected 3 items

test-bach.py::test_login_user PASSED
test-bach.py::test_create_post PASSED
test-bach.py::test_get_posts PASSED

============================== 3 passed in 0.73s ===============================
```

Апробація результатів



18 квітня 2025 рік | Львів, Україна | Міжнародна наукова ліга

Дропов Ілля Олександрович, здобувач вищої освіти факультету комп'ютерних наук *«Харківський національний університет радіоелектроніки». Україна*

Науковий керівник: Русакова Наталія Євгенівна, доцент кафедри Програмної інженерії *«Харківський національний університет радіоелектроніки». Україна*

ПРОГРАМНА СИСТЕМА ДЛЯ НАДАННЯ ТА ОТРИМАННЯ НАВЧАЛЬНИХ ПОСЛУГ

Сучасні менторські платформи потребують розумних рішень для управління користувачами, сесіями та комунікацією [1]. В цій роботі представлено систему «Knowlify» – спрощену платформу, створену спеціально для підтримки індивідуальних менторів, які пропонують свої послуги онлайн. Вона спрощує планування сесій, управління профілями та безпечну взаємодію між менторами та учнями. Система передбачає рольовий доступ: ментори можуть керувати доступністю, цінами та повідомленнями, а учні можуть знаходити менторів, бронювати сесії та залишати відгуки. Створена на основі сучасних веб-технологій - Next.js, TailwindCSS та Shadcn - вона пропонує швидкий, чистий та чуйний користувацький інтерфейс [2]. Безпечна автентифікація здійснюється за допомогою JWT та Google Sign-in, а основна логіка працює на основі FastAPI. «Knowlify» дає можливість індивідуальним менторам ефективно управляти своїми послугами та надавати високоякісні рекомендації своїм підопічних.

На сучасному ринку існує кілька платформ, орієнтованих на освітні сервіси та менторство. Серед них — Clarity.fm [3], сервіс для експертних консультацій, що підходить переважно для бізнес-напрямів, але має обмежені можливості персоналізації, а також MentorCruise [4], що пропонує розширену систему пошуку менторів, проте має менш гнучкий функціонал для індивідуального брендингу та кастомізації (рис. 1).

Рис. 1. **Користувацький інтерфейс аналогів (Clarity.fm .ліворуч, MentorCruise праворуч)**

Запропонована в роботі система "Knowlify" вирізняється високим рівнем автоматизації процесів, інтуїтивно зрозумілим інтерфейсом і легкою інтеграцією з сучасними сервісами, такими як Google-автентифікація та Stripe. Завдяки цим перевагам система сприяє підвищенню якості надання менторських послуг та забезпечує ефективну взаємодію між користувачами. У цій роботі представлено опис

програмної системи "Knowlify", розробленої для підтримки індивідуальних менторів і їхніх клієнтів, з чітким розподілом ролей та функціональним інтерфейсом.

У системі "Knowlify" основну функціональність реалізують ролі Ментор та Учень. Кожен користувач може виступати в обох ролях залежно від заповнених профільних даних: наявності резюме, відео-презентації, опису досвіду та встановленої вартості послуг. Ментори можуть створювати оголошення про свої послуги, обираючи категорії діяльності, тип консультації та ціну. Учні мають змогу переглядати ці оголошення, фільтрувати за напрямами та напряму взаємодіяти з менторами через вбудовану систему чатів. Комунікація реалізована за допомогою таблиць "chat_conversations" і "chat_messages", які забезпечують зберігання історії діалогів, повідомлень, файлів та статусів прочитання.

Роль Адміністратора в системі не є основною, але виконує важливі модераційні функції: перевірка профілів менторів, контроль якості контенту, розгляд скарг та управління загальною безпекою платформи.

На відміну від традиційних освітніх платформ, "Knowlify" вирізняється гнучкою архітектурою та підтримкою індивідуальних менторів. Інтерфейс, побудований з використанням Next.js, TailwindCSS та Shadcn, забезпечує простоту, адаптивність і зручність користування. Система використовує JWT та Google Sign-In для автентифікації, а основна бізнес-логіка реалізована через FastAPI. Автоматизація взаємодії між ментором і учнем, а також обробка транзакцій сприяють ефективності та зручності користування платформою.

На рисунку 2 представлено модель реляційної бази даних, реалізовану в PostgreSQL. Відношення "users" зберігає дані про користувачів, включаючи статус верифікації, координати, фінансову інформацію та профільні атрибути. Створені менторами оголошення зберігаються у таблиці "advertisement" та пов'язуються з категоріями через таблицю-зв'язку.

Рис. 2. **Схема реляційної бази даних до програмної системи**

Підсумки

У результаті виконання кваліфікаційної роботи було розроблено веб-платформу, яка забезпечує зручну взаємодію між менторами та учнями.
Проведено повний цикл проєктування: від аналізу аналогів та постановки вимог — до реалізації серверної та клієнтської частин з урахуванням сучасних технологій.
Реалізовано функціонал:

- реєстрація та авторизація з Google або через email;
- створення профілів учнів і менторів;
- публікація постів з фільтрацією за категоріями;
- внутрішній чат та виставлення інвойсів.

Проведено тестування, яке підтвердило коректність роботи основного функціоналу.
Проєкт має потенціал до масштабування та розширення функціоналу (преміум-функції, рекомендації, календар сесій тощо).

ДОДАТОК Г

Специфікація програмного продукту

Software Requirements Specification

for

Knowlity

Version 1.0 approved

Prepared by Illia Dronov

NURE

19.05.2025

Історія версій

Дата	Опис	Автор	Коментар
19.05.25	Версія 1.0	Дронов Ілля	Створення документу

Зміст

1. Вступ

1.1 Огляд продукту

Платформа **Knowlity**— це веб-програмна система для надання та отримання освітніх і консультаційних послуг у форматі менторства. Вона дозволяє користувачам створювати профілі менторів або учнів, публікувати оголошення про послуги, бронювати сесії, здійснювати оплату через Stripe та вести комунікацію через вбудований чат (Talk.js). Система спрямована на спрощення доступу до менторської підтримки в різних галузях (програмування, дизайн, маркетинг тощо), підвищення якості взаємодії між користувачами та ефективне керування навчальним процесом. Монетизація реалізується через комісію з проведених транзакцій або платні функції для менторів.

1.2 Мета

Метою розробки програмної системи є створення безпечної, масштабованої та зручної платформи, яка дозволяє користувачам у ролі менторів та учнів ефективно взаємодіяти між собою. Система повинна забезпечити централізоване управління профілями, можливість фільтрації послуг за категоріями, швидке бронювання, онлайн-оплату, а також інтегрований канал комунікації. Проєкт також спрямований на підвищення якості надання освітніх послуг, розширення доступу до досвідчених фахівців, а також розвиток культури наставництва. Платформа має на меті створити позитивний користувацький досвід завдяки сучасному інтерфейсу, простому процесу реєстрації та продуманій системі ролей.

1.3 Межі

Програмна система Knowlity включає веб-додаток, який реалізує повний функціонал платформи: реєстрація, автентифікація (через email та Google), створення і перегляд оголошень, внутрішній чат, перегляд розкладу, бронювання сесій та здійснення платежів. Система не включає окремий мобільний додаток.

Підтримуються три основні ролі користувачів: учень, ментор та адміністратор, кожна з яких має доступ лише до відповідного функціоналу.

Серверна частина реалізована за допомогою FastAPI у монолітній структурі, однак з чітким поділом відповідальності між модулями. Клієнтська частина створена на базі Next.js з використанням компонентної архітектури. Контроль за процесом розробки здійснюється через систему управління проектами Trello.

1.4 Посилання

Текст документу містить наступні посилання:

1. Посилання на репозиторії проекту:
   https://github.com/TeraBasedProgrammer/bachelor-work-backend
   https://github.com/TeraBasedProgrammer/bachelor-work-backend

1.5 Означення та абревіатури

В документі використовуються наступні означення та абревіатури:

HTTP – HyperText Transfer Protocol

REST – REpresentational State Transfer

SQL – Structured Query Language

AWS – Amazon Web Services

EC2 - Amazon Elastic Compute Cloud

K8S - Kubernetes

2. Загальний опис

2.1 Перспективи продукту

У зв’язку зі зростанням попиту на індивідуальне навчання, розвитком EdTech-сектору та популяризацією моделі mentor-to-mentee, створення платформи **Knowlity** є актуальним та комерційно доцільним рішенням. Онлайн-менторство стало важливою складовою кар’єрного розвитку, перекваліфікації та професійного зростання, особливо в умовах гнучкого ринку праці. За даними LinkedIn Learning, попит на персоналізоване навчання щороку зростає, що створює сприятливе середовище для подібних цифрових сервісів.

Програмна система **Knowlity** орієнтована на такі сегменти ринку:

1. **Ментори-фахівці** — можуть проводити монетизацію своїх знань, створюючи публічні профілі та приймаючи бронювання.
2. **Учні та фахівці-початківці** — отримують швидкий доступ до консультацій, навчальних сесій та фідбеку.
3. **B2B-партнери (курси, школи, HR-компанії)** — можуть інтегрувати платформу як частину внутрішніх програм навчання або профорієнтації.
4. **EdTech-ринки** — розширення функціоналу платформи дозволяє масштабувати її в сегменти колективного навчання, міні-курсів та сертифікації.
5. **Фриланс та консультаційні ринки** — за допомогою Knowlity ментори можуть не лише навчати, а й супроводжувати фахівців у робочих проектах.
6. **Платформи для професійного зростання** — система може стати інструментом для побудови персоналізованих траєкторій розвитку кар’єри.

2.2 Функції продукту

Knowlity — це веб-платформа з повним циклом взаємодії між ментором і учнем. Серед основного функціоналу:

- **Профілі користувачів** — реєстрація, налаштування профілю, вибір ролі (ментор, учень, обидва).

- **Публікація оголошень** — створення пропозицій щодо консультацій або менторських сесій з можливістю вказати категорію, ціну та опис.
- **Фільтрація та пошук** — розширений пошук менторів за спеціалізацією, ціною, рейтингом тощо.
- **Система бронювання** — можливість бронювати сесії, керувати розкладом.
- **Платіжна система** — інтеграція з Stripe для здійснення онлайн оплат.
- **Чат** — вбудований месенджер на основі Talk.js для попереднього спілкування.
- **Панель адміністратора** — верифікація менторів.
- **Система повідомлень** — email та in-app сповіщення про бронювання, нові повідомлення та статуси оплати.
- **Безпека** — автентифікація через JWT/Google, обмеження доступу до чутливих дій згідно ролей.

2.3 Характеристики користувачів

Система Knowlity взаємодіє з трьома основними типами користувачів:

- **Ментор** — створює оголошення, керує сесіями, спілкується з учнями, переглядає історію оплат.
- **Учень** — переглядає оголошення, фільтрує менторів, бронює сесії, оплачує послуги.
- **Адміністратор** — проводить верифікацію менторів

Усі користувачі можуть зареєструватися за допомогою email або Google, редагувати профіль, переглядати історію своїх взаємодій та отримувати повідомлення про дії в системі.

2.4 Загальні обмеження

Knowlity складається з клієнтської та серверної частин. Сервер реалізовано за допомогою **FastAPI** (Python), клієнт — з використанням **Next.js**. Архітектура побудована у вигляді модульного моноліту з чітко поділеними зонами відповідальності, що дозволяє легко масштабувати окремі компоненти.

Для зберігання даних використовується **PostgreSQL**, медіа файли обробляються через **Cloudinary**. Платформа розгортається в хмарі (AWS EC2), з підтримкою CI/CD (GitHub Actions). У майбутньому розглядається можливість переходу до мікросервісної структури з використанням та K8S.

Клієнтська частина розробляється з урахуванням адаптивності, використовуючи Tailwind CSS та компонентну архітектуру. NextAuth забезпечує захищену автентифікацію, а Talk.js — стабільну чат-комунікацію. Stripe використовується для обробки онлайн-платежів.

2.5 Припущення та залежності

Припущення:

1. Ринок потребує інструментів для менторства та індивідуального навчання.
2. Користувачі мають базову цифрову грамотність для взаємодії з платформою.
3. Інтегровані сторонні сервіси (Stripe, Talk.js) працюватимуть стабільно.

Залежності:

1. Зовнішні API: Stripe (оплата), Talk.js (чат), Google OAuth (автентифікація).
2. Серверна частина — FastAPI, PostgreSQL, Cloudinary.
3. Клієнтська частина — Next.js, Tailwind CSS, NextAuth, React Hook Form.
4. CI/CD — GitHub Actions.
5. Хостинг — AWS.
6. Продуктивність платформи залежить від стабільності інтернет-з'єднання користувача та середовища хостинга.
7. Процес розробки залежить від часу команди на вивчення, впровадження й тестування функціоналу.

3. Конкретні вимоги

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Після переходу на платформу Knowlity, неавторизованого користувача автоматично перенаправляє на сторінку входу в систему (рисунок 1).

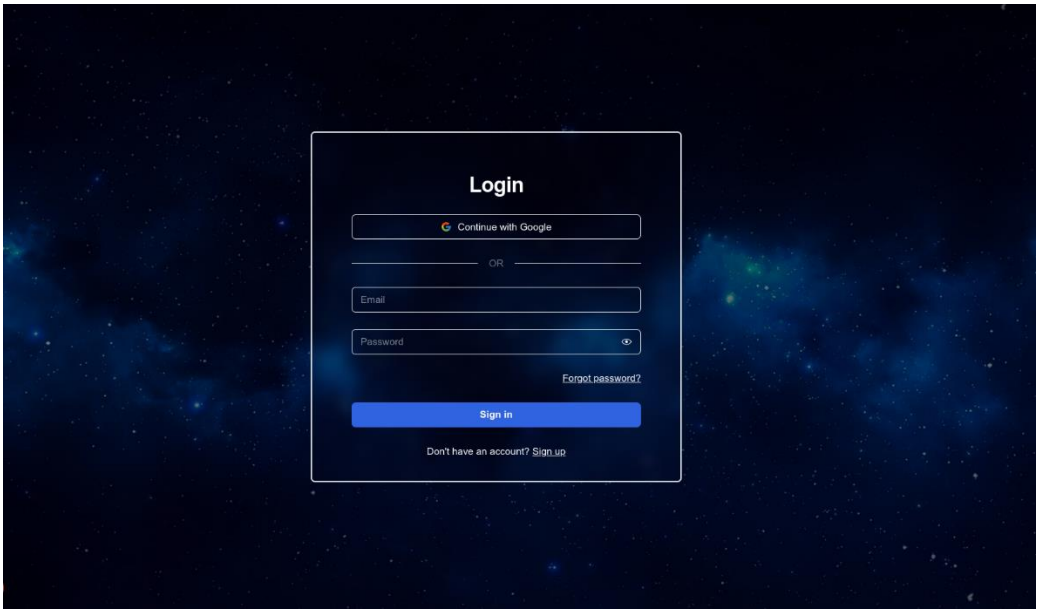


Рисунок 1 — Сторінка входу

Сторінка входу містить форму автентифікації, яка включає поля для введення електронної пошти та пароля, а також кнопку входу через Google. Під формою знаходиться кнопка переходу на сторінку реєстрації. Оформлення має бути простим, інтуїтивно зрозумілим, з фокусом на зручність входу до системи.

На сторінці реєстрації (рисунок 2) користувач має змогу створити обліковий запис як учень. Поля реєстрації включають: електронну пошту, номер телефону, ім’я та пароль. Також присутня кнопка входу через Google. Роль користувача не вибирається вручну — за замовчуванням усі зареєстровані користувачі отримують

роль учня. Перед завершенням реєстрації користувач має погодитися з політикою конфіденційності. Додатково присутня кнопка переходу на форму входу.

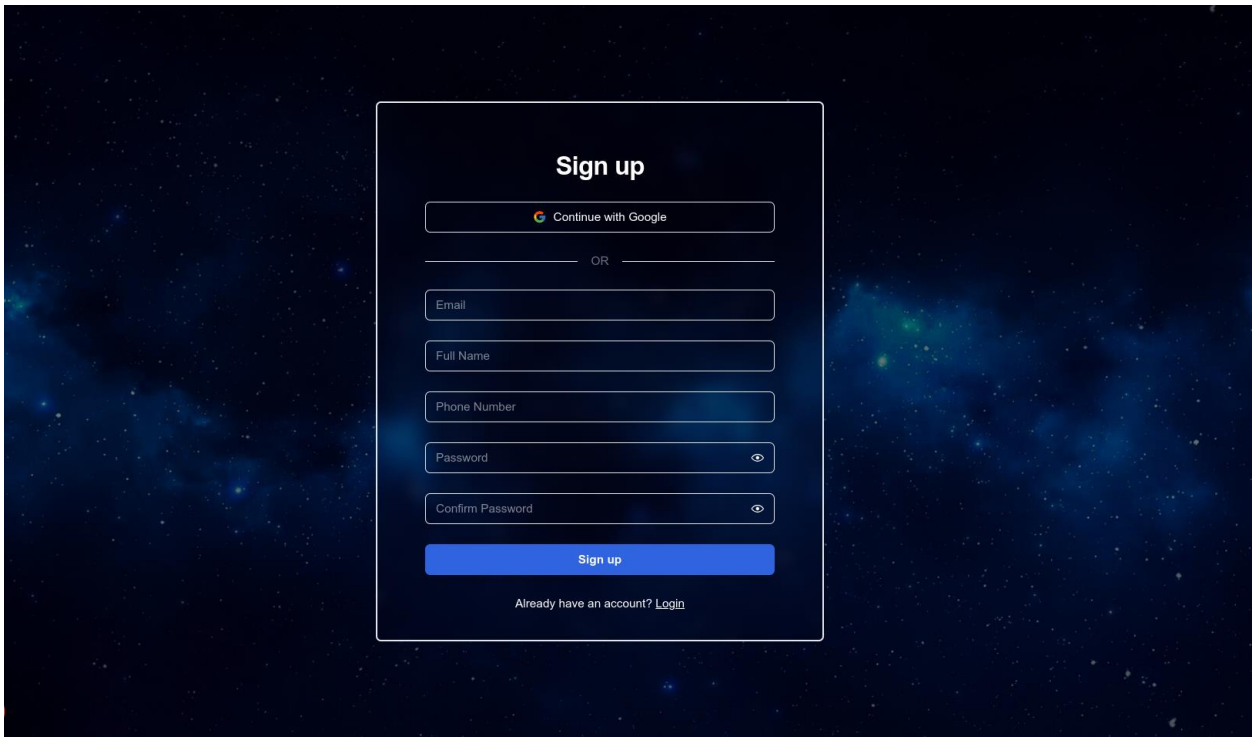


Рисунок 2 — Форма реєстрації користувача

Після входу до системи користувач потрапляє у профіль. Спочатку йому пропонується заповнити базові особисті дані (рисунок 3): завантажити фотографію, додати номер телефону, обрати категорії для пошуку оголошень.

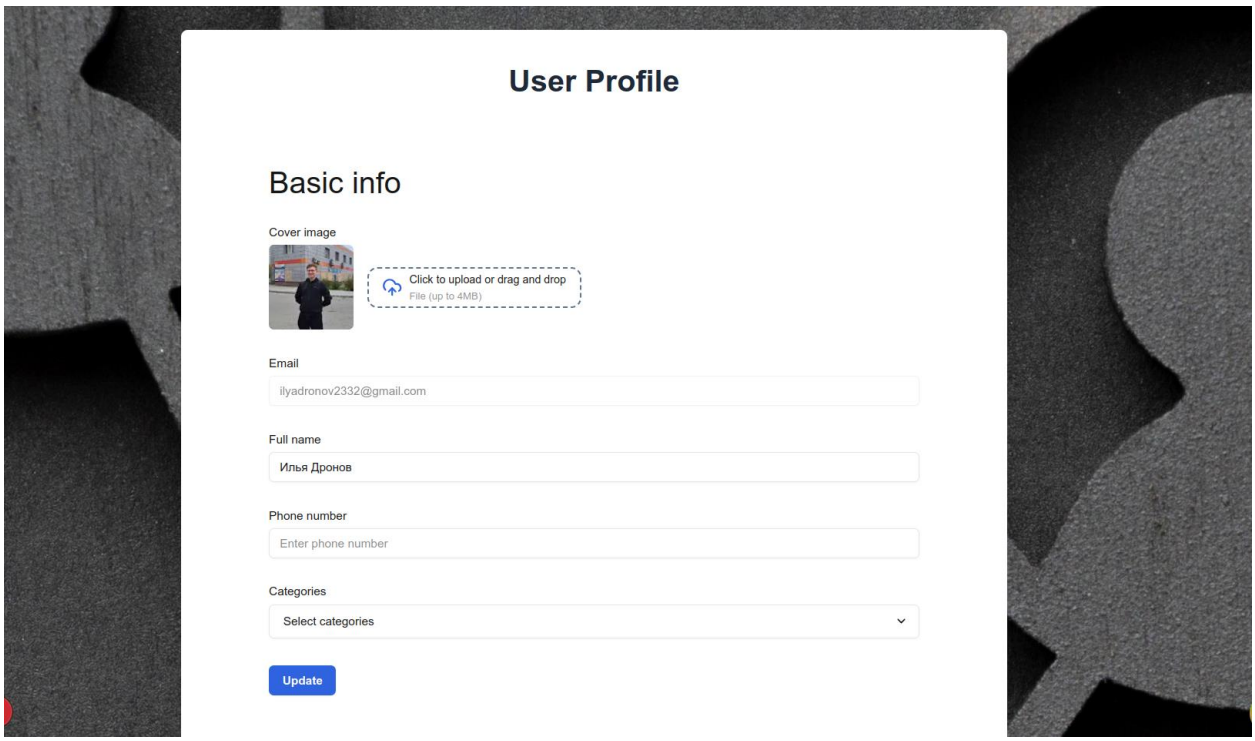


Рисунок 3 — Заповнення особистого профілю

Після цього користувач може подати заявку на створення менторського профілю (рисунок 4). Ця форма включає: завантаження фото паспорта, текст про себе, відео про себе, резюме (файл) та вибір категорій, у яких користувач бажає надавати менторські послуги.

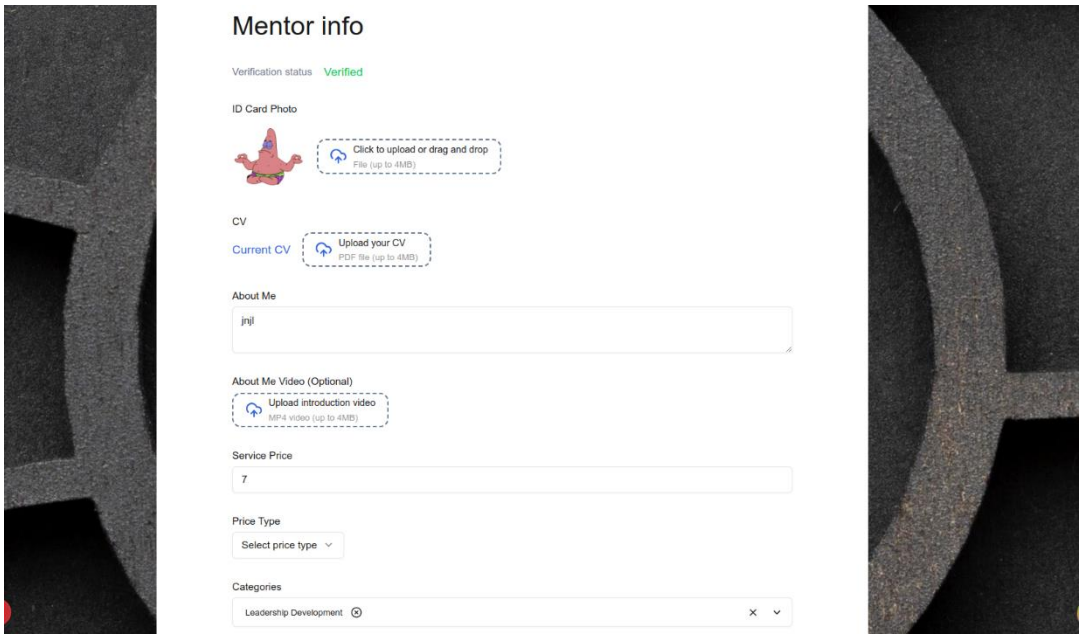


Рисунок 4 — Форма створення менторського профілю

Комунікація між учнем і ментором реалізована у вигляді вбудованого чату (рисунок 5). У чаті відображаються повідомлення, інформація про співрозмовника та зручний інтерфейс для відповіді. Платформа підтримує як текстові повідомлення, так і push-сповіщення про нові вхідні.

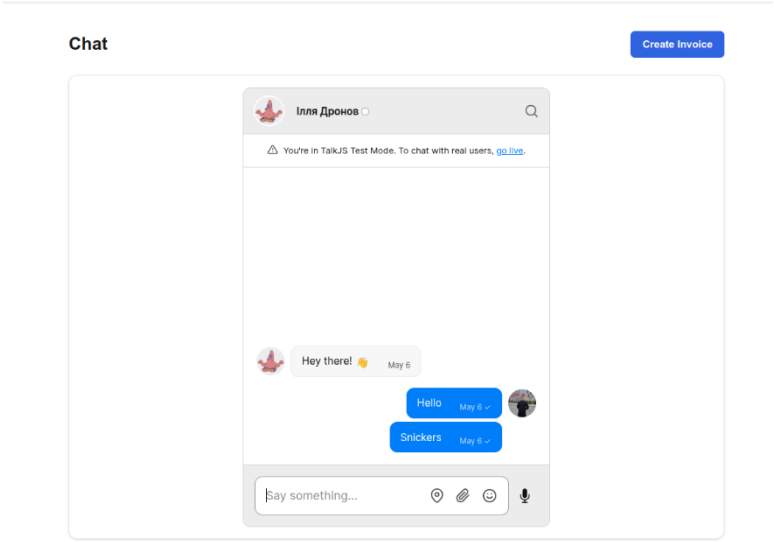


Рисунок 5 — Чат між користувачами

Усі фінансові операції (оплати сесій, комісії) фіксуються у вигляді інвойсів, які доступні користувачам у відповідному розділі профілю (рисунок 6). Кожен інвойс містить інформацію про дату, суму та сторони оплати

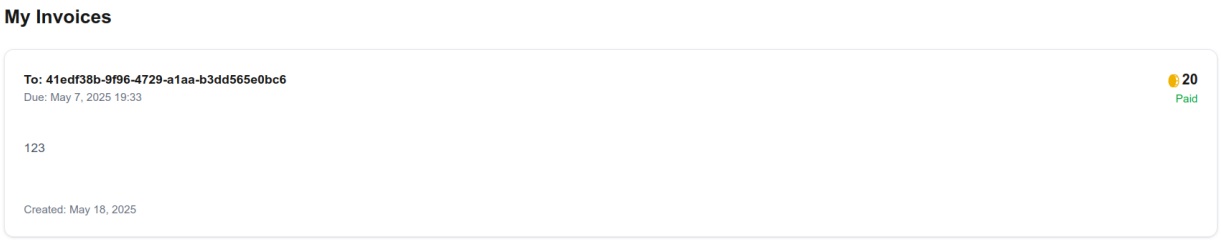


Рисунок 6 — Інвойс користувача

3.1.2 Апаратний інтерфейс

Система Knowlity не взаємодіє з апаратними пристроями напряму, тому специфічні апаратні інтерфейси відсутні.

3.1.3 Програмний інтерфейс

Система сумісна з усіма сучасними браузерами (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge). Платформа розроблена з використанням HTML5, CSS3, JavaScript ES6, React та Next.js. Мобільна версія доступна через адаптивний веб-інтерфейс, окремі застосунки для iOS/Android поки не розробляються.

3.1.4 Комунікаційний протокол

Взаємодія між клієнтською та серверною частинами реалізується через HTTPS та REST API. Комунікація в реальному часі (чат, оновлення повідомлень) забезпечується через WebSocket (Talk.js). Внутрішня взаємодія між модулями може доповнюватися асинхронною обробкою через черги повідомлень (у майбутньому).

3.1.5 Обмеження пам’яті

Жодних специфічних обмежень пам’яті не встановлюється. Платформа працює у веб-браузері, і пам’ять визначається ресурсами пристрою користувача та налаштуваннями браузера.

3.1.6 Операції

Платформа реалізує REST-архітектуру з такими HTTP-методами:

- GET — отримання даних (профіль, оголошення, повідомлення)
- POST — створення (реєстрація, нове бронювання)
- PUT / PATCH — редагування (профілю, послуги)
- DELETE — видалення (бронювання, оголошення)

3.1.7 Функції продукту

ГФ-1: Реєстрація та автентифікація користувачів

ГФ-2: Створення, перегляд, редагування та видалення оголошень

ГФ-3: Бронювання сесій між учнями та менторами

ГФ-4: Інтеграція з Stripe для обробки платежів

ГФ-5: Вбудований чат між користувачами (Talk.js)

ГФ-6: Рейтингова система, можливість залишати відгуки

ГФ-7: Панель адміністратора для модерації контенту та верифікації менторів

ГФ-8: Система повідомлень (in-app та email)

ГФ-9: Багатомовна підтримка (українська / англійська)

3.2 Властивості програмного продукту

Платформа Knowlity орієнтована на широку аудиторію з різних регіонів. Стабільне інтернет-з’єднання є обов’язковим. Час відповіді системи не повинен перевищувати 1 хвилини, навіть при високому навантаженні. Розмежування функціоналу здійснюється за ролями (ментор, учень, адміністратор). Дані зберігаються централізовано у захищеній базі даних PostgreSQL. Інтерфейс адаптивний і підтримує мобільні пристрої.

3.3 Атрибути програмного продукту

3.3.1 Надійність

- Система повинна повертати лише перевірені дані.
- У випадку помилки — показує зрозуміле повідомлення без внутрішніх технічних деталей.

3.3.2 Доступність

- Доступна українською та англійською мовами.
- Доступ до функціоналу мають лише авторизовані користувачі.

- Адміністративні функції доступні лише адміністраторам.

3.3.3 Безпека

- Обліковий запис захищений паролем (мінімум 6 символів і хоча б одна цифра).
- Доступ лише після вірного введення логіна і пароля / авторизації через Google.
- Дані компаній зберігаються окремо від даних користувачів.
- Функціонал доступний згідно з ролями.

3.3.4 Супровід

- Система має бути легко розширювана новими функціями.
- Компоненти системи (сервер, клієнт, чат, оплати) мають бути відокремлені.

3.3.5 Переносимість

- Веб-додаток підтримується у сучасних браузерах.
- Адаптивний дизайн забезпечує підтримку планшетів і смартфонів.

3.3.6 Продуктивність

- Перехід між вкладками — до 1 секунди.
- Лист підтвердження дій (наприклад, запрошення) надсилається протягом 5 хвилин.

3.4 Вимоги до бази даних

У проєкті Knowlity використовується PostgreSQL як основна СУБД. Переваги:

- висока масштабованість;
- транзакційна надійність (ACID);
- потужні засоби контролю доступу;

- підтримка складних зв’язків та індексації;
- інтеграція з FastAPI та SQLAlchemy.

База даних зберігає користувачів, оголошення, категорії, сесії, транзакції, повідомлення, відгуки та адміністративні записи.

3.5 Інші вимоги

- Незареєстровані користувачі мають доступ лише до загальнодоступної головної сторінки.
- Уся інша функціональність вимагає авторизації.
- Всі використовувані зображення, ілюстрації, та медіа мають бути вільні від порушення авторських прав.
- Інтерфейс має бути повністю адаптивним — автоматично підлаштовуватися під ширину екрана.
- Усі змінні шрифти, кнопки, елементи інтерфейсу повинні бути придатні до взаємодії на мобільних пристроях.

4. Додаткові матеріали

4.1 Діаграми баз даних

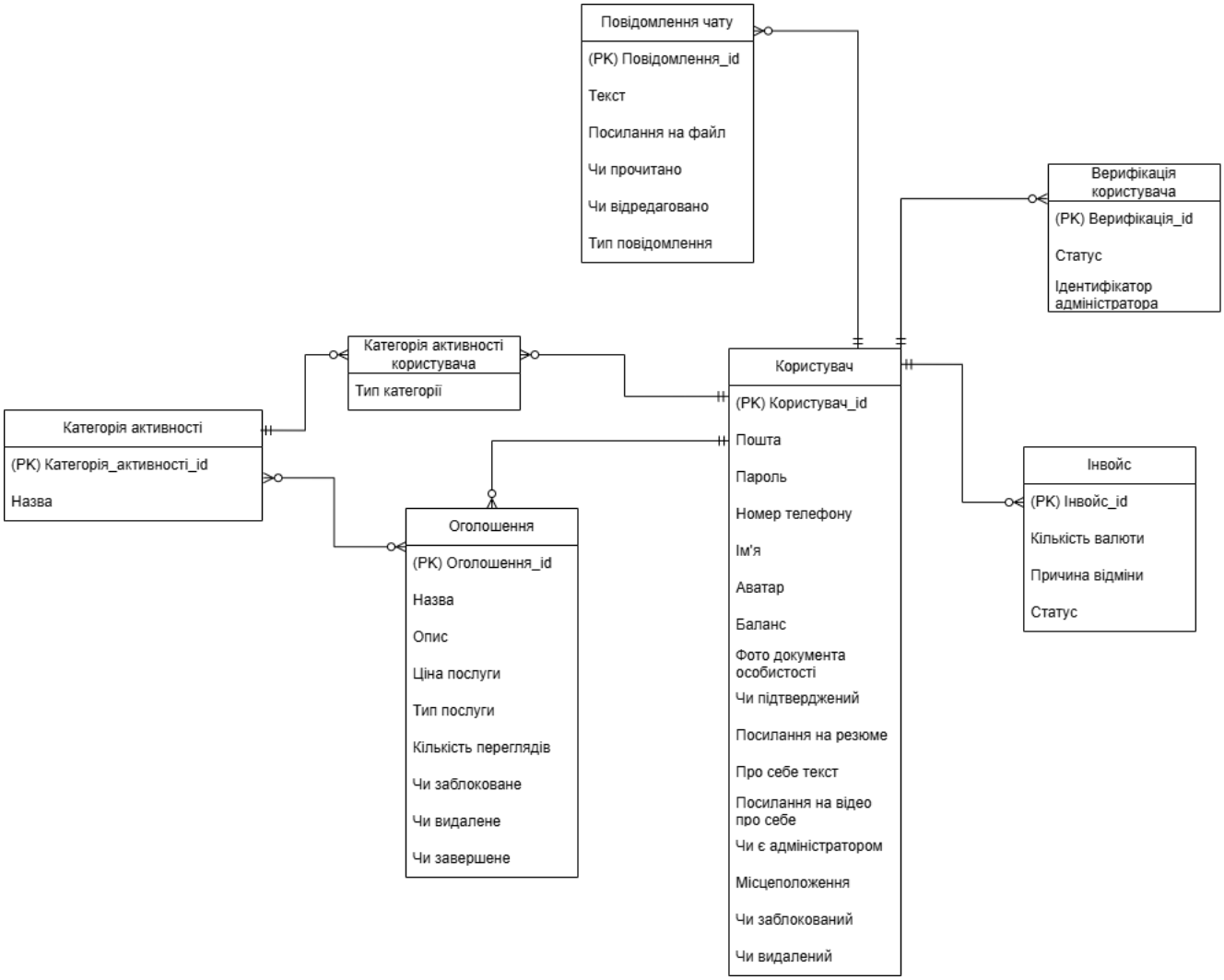


Рисунок 7 – ER-діаграма бази даних

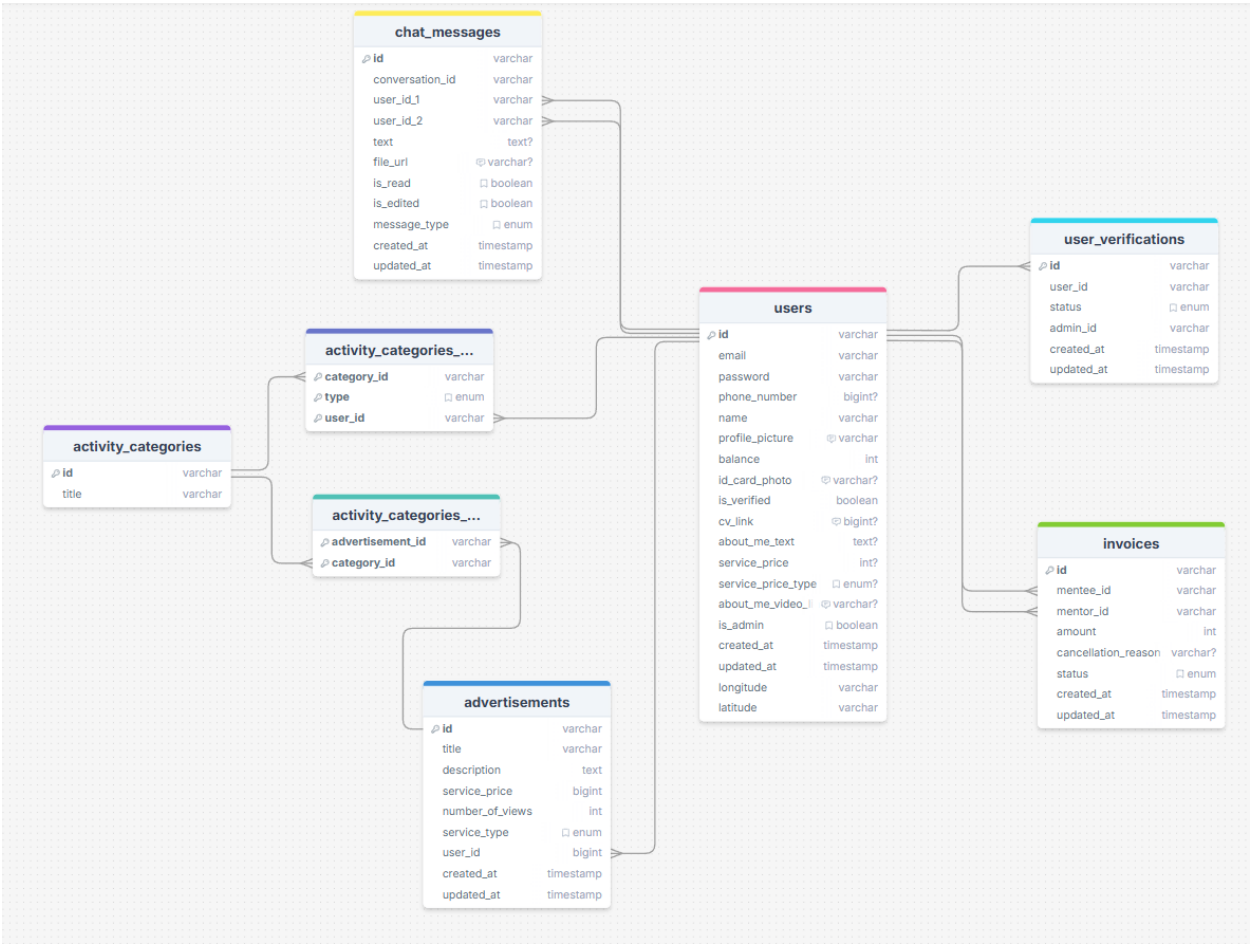


Рисунок 8 – Логічна модель бази даних

4.2 Діаграма прецедентів

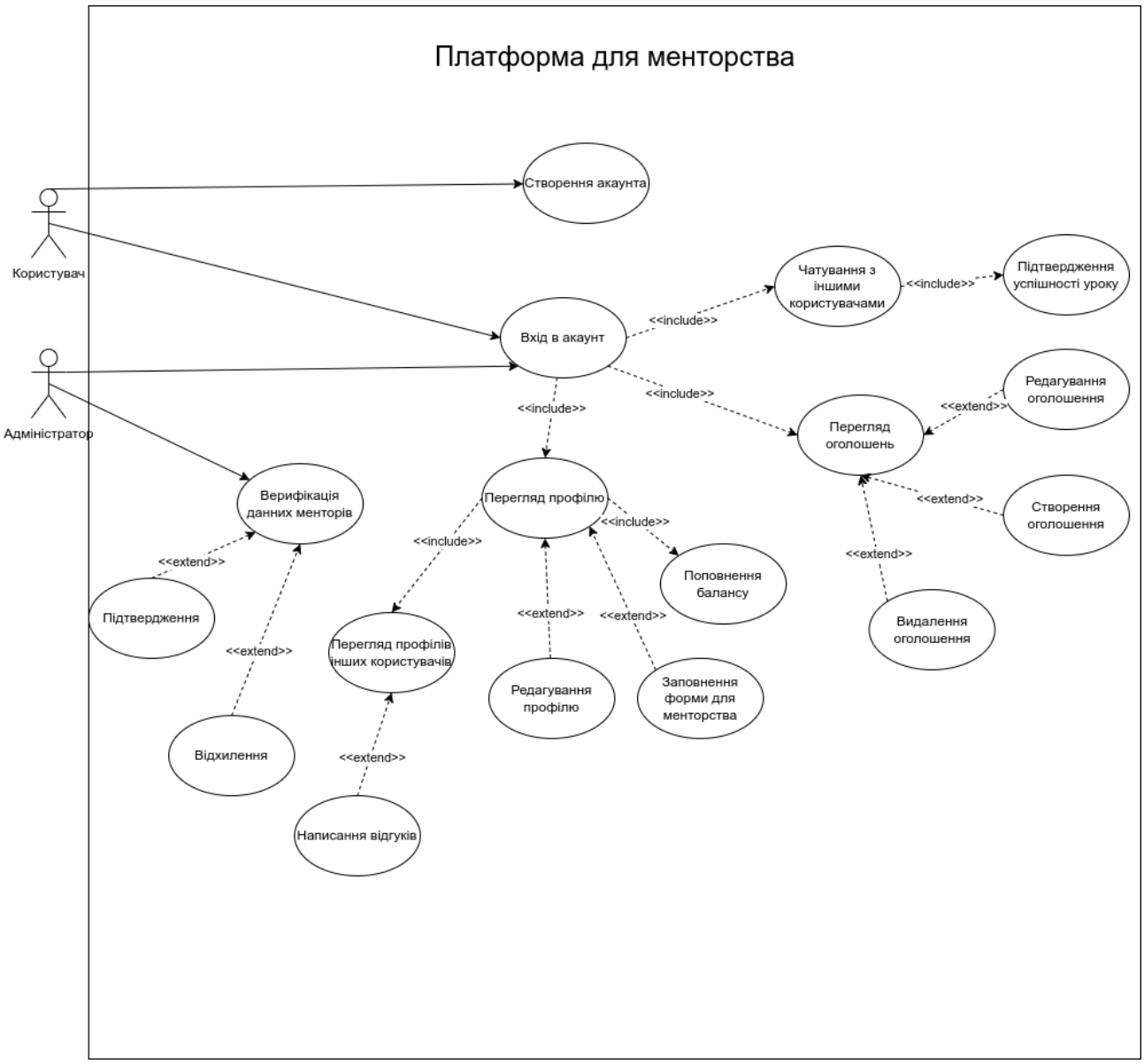


Рисунок 9 – Діаграма прецедентів

4.3 BPMN діаграма

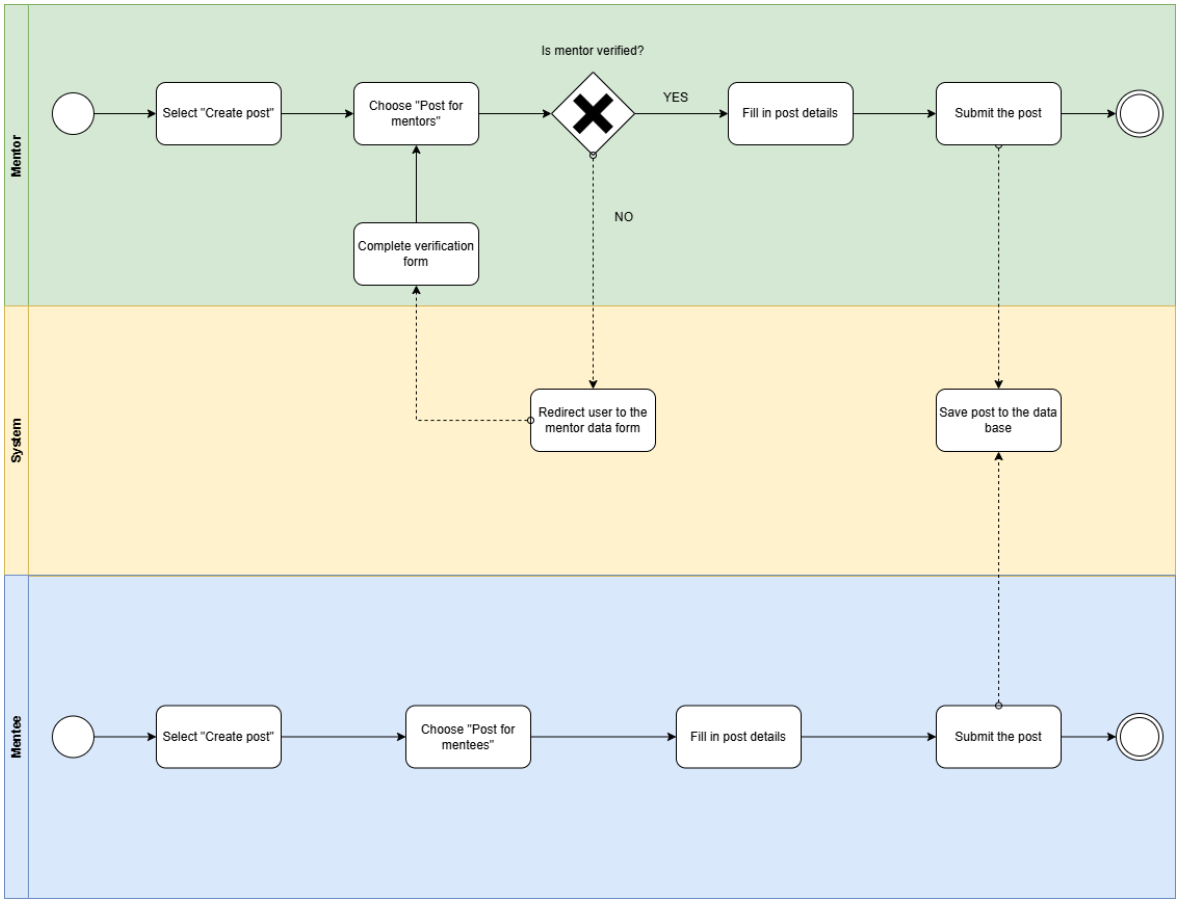
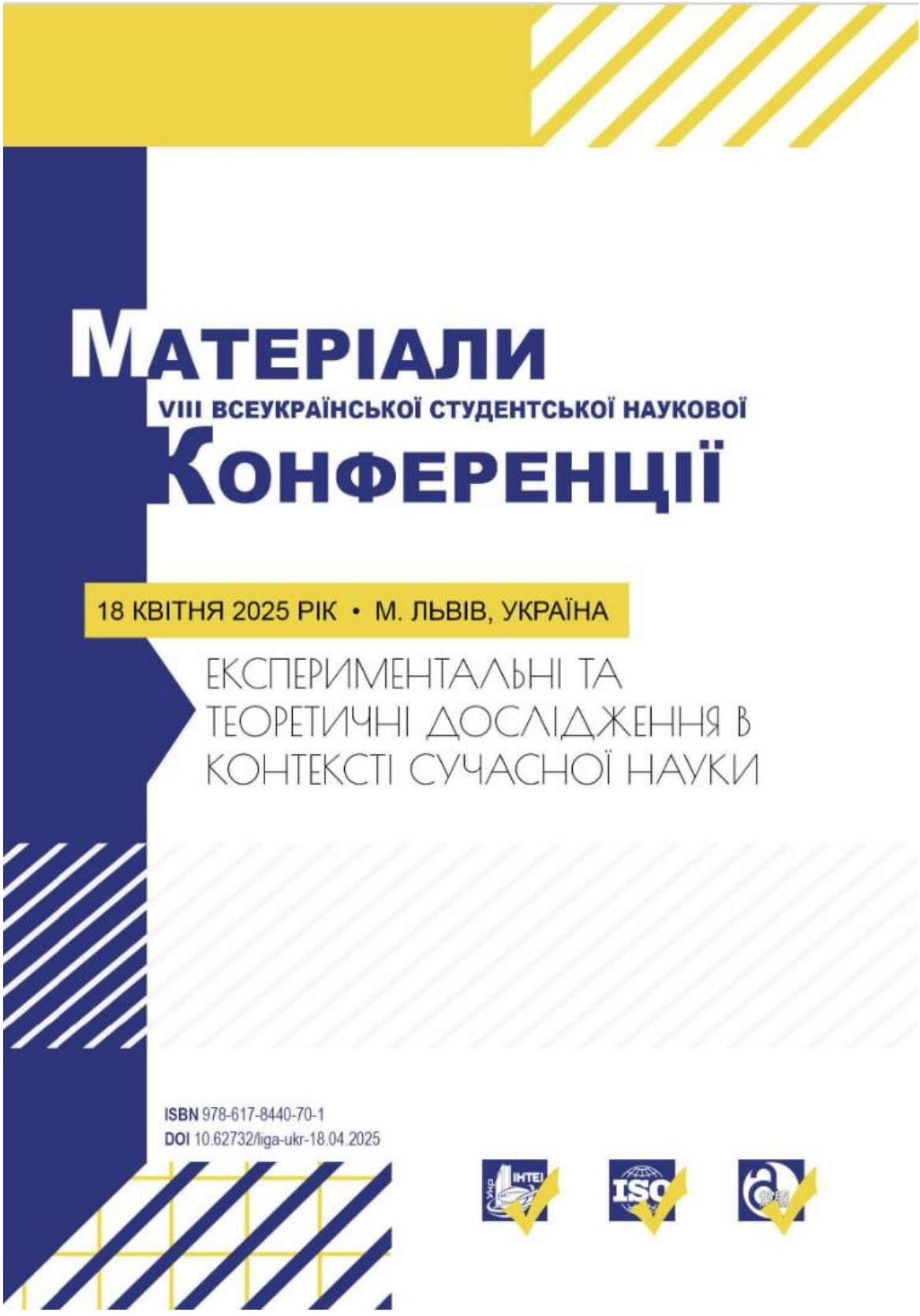


Рисунок 10 – BPMN діаграма

ДОДАТОК Д

Тези доповіді



Дронов Ілля Олександрович, здобувач вищої освіти факультету комп'ютерних наук
«Харківський національний університет радіоелектроніки», Україна

Науковий керівник: Русакова Наталія Євгенівна, доцент кафедри
Програмної інженерії
«Харківський національний університет радіоелектроніки», Україна

ПРОГРАМНА СИСТЕМА ДЛЯ НАДАННЯ ТА ОТРИМАННЯ НАВЧАЛЬНИХ ПОСЛУГ

Сучасні менторські платформи потребують розумних рішень для управління користувачами, сесіями та комунікацією [1]. В цій роботі представлено систему «Knowlity» - спрощену платформу, створену спеціально для підтримки індивідуальних менторів, які пропонують свої послуги онлайн. Вона спрощує планування сесій, управління профілями та безпечну взаємодію між менторами та учнями. Система передбачає рольовий доступ: ментори можуть керувати доступністю, цінами та повідомленнями, а учні можуть знаходити менторів, бронювати сесії та залишати відгуки. Створена на основі сучасних веб-технологій - Next.js, TailwindCSS та Shadcn - вона пропонує швидкий, чистий та чуйний користувацький інтерфейс [2]. Безпечна автентифікація здійснюється за допомогою JWT та Google Sign-in, а основна логіка працює на основі FastAPI. «Knowlity» дає можливість індивідуальним менторам ефективно управляти своїми послугами та надавати високоякісні рекомендації своїм підопічним.

На сучасному ринку існує кілька платформ, орієнтованих на освітні сервіси та менторство. Серед них — Clarity.fm [3], сервіс для експертних консультацій, що підходить переважно для бізнес-напрямів, але має обмежені можливості персоналізації, а також MentorCruise [4], що пропонує розширену систему пошуку менторів, проте має менш гнучкий функціонал для індивідуального брендингу та кастомізації (рис. 1).

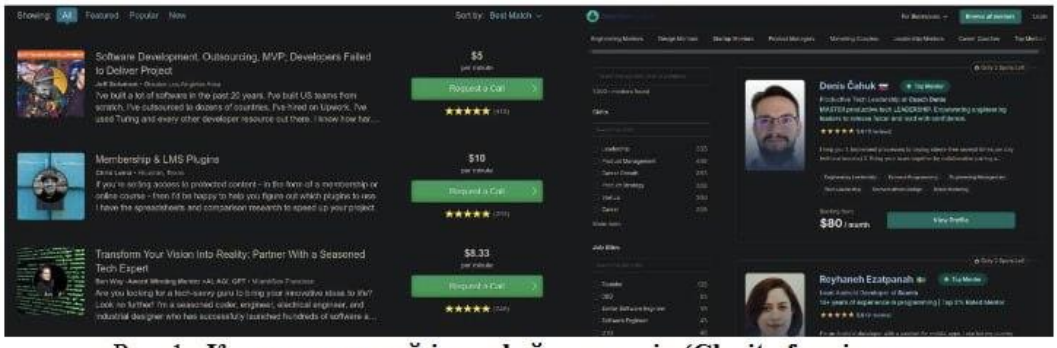


Рис. 1. **Користувацький інтерфейс аналогів (Clarity.fm ліворуч, MentorCruise праворуч)**

Запропонована в роботі система "Knowlity" вирізняється високим рівнем автоматизації процесів, інтуїтивно зрозумілим інтерфейсом і легкою інтеграцією з сучасними сервісами, такими як Google-автентифікація та Stripe. Завдяки цим перевагам система сприяє підвищенню якості надання менторських послуг та забезпечує ефективну взаємодію між користувачами. У цій роботі представлено опис

програмної системи "Knowlity", розробленої для підтримки індивідуальних менторів і їхніх клієнтів, з чітким розподілом ролей та функціональним інтерфейсом.

У системі "Knowlity" основну функціональність реалізують ролі Ментор та Учень. Кожен користувач може виступати в обох ролях залежно від заповнених профільних даних: наявності резюме, відео-презентації, опису досвіду та встановленої вартості послуг. Ментори можуть створювати оголошення про свої послуги, обираючи категорії діяльності, тип консультації та ціну. Учні мають змогу переглядати ці оголошення, фільтрувати за напрямками та напряму взаємодіяти з менторами через вбудовану систему чатів. Комунікація реалізована за допомогою таблиць "chat_conversations" і "chat_messages", які забезпечують зберігання історії діалогів, повідомлень, файлів та статусів прочитання.

Роль Адміністратора в системі не є основною, але виконує важливі модераційні функції: перевірка профілів менторів, контроль якості контенту, розгляд скарг та управління загальною безпекою платформи.

На відміну від традиційних освітніх платформ, "Knowlity" вирізняється гнучкою архітектурою та підтримкою індивідуальних менторів. Інтерфейс, побудований з використанням Next.js, TailwindCSS та Shadcn, забезпечує простоту, адаптивність і зручність користування. Система використовує JWT та Google Sign-In для автентифікації, а основна бізнес-логіка реалізована через FastAPI. Автоматизація взаємодії між ментором і учнем, а також обробка транзакцій сприяють ефективності та зручності користування платформою.

На рисунку 2 представлено модель реляційної бази даних, реалізовану в PostgreSQL. Відношення "users" зберігає дані про користувачів, включаючи статус верифікації, координати, фінансову інформацію та профільні атрибути. Створені менторами оголошення зберігаються у таблиці "advertisements" та пов'язуються з категоріями через таблицю-зв'язку.

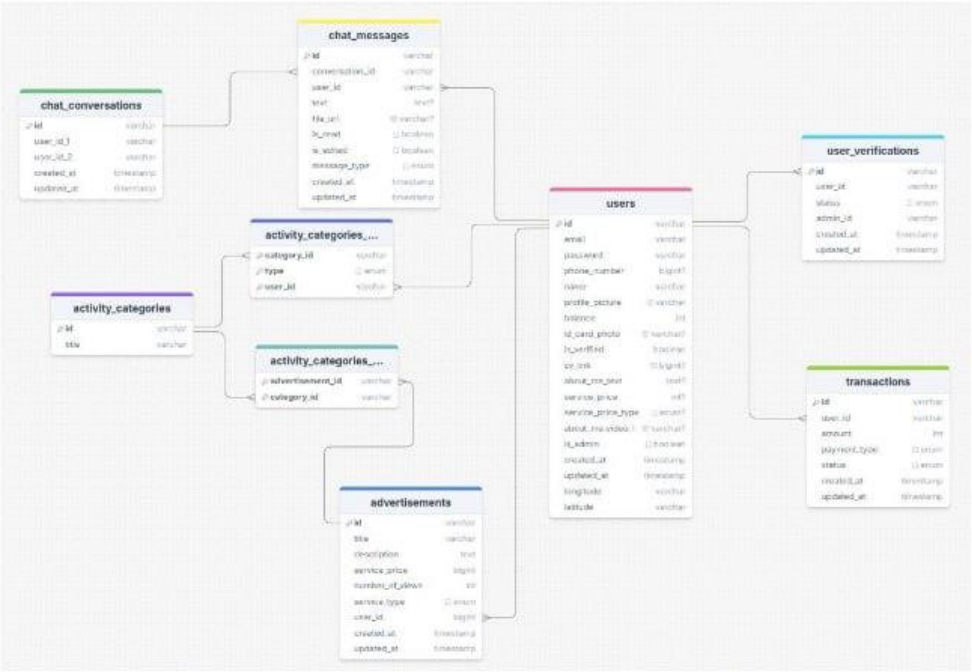


Рис. 2. **Схема реляційної бази даних до програмної системи**

Модуль фінансових операцій реалізовано через таблицю “transactions”, а процес верифікації менторів — через “user_verifications”. Система спроєктована з урахуванням масштабованості, безпеки та зручності для підтримки менторських взаємодій у цифровому середовищі. Під час сесій можливе надання додаткових матеріалів — документів, презентацій, відео — що зберігаються у таблиці “advertisements” або у профільних полях користувача. Взаємодія між ментором і учнем фіксується через систему чатів, реалізовану у таблицях “chat_messages” та “chat_conversations”, що забезпечує обмін повідомленнями, файлами та посиланнями.

Веб-застосунок реалізує ключовий функціонал для ефективної взаємодії: пошук оголошень за ключовими словами, категоріями послуг і застосуванням фільтрів. Система чатів дозволяє учасникам обмінюватися повідомленнями й файлами. Оплата здійснюється під час бронювання, але кошти резервуються і надходять ментору лише після підтвердження завершення сесії. Адміністративний модуль передбачає перевірку заявок на менторство, модерацію профілів і моніторинг активності користувачів.

Список використаних джерел:

1. Mentorship Platforms and Communities for Founders and Entrepreneurs [Електронний ресурс] // IT Jet. – Режим доступу: https://itjet.io/blog/mentorship-platforms (дата звернення: 10.04.2025).
2. Вовк О. В., Феделова З.Є. Важливість прототипування в розробці веб ресурсів // Проблеми моделювання : Матеріали молодіжної школи-семінару, 14-28 травня 2024 р. – Харків : ХНУРЕ, 2024. – С. 175.
3. Clarity.fm – On Demand Business Advice [Електронний ресурс] // Clarity.fm. – Режим доступу: https://clarity.fm/ (дата звернення: 05.04.2025).
4. MentorCruise – Find a Mentor to Help You Grow [Електронний ресурс] // MentorCruise. – Режим доступу: https://mentorcruise.com/ (дата звернення: 10.04.2025).