НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Курсова робота

з дисципліни «Програмування»

на тему: «Форум»

Виконав:
студент 1 курсу, групи ІА-33
Вовчок Максим Миколайович _____
(підпис)

Керівник:
асистент кафедри ІСТ
Мягкий Михайло Юрійович _____
(підпис)

Засвідчую, що у цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2024 року

ЗМІСТ

ВСТУП

З розвитком Інтернету та інформаційних технологій онлайн-комунікація стала невід'ємною частиною повсякденного життя людей. Форум, як одна з форм цієї комунікації, відіграє важливу роль у створенні спільнот за інтересами, обміні знаннями та досвідом, а також у вирішенні різноманітних питань. Від технічних обговорень до мистецьких диспутів – форуми надають платформу для відкритої та вільної дискусії.

Сучасні форуми надають широкий спектр можливостей для користувачів: від простого обміну текстовими повідомленнями до інтеграції мультимедійних елементів та функцій соціальних мереж. Зручний інтерфейс, можливість швидкого пошуку інформації та активне модераторство – все це сприяє зростанню популярності форумів.

Метою даної роботи є створення системи форуму, що дозволяє створювати теми, обговорювати їх, обмінюватись думками та отримувати зворотній зв'язок у зручній та організованій формі. Для досягнення цієї мети система повинна мати такі властивості та вирішувати такі задачі:

- Забезпечення зручного інтерфейсу для користувачів різних ролей (гості, користувач, адміністратор).
- Можливість створення та видалення тем.
- Інтеграція з базою даних для збереження інформації про теми та повідомлення.
- Наявність пошукової функції, для того, аби швидкого знаходити необхідну інформацію за ключовими словами, та категоріями.
- Таким чином, розробка форуму не тільки задовольнить потреби користувачів у комунікації, але й сприятиме створенню активних та взаємодіючих спільнот.

1 ВИМОГИ ДО СИСТЕМИ

1.1 Функціональні вимоги до системи

Гість:

Гість повинен мати можливість переглядати теми.

Гість повинен мати можливість переглядати дописи у темах.

Зареєстрований користувач:

Зареєстрований користувач повинен мати можливість створювати нові дописи у вже існуючих темах.

Зареєстрований користувач повинен мати можливість переглядати теми.

Зареєстрований користувач повинен мати можливість переглядати дописи у темах.

Адміністратор:

Адміністратор повинен мати можливість створювати нові теми.

Адміністратор повинен мати можливість видаляти та оновлювати існуючі теми.

Адміністратор повинен мати можливість переглядати теми.

Адміністратор повинен мати можливість переглядати дописи у темах.

1.2 Нефункціональні вимоги до системи

Система має відповідати наступним функціональним вимогам:

– система повинна мати відкриту архітектуру;

– система повинна мати веб-інтерфейс;

– інтерфейс користувача має бути зручним та інтуїтивно-зрозумілим;

– система повинна бути крос-платформною.

2 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

2.1 Діаграма прецедентів

Діаграма прецедентів системи представлена на рис. 2.1.

Гість має можливість переглядати теми та дописи у темах. Зареєстрований користувач, крім перегляду тем та дописів, має можливість створювати нові дописи у вже існуючих темах. Адміністратор, окрім функціоналу доступного гостю та зареєстрованому користувачу, повинен мати можливість створювати нові теми та видаляти існуючі.

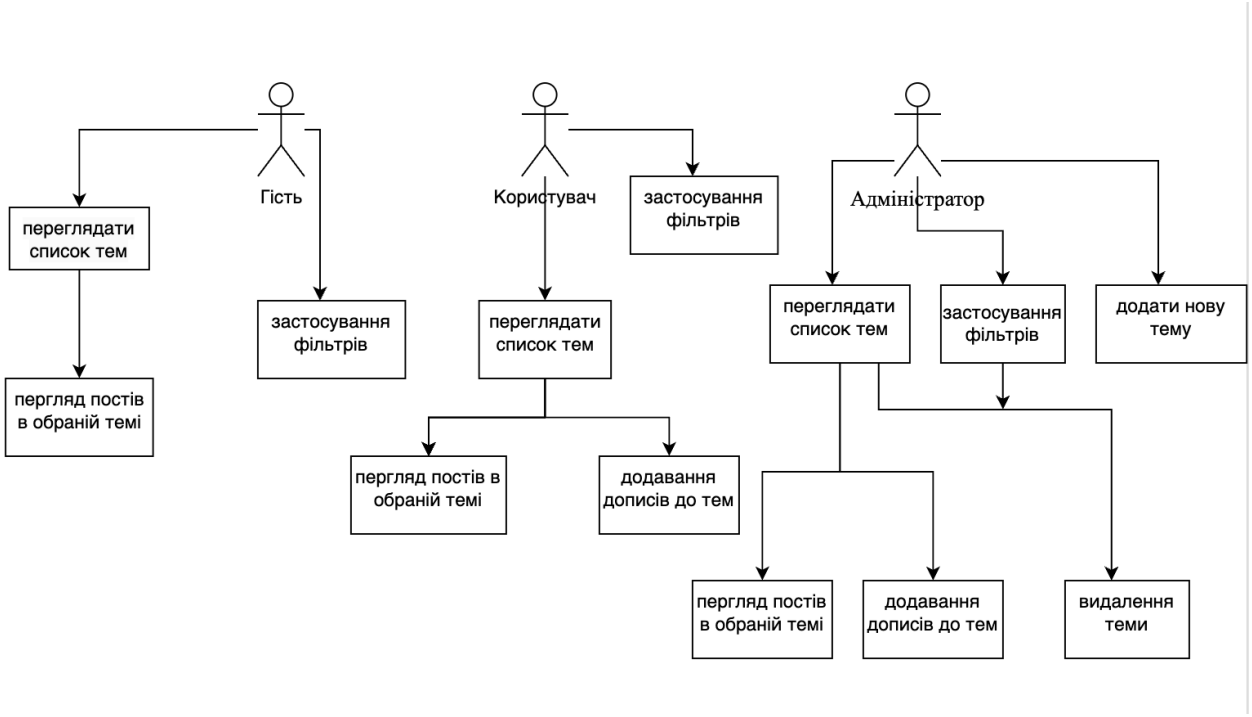


Рисунок 2.1 – Діаграма прецедентів

2.2 Опис сценаріїв використання системи

Детальні описи сценаріїв використання наведено у таблицях 2.1 – 2.5.

Назва	Переглянути список всіх тем
ID	1
Опис	Гість або користувач, можуть переглядати всітеми, які були додані адміністратором
Актори	Гість, користувач, адміністратором
Вигоди компанії	Можливість перегляду списку всіх тем користувачами дозволяє компанії забезпечити більш зручний та швидкий доступ до інформації, що стимулює активність користувачів на форумі.
Частота користування	Постійно
Тригери	Користувачу потрібно зайти, як гість, або увійти в акаунт
Передумови	Зайти, як гість, або увійти в акаунт
Постумови	Користувач потрапляє на сторінку зі списком всіх тем
Основний розвиток	Користувач потрапляє на сторінку з темами та може їх переглядати
Альтернативні розвитки	–
Виняткові ситуації	–

Назва	Додати нову тему
ID	2
Опис	Користувач може обрати категорію і зробити пошук по вибраній категорії, або зробити пошук для всіх категорій
Актори	Адміністратор
Вигоди компанії	Активний та насичений контентом форум приваблює нових користувачів, що збільшує кількість відвідувачів
Частота користування	Постійно
Тригери	Адміністратору треба натиснути кнопку для додавання теми та заповнити всі поля
Передумови	Адміністратору потрібно натиснути на кнопку add topic та заповнити всі поля і відправити
Постумови	Адміністратор додає нову тему до всіх інших тем
Основний розвиток	Спочатку адміністратор додає нову тему, потім його перекине на сторінку з усіма темами, де також буде відображатись щойно додана тема
Альтернативні розвитки	–
Виняткові ситуації	–

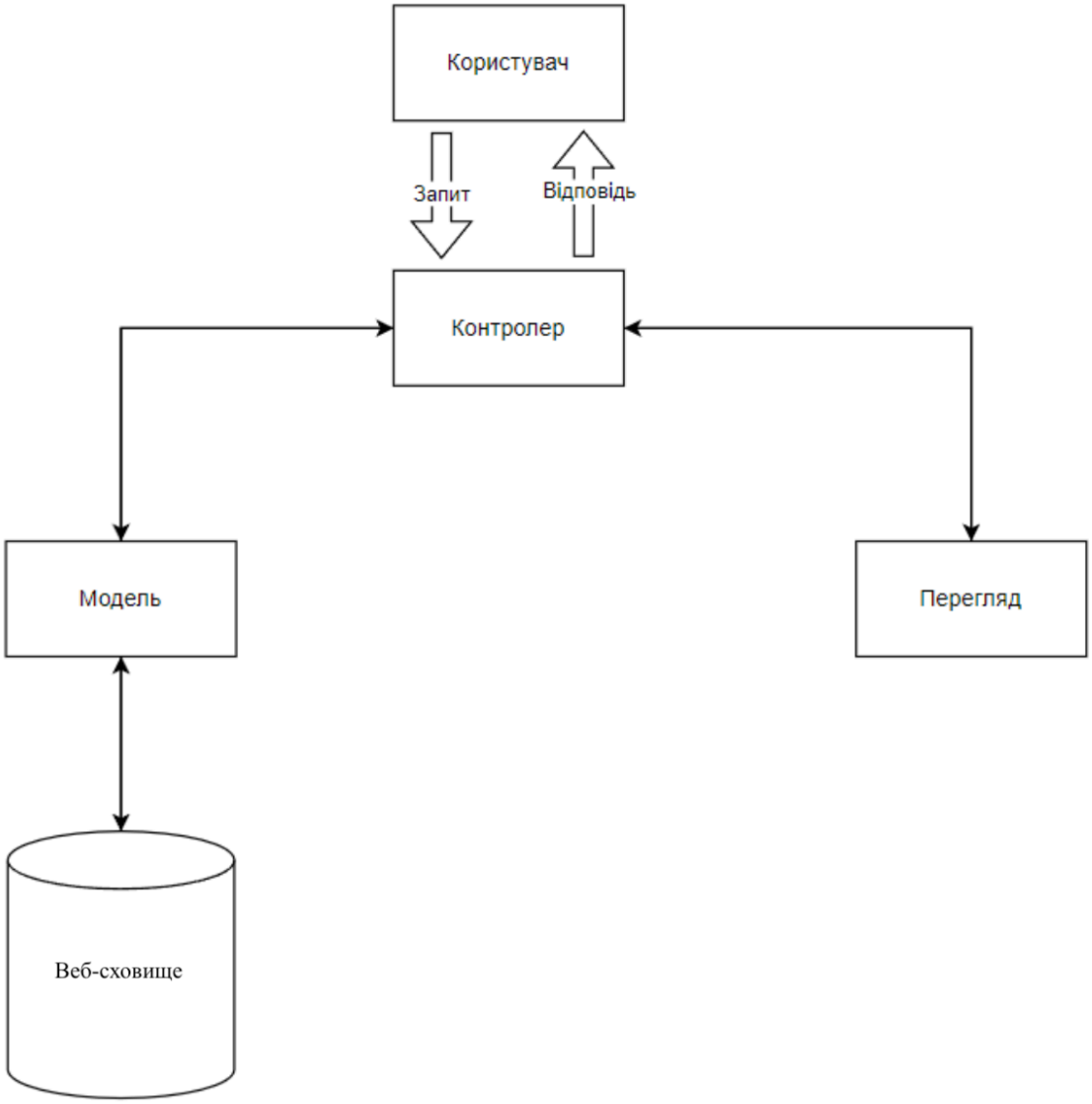
Назва	Додати новий допис
ID	3
Опис	Користувач може зайти на тему та додати допис до цієї теми
Актори	Адміністратор, користувач
Вигоди компанії	Регулярні нові дописи стимулюють постійні обговорення та взаємодію між користувачами, що підтримує активність на форумі.
Частота користування	Постійно
Тригери	Користувачу треба натиснути add post
Передумови	Користувачу треба зайти на тему та натиснути add post та заповнити поле
Постумови	Користувач додає новий допис до обраної теми
Основний розвиток	Спочатку адміністратор, або користувач натискає на тему, потім його перекине на сторінку з цією темою де можна додати допис до обраної теми
Альтернативні розвитки	–
Виняткові ситуації	–

Назва	Видалити тему
ID	4
Опис	Адміністратор натиснувши кнопку delete може видалити тему навпроти якої була кнопка delete
Актори	Адміністратор
Вигоди компанії	Видалення непридатних або застарілих тем допомагає підтримувати високий рівень якості контенту на форумі, роблячи його більш корисним та релевантним для користувачів.
Частота користування	Постійно
Тригери	Адміністратор треба натиснути delete
Передумови	Адміністратору треба зайти на список тем та натиснути delete
Постумови	Адміністратору видаляє тему зі списку
Основний розвиток	Маємо зайти від акаунта Адміністратор і ми потрапляємо на список тем, якщо тем не має, то ми можемо їх додати, потім натиснувши на delete видалиться тема
Альтернативні розвитки	–
Виняткові ситуації	–

Назва	Редагування теми
ID	5
Опис	Адміністратор має можливість редагувати інформацію про теми
Актори	Адміністратор
Вигоди компанії	Редагування тем дозволяє адаптувати контент до потреб та запитів користувачів, що сприяє підвищенню задоволення від користування форумом.
Частота користування	Часто
Тригери	Адміністратор натискає на edit
Передумови	Після створення теми, у адміністратора з’являється можливість редагувати будь-яку інформацію про тему
Постумови	Оновлена інформація зберігається і рендириться
Основний розвиток	Адміністратор редагує обрану тему, натискає на кнопку save, після чого тема вважається відредагованою і всі зміни зберігаються
Альтернативні розвитки	
Виняткові ситуації	

3 АРХІТЕКТУРА СИСТЕМИ

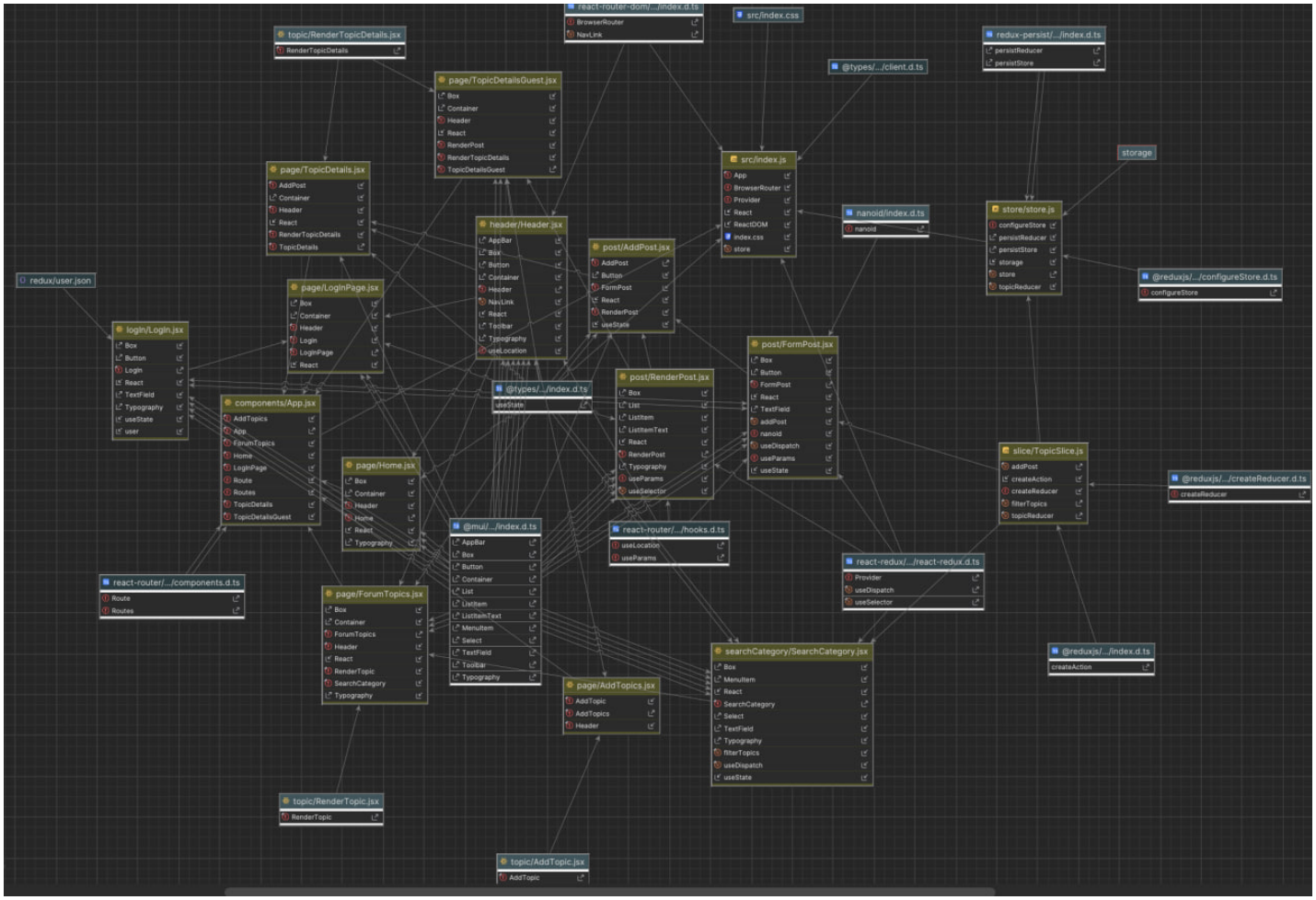
Загальна архітектура системи наведена на рис. 3.1



4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

Загальна реалізація всіх компонентів системи зображена на рис. 4.0

Рисунок 4.0 – Загальна реалізація всіх компонентів системи



4.1 Загальна структура проекту

Загальна структура проекту представлена на рис.4.1

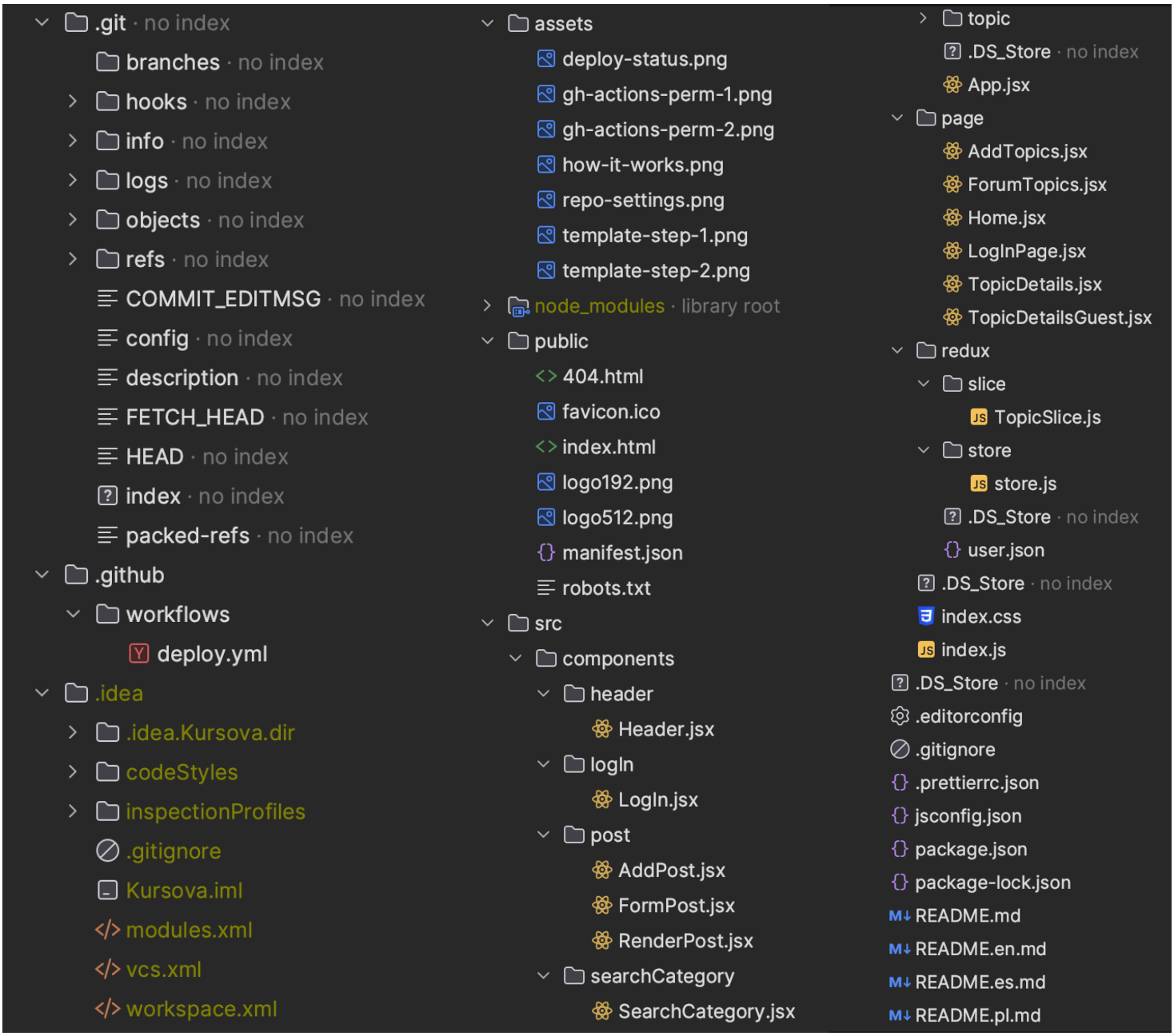


Рисунок 4.1 – Загальна структура проекту

Проект складається з веб-ресурсів, бібліотек та вихідного коду, який, в свою чергу, можна поділити на компоненти рівня доступу до даних, компоненти бізнес-логіки та веб-компоненти. Веб-ресурси включають HTML, CSS та

JavaScript файли, які забезпечують інтерфейс користувача та взаємодію з сервером. Бібліотеки, такі як React, Redux, та інші, забезпечують основу для створення динамічних інтерфейсів та управління станом додатку.

4.2 Компоненти рівня доступу до даних

Основні сутності та інтерфейси рівня доступу до даних наведені на рис. 4.2

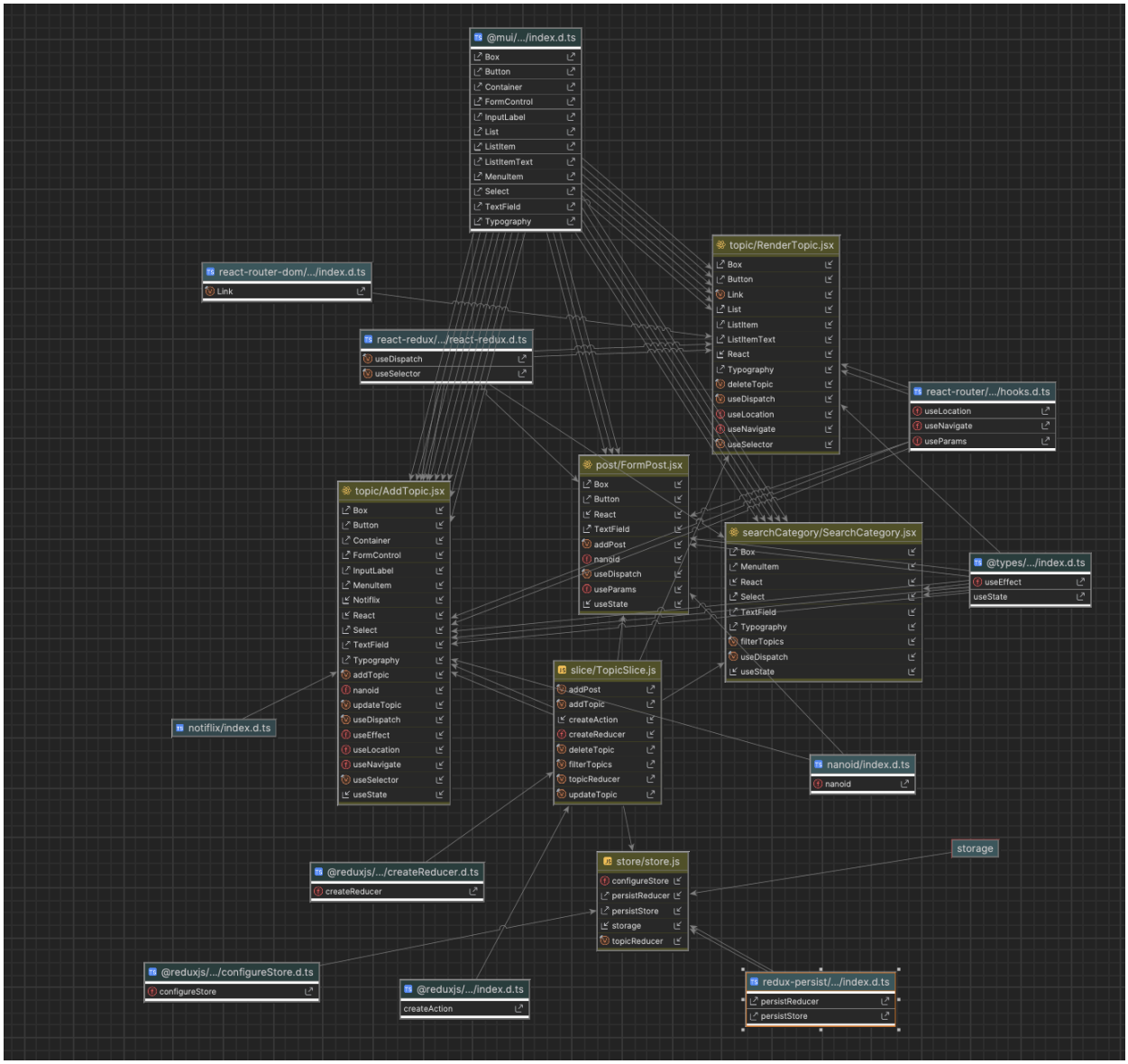


Рисунок 4.2 – Основні сутності та інтерфейси рівня доступу до даних

Сутність мого прокату - це форум. Додавання, та перегляд тем. Акторами є адміністратор, користувач та гість кожна тема має свою унікальну назву за якою потім її можна буде з легкістю знайти

4.3 Компоненти рівня бізнес-логіки

Основні компоненти рівня бізнес-логіки наведені на рис. 4.3

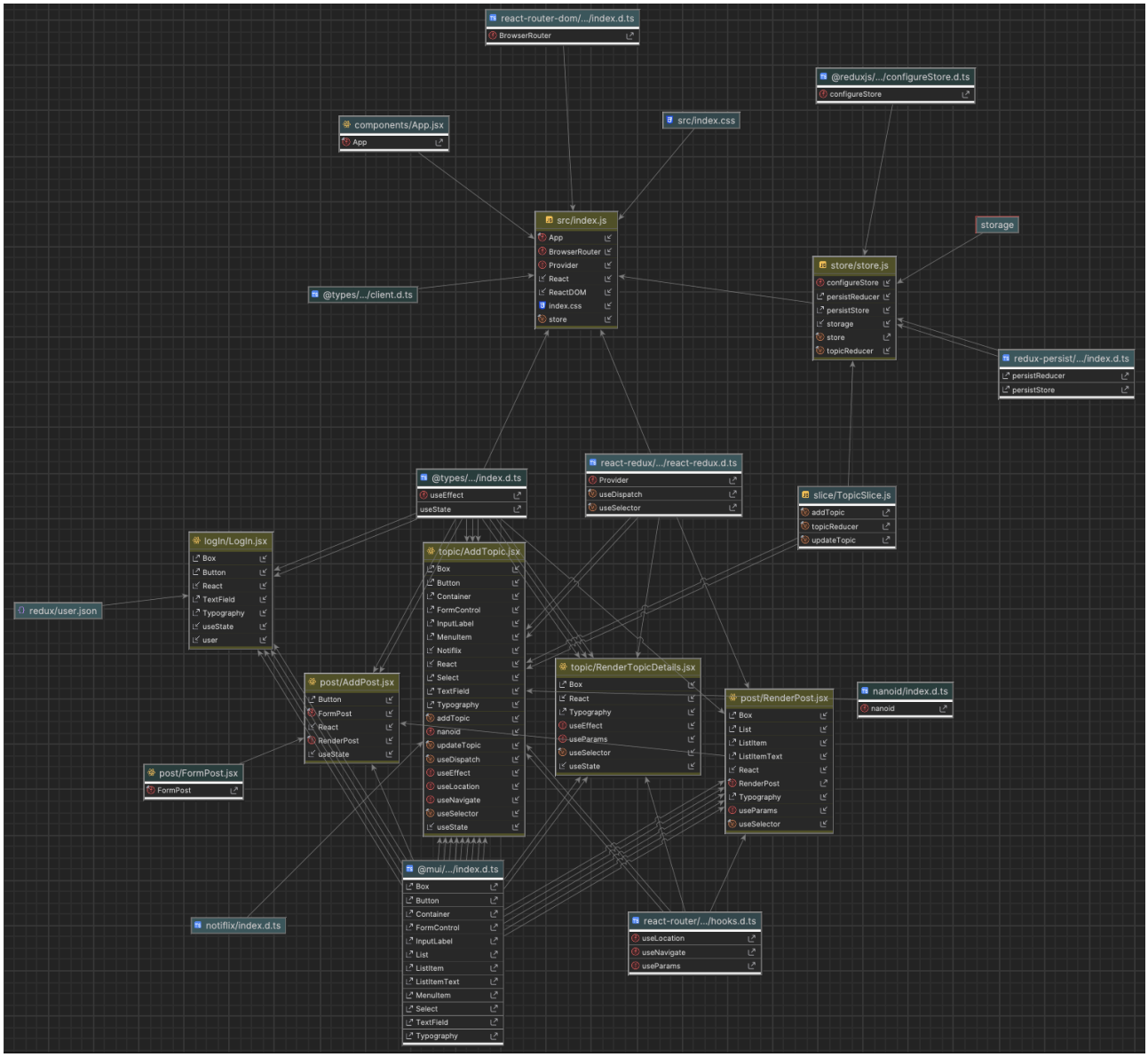
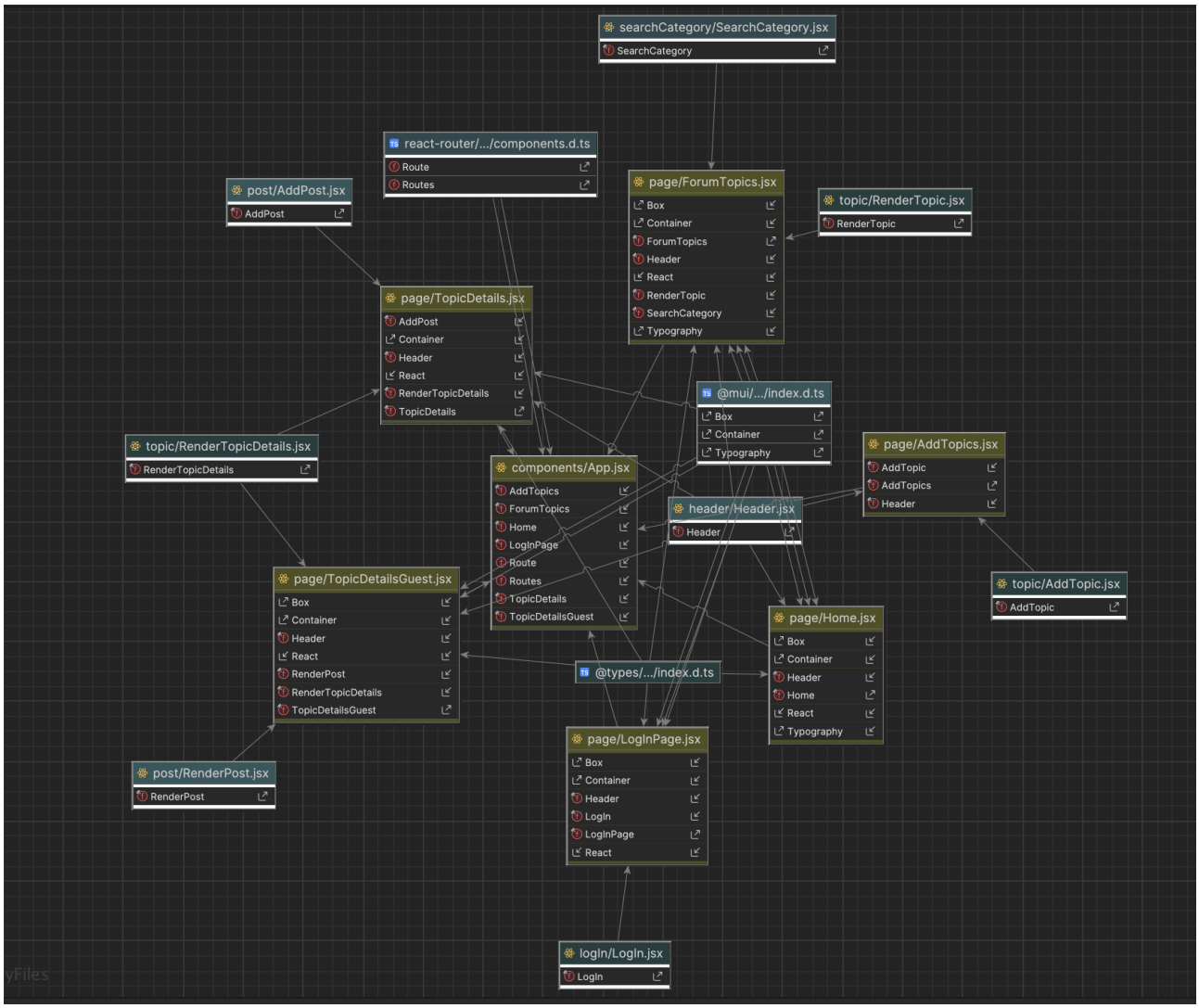


Рисунок 4.3 – Компоненти рівня бізнес-логіки

4.4 Компоненти рівня інтерфейсу користувача



ВИСНОВКИ

Під час розробки проекту форуму було визначено основні сутності системи, такі як тема і допис, а також три типи користувачів: адміністратор, зареєстрований користувач і гість. Було проаналізовано вимоги до системи та визначено функціональні та нефункціональні вимоги, що визначають очікувану поведінку системи. Основні функціональні вимоги включають можливість гостя переглядати теми та дописи, зареєстрованого користувача – створювати нові дописи та переглядати існуючі теми й дописи, а адміністратора – створювати, редагувати та видаляти теми. Наступним кроком було обрано технології для реалізації проекту. Було обрано React, який забезпечує швидку та ефективну розробку інтерфейсу користувача. Redux було використано для управління станом додатка, забезпечуючи централізоване зберігання даних та полегшуючи їхню синхронізацію між компонентами. Material-UI було обрано для побудови інтерфейсу користувача завдяки його зручності та широкому набору готових компонентів. Архітектура проекту була розроблена з урахуванням принципів модульності та масштабованості. Проект складається з веб-ресурсів, бібліотек, та вихідного коду, який можна поділити на компоненти рівня доступу до даних, компоненти бізнес-логіки та веб-компоненти. Основні компоненти включають `AddTopic`, `RenderTopic`, `FormPost`, які забезпечують основний функціонал форуму. Завдяки використанню сучасних бібліотек та фреймворків, таких як `react-redux`, `redux-persist`, `nanoid` та `notiflix`, вдалося забезпечити стабільність і надійність роботи форуму. `React-router-dom` було використано для організації маршрутизації в додатку, що дозволяє користувачам легко переміщуватися між різними сторінками. Використання `redux-toolkit` спростило процес налаштування Redux, зменшивши обсяг шаблонного коду. Було розроблено загальну архітектуру системи, яка передбачає розподіл на логічні рівні. Це забезпечило гнучкість та можливість подальшого розширення функціоналу системи. ER-модель була розроблена для забезпечення зручного управління даними. Завдяки цьому, кожна тема та допис мають унікальний ідентифікатор, що полегшує їх пошук і маніпуляції з ними.

У результаті розробки було створено гнучку та легко налаштовувану систему, яка відповідає всім визначеним вимогам. Основні переваги створеної системи включають простоту налаштування, можливість легко розширювати функціонал та зручний графічний інтерфейс. Система має потенціал до розвитку завдяки своїй відкритій архітектурі та використанню сучасних технологій.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Redux-toolkit: https://redux-toolkit.js.org
2. Redux-persist: https://www.npmjs.com/package/redux-persist
3. Nanoid: https://www.npmjs.com/package/nanoid
4. Notiflix: https://www.npmjs.com/package/notiflix
5. React: https://uk.legacy.reactjs.org
6. JavaScript: https://developer.mozilla.org/en-US/docs/Web/JavaScript

ДОДАТОК А

Лістінг програми

https://github.com/maxvovchok/forum

Header.jsx

```
import React from 'react';
import { NavLink, useLocation } from 'react-router-dom';
import {
  AppBar,
  Toolbar,
  Button,
  Typography,
  Box,
  Container,
} from '@mui/material';

export const Header = () => {
  const location = useLocation();
  const isForum = location.pathname === '/' || location.pathname
=== '/login';
  const isAdmin = location.pathname === '/administrator';
  const isHome =
    location.pathname === '/administrator/addtopic' ||
    location.pathname.includes('/topic');
  const isLogin =
    location.pathname === '/' ||
    location.pathname === '/login' ||
    location.pathname.includes('/guest');

  return (
    <Box sx={{ width: '100%', marginBottom: 4 }}>
      <AppBar position="static">
        <Container maxWidth="lg">
          <Toolbar disableGutters>
            <Box sx={{ display: 'flex', alignItems: 'center' }}>
              <Typography variant="h6" component="div">
                Forum
              </Typography>
            </Box>
            <Box sx={{ width: '20px' }} />{' '}
            <Box sx={{ display: 'flex', alignItems: 'center' }}>
              {isForum && (
                <Button component={NavLink} to="/guest"
color="inherit">
                  Continue as Guest
                </Button>
              )}
              {isHome && (
```

```
                <Button
                  component={NavLink}
                  to={
                    location.pathname.includes('/guest')
                      ? '/guest'
                      : location.pathname.includes('/user')
                      ? '/user'
                      : '/administrator'
                  }
                  color="inherit"
                >
                  Home
                </Button>
              )}
```

```
              {isAdmin && (
                <Button
                  component={NavLink}
                  to="/administrator/addtopic"
                  color="inherit"
                >
                  Add topic
                </Button>
              )}
            </Box>
            <Box sx={{ flexGrow: 1 }} />{' '}
            <Box sx={{ display: 'flex', alignItems: 'center' }}>
              {isLogin ? (
                <Button component={NavLink} to="/login"
color="inherit">
                  Log in
                </Button>
              ) : (
                <Button
                  component={NavLink}
                  to="/"
                  color="error"
                  variant="contained"
                >
                  Log out
                </Button>
              )}
            </Box>
          </Toolbar>
        </Container>
      </AppBar>
    </Box>
  );
};
```

LogIn.jsx

```jsx
import React, { useState } from 'react';
import { TextField, Button, Typography, Box } from '@mui/
material';
import user from '../../redux/user.json';

export const LogIn = () => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [message, setMessage] = useState('');

  const handleLogin = () => {
    const isAdmin =
      user.administrator.email === email &&
      user.administrator.password === password;
    const isUser = user.user.email === email && user.user.password
=== password;

    if (isAdmin) {
      window.location.href = 'administrator';
    } else if (isUser) {
      window.location.href = 'user';
    } else {
      setMessage('Invalid email or password');
    }
  };

  return (
    <Box
      sx={{
        display: 'flex',
        flexDirection: 'column',
        alignItems: 'center',
        justifyContent: 'center',
        minHeight: '100vh',
      }}
    >
      <Typography variant="h4" gutterBottom>
        Log In
      </Typography>
      <TextField
        label="Email"
        type="email"
        value={email}
        onChange={e => setEmail(e.target.value)}
        required
        fullWidth
```

```
        margin="normal"
      />
      <TextField
        label="Password"
        type="password"
        value={password}
        onChange={e => setPassword(e.target.value)}
        required
        fullWidth
        margin="normal"
      />
      <Button variant="contained" onClick={handleLogin} fullWidth>
        LOG IN
      </Button>
      <Typography color="error" variant="body1" gutterBottom>
        {message}
      </Typography>
    </Box>
  );
};
```

AddPost.jsx

```jsx
import React, { useState } from 'react';
import { Button } from '@mui/material';
import { FormPost } from './FormPost';
import { RenderPost } from './RenderPost';

export const AddPost = () => {
  const [showForm, setShowForm] = useState(false);

  const addPost = () => {
    setShowForm(true);
  };

  const hideForm = () => {
    setShowForm(false);
  };

  return (
    <div style={{ marginTop: '20px' }}>
      {!showForm && (
        <Button
          variant="contained"
          onClick={addPost}
          sx={{ marginBottom: '30px' }}
        >
          Add a post
        </Button>
      )}
      {showForm && <FormPost hideForm={hideForm} />}
      <RenderPost />
    </div>
  );
};
```

FormPost.jsx

```jsx
import React, { useState } from 'react';
import { useDispatch } from 'react-redux';
import { nanoid } from 'nanoid';
import { addPost } from '../../redux/slice/TopicSlice';
import { useParams } from 'react-router-dom';
import { TextField, Button, Box } from '@mui/material';

export const FormPost = ({ hideForm }) => {
  const [post, setPost] = useState('');
  const dispatch = useDispatch();
  const { Id } = useParams();

  const handleSubmit = e => {
    e.preventDefault();

    const newPost = {
      Id: nanoid(),
      content: post,
    };

    dispatch(addPost({ Id, post: newPost }));

    setPost('');
    hideForm();
  };

  const handleChange = e => {
    const value = e.currentTarget.value;
    setPost(value);
  };

  return (
    <div>
      <form onSubmit={handleSubmit}>
        <Box display="flex" alignItems="center"
justifyContent="center" mb={2}>
          <TextField
            type="text"
            name="post"
            label="Enter your post"
            variant="outlined"
            onChange={handleChange}
            value={post}
            fullWidth
            sx={{ mr: 2 }}
          />
```

```
          <Button type="submit" variant="contained"
color="primary">
            Publish
          </Button>
        </Box>
      </form>
    </div>
  );
};
```

RenderPost.jsx

```jsx
import React from 'react';
import { useSelector } from 'react-redux';
import { useParams } from 'react-router-dom';
import { Typography, List, ListItem, ListItemText, Box } from
'@mui/material';

export const RenderPost = () => {
  const { Id } = useParams();
  const allTopic = useSelector(state => state.topics.topicsArr);
  const topic = allTopic.find(t => t.id === Id);

  return (
    <Box mt={4}>
      {topic && topic.post.length > 0 ? (
        <>
          <Typography variant="h5" gutterBottom>
            All posts
          </Typography>
          <List>
            {topic.post.map(({ content, id }, index) => (
              <ListItem
                key={index}
                sx={{
                  border: '1px solid #ccc',
                  borderRadius: '8px',
                  marginBottom: '10px',
                  padding: '10px',
                  backgroundColor: '#f9f9f9',
                }}
              >
                <ListItemText primary={content} />
              </ListItem>
            ))}
          </List>
        </>
      ) : (
        <Typography variant="h6">There are no posts</Typography>
      )}
    </Box>
  );
};
```

SearchCategory.jsx

```
import React, { useState } from 'react';
import { useDispatch } from 'react-redux';
import { filterTopics } from '../../redux/slice/TopicSlice';
import { Typography, Select, MenuItem, TextField, Box } from
'@mui/material';

export const SearchCategory = () => {
  const [selectedCategory, setSelectedCategory] = useState('');
  const [searchTerm, setSearchTerm] = useState('');

  const dispatch = useDispatch();

  const handleCategoryChange = e => {
    setSelectedCategory(e.target.value);
    dispatch(filterTopics({ category: e.target.value,
searchTerm }));
  };

  const handleSearchTermChange = e => {
    setSearchTerm(e.target.value);
    console.log({ category: selectedCategory, searchTerm:
e.target.value });
    dispatch(
      filterTopics({ category: selectedCategory, searchTerm:
e.target.value })
    );
  };

  return (
    <Box>
      <Typography>Choose by categories</Typography>
      <Select
        value={selectedCategory}
        onChange={handleCategoryChange}
        sx={{ marginBottom: '20px', width: '100%' }}
      >
        <MenuItem value="">All Categories</MenuItem>
        <MenuItem value="Programming">Programming</MenuItem>
        <MenuItem value="Announcements and news">
          Announcements and news
        </MenuItem>
        <MenuItem value="General discussions">General
discussions</MenuItem>
        <MenuItem value="Healthy Lifestyle">Healthy Lifestyle</
MenuItem>
      </Select>
      <Typography>Search</Typography>
      <TextField
        multiline
        rows={4}
```

```
        variant="outlined"
        sx={{ width: '100%', marginBottom: '70px' }}
        type="text"
        value={searchTerm}
        onChange={handleSearchTermChange}
      />
    </Box>
  );
};
```

AddTopic.jsx

```
import React, { useState, useEffect } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { useNavigate, useLocation } from 'react-router-dom';
import Notiflix from 'notiflix';
import { nanoid } from 'nanoid';
import {
  Typography,
  TextField,
  Button,
  Container,
  MenuItem,
  Select,
  FormControl,
  InputLabel,
  Box,
} from '@mui/material';
import { addTopic, updateTopic } from '../../redux/slice/
TopicSlice';

export const AddTopic = () => {
  const location = useLocation();
  const topicToEdit = location.state?.topic || null;

  const [addNameTopic, setAddNameTopic] = useState('');
  const [descriptionTopic, setDescriptionTopic] = useState('');
  const [category, setCategory] = useState('');

  useEffect(() => {
    if (topicToEdit) {
      setAddNameTopic(topicToEdit.nameTopic);
      setDescriptionTopic(topicToEdit.descriptionTopic);
      setCategory(topicToEdit.category);
    }
  }, [topicToEdit]);

  const allTopics = useSelector(state => state.topics.topicsArr);
  const dispatch = useDispatch();
  const navigate = useNavigate();

  const handleSubmit = e => {
    e.preventDefault();

    const isTopicExists = allTopics.some(
      topic => topic.nameTopic === addNameTopic && topic.id !==
topicToEdit?.id
    );

    if (isTopicExists) {
```

```
      Notiflix.Notify.failure('The topic already exists');
    } else {
      const newTopic = {
        id: topicToEdit ? topicToEdit.id : nanoid(),
        nameTopic: addNameTopic,
        descriptionTopic,
        category,
        post: topicToEdit ? topicToEdit.post : [],
      };

      if (topicToEdit) {
        dispatch(updateTopic(newTopic));
        Notiflix.Notify.success('Topic edited successfully');
      } else {
        dispatch(addTopic(newTopic));
        Notiflix.Notify.success('Topic successfully added');
      }

      setAddNameTopic('');
      setDescriptionTopic('');
      setCategory('');

      navigate('/administrator');
    }
  };

  const handleChange = e => {
    const { name, value } = e.target;

    switch (name) {
      case 'nameTopic':
        setAddNameTopic(value);
        break;

      case 'descriptionTopic':
        setDescriptionTopic(value);
        break;

      case 'category':
        setCategory(value);
        break;

      default:
        break;
    }
  };

  return (
    <Container>
      <Typography variant="h4" component="h2" gutterBottom>
        {topicToEdit ? 'Edit Topic' : 'Add a New Topic'}
      </Typography>
```

```
      <form onSubmit={handleSubmit}>
        <Box display="flex" flexDirection="column" gap={2}>
          <TextField
            label="Topic Name"
            name="nameTopic"
            value={addNameTopic}
            onChange={handleChange}
            fullWidth
            required
          />
          <TextField
            label="Description"
            name="descriptionTopic"
            value={descriptionTopic}
            onChange={handleChange}
            fullWidth
            required
          />
          <FormControl fullWidth>
            <InputLabel>Category</InputLabel>
            <Select
              name="category"
              value={category}
              onChange={handleChange}
              required
            >
              <MenuItem value="Programming">Programming</MenuItem>
              <MenuItem value="Announcements and news">
                Announcements and News
              </MenuItem>
              <MenuItem value="General discussions">
                General Discussions
              </MenuItem>
              <MenuItem value="Healthy Lifestyle">Healthy
Lifestyle</MenuItem>
            </Select>
          </FormControl>
          <Button
            type="submit"
            variant="contained"
            color="primary"
            sx={{ width: '100px' }}
          >
            {topicToEdit ? 'Save' : 'Publish'}
          </Button>
        </Box>
      </form>
    </Container>
  );
};
```

RenderTopic.jsx

```jsx
import React from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { Link, useLocation, useNavigate } from 'react-router-dom';
import { deleteTopic } from '../../redux/slice/TopicSlice';
import {
  Typography,
  Button,
  ListItem,
  ListItemText,
  List,
  Box,
} from '@mui/material';

export const RenderTopic = () => {
  const filteredTopics = useSelector(state =>
state.topics.filteredTopics);
  const location = useLocation();
  const isAdminPage = location.pathname.startsWith('/
administrator');
  const isUserPage = location.pathname.startsWith('/user');
  const dispatch = useDispatch();
  const navigate = useNavigate();

  const handleDelete = id => {
    dispatch(deleteTopic(id));
  };

  const handleEdit = topic => {
    navigate('/administrator/addtopic', { state: { topic } });
  };

  return (
    <Box mr={2} mt={-4}>
      {filteredTopics.length > 0 ? (
        <>
          <Typography variant="h5" gutterBottom>
            All Topics
          </Typography>
          <List>
            {filteredTopics.map(
              ({ id, nameTopic, descriptionTopic, category }) => (
                <Box
                  key={id}
                  sx={{
                    border: '1px solid #ccc',
                    borderRadius: '8px',
                    marginBottom: '10px',
                    padding: '10px',
                  }}
```

```
                >
                  <ListItem disablePadding>
                    <ListItemText sx={{ '& > *': { marginBottom:
'8px' } }}>
                      <Link
                        to={
                          isAdminPage
                            ? `/administrator/topic/${id}`
                            : isUserPage
                            ? `/user/topic/${id}`
                            : `/guest/topic/${id}`
                        }
                        style={{ textDecoration: 'none', color:
'inherit' }}
                      >
                        <Typography variant="h6" sx={{ fontWeight:
'bold' }}>
                          {nameTopic}
                        </Typography>
                        <Typography
                          variant="body1"
                          sx={{ marginTop: '8px', marginBottom:
'4px' }}
                        >
                          {descriptionTopic}
                        </Typography>
                        <Typography variant="subtitle2"
sx={{ color: '#666' }}>
                          Category: {category}
                        </Typography>
                      </Link>
                    </ListItemText>
                    {isAdminPage && (
                      <>
                        <Button
                          onClick={() =>
                            handleEdit({
                              id,
                              nameTopic,
                              descriptionTopic,
                              category,
                            })
                          }
                          variant="outlined"
                          color="primary"
                          sx={{ height: '100%', marginRight: '8px'
}}
                        >
                          Edit
                        </Button>

                        <Button
```

```
                        onClick={() => handleDelete(id)}
                        variant="outlined"
                        color="error"
                        sx={{ height: '100%' }}
                      >
                        Delete
                      </Button>
                    </>
                  )}
                </ListItem>
              </Box>
            )
          )}
        </List>
      </>
    ) : (
      <Typography variant="h6">There are no topics</Typography>
    )}
  </Box>
 );
};
```

RenderTopicDetails.jsx

```jsx
import React, { useState, useEffect } from 'react';
import { useSelector } from 'react-redux';
import { useParams } from 'react-router-dom';
import { Typography, Box } from '@mui/material';

export const RenderTopicDetails = () => {
  const [renderName, setRenderName] = useState('');
  const [renderDescriptionTopic, setRenderDescriptionTopic] =
useState('');
  const allTopic = useSelector(state => state.topics.topicsArr);
  const params = useParams();

  useEffect(() => {
    function getTopic() {
      const topic = allTopic.find(idTopic => idTopic.id ===
params.Id);
      if (topic) {
        setRenderName(topic.nameTopic);
        setRenderDescriptionTopic(topic.descriptionTopic);
      }
    }
    getTopic();
  }, [params.Id, allTopic]);

  return (
    <Box p={2}>
      <Typography variant="h4" sx={{ marginBottom: '16px' }}>
        {renderName}
      </Typography>
      <Typography>{renderDescriptionTopic}</Typography>
    </Box>
  );
};
```

App.jsx

```
import { Route, Routes } from 'react-router-dom';
import { Home } from 'page/Home';
import { ForumTopics } from 'page/ForumTopics';
import { AddTopics } from '../page/AddTopics';
import { TopicDetails } from 'page/TopicDetails';
import { TopicDetailsGuest } from 'page/TopicDetailsGuest';
import { LogInPage } from 'page/LogInPage';

export const App = () => {
  return (
    <div>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/administrator" element={<ForumTopics />} />
        <Route path="/administrator/addtopic"
element={<AddTopics />} />
        <Route path="/administrator/topic/:Id"
element={<TopicDetails />} />
        <Route path="/login" element={<LogInPage />} />

        <Route path="/user" element={<ForumTopics />} />
        <Route path="/user/topic/:Id" element={<TopicDetails />} /
>
        <Route path="/guest" element={<ForumTopics />} />
        <Route path="/guest/topic/:Id" element={<TopicDetailsGuest
/>} />
      </Routes>
    </div>
  );
};
```

AddTopics.jsx

```jsx
import { Header } from 'components/header/Header';
import { AddTopic } from 'components/topic/AddTopic';

export const AddTopics = () => {
  return (
    <div>
      <Header />
      <AddTopic />
    </div>
  );
};
```

ForumTopics.jsx

```
import React from 'react';
import { Header } from 'components/header/Header';
import { SearchCategory } from 'components/searchCategory/
SearchCategory';
import { RenderTopic } from '../components/topic/RenderTopic';
import { Container, Box, Typography } from '@mui/material';

export const ForumTopics = () => {
  return (
    <Box
      display="flex"
      flexDirection="column"
      alignItems="center"
      minHeight="100vh"
    >
      <Header />
      <Container maxWidth="md" sx={{ width: '600px', minHeight:
'400px' }}>
        <Box mt={4} mb={4}>
          <Typography variant="h6" align="center" gutterBottom>
            Topics
          </Typography>
          <SearchCategory />
        </Box>
        <RenderTopic />
      </Container>
    </Box>
  );
};
```

Home.jsx

```jsx
import React from 'react';
import { Header } from 'components/header/Header';
import { Container, Typography, Box } from '@mui/material';

export const Home = () => {
  return (
    <Box
      display="flex"
      flexDirection="column"
      alignItems="center"
      minHeight="100vh"
    >
      <Header />
      <Container maxWidth="md" sx={{ textAlign: 'center', mt: 4 }}
>
        <Typography variant="h1" component="h1" gutterBottom>
          Welcome to the forum!
        </Typography>
        <Typography variant="body1">
          You must be logged in or continue as a guest to
continue.
        </Typography>
      </Container>
    </Box>
  );
};
```

LogInPage.jsx

```jsx
import React from 'react';
import { Header } from 'components/header/Header';
import { LogIn } from 'components/logIn/LogIn';
import { Container, Box } from '@mui/material';

export const LogInPage = () => {
  return (
    <Box
      display="flex"
      flexDirection="column"
      alignItems="center"
      minHeight="100vh"
    >
      <Header />
      <Container
        maxWidth="md"
        sx={{ marginTop: '-180px', width: '400px', height: '400px'
}}
      >
        <LogIn />
      </Container>
    </Box>
  );
};
```

TopicDetails.jsx

```jsx
import React from 'react';
import { RenderTopicDetails } from 'components/topic/
RenderTopicDetails';
import { AddPost } from 'components/post/AddPost';
import { Header } from 'components/header/Header';
import { Container } from '@mui/material';

export const TopicDetails = () => {
  return (
    <div>
      <Header />
      <Container maxWidth="md" sx={{ mt: 4 }}>
        <RenderTopicDetails />
        <AddPost />
      </Container>
    </div>
  );
};
```

TopicDetailsGuest.jsx

```jsx
import React from 'react';
import { RenderPost } from 'components/post/RenderPost';
import { RenderTopicDetails } from 'components/topic/
RenderTopicDetails';
import { Header } from 'components/header/Header';
import { Container, Box } from '@mui/material';

export const TopicDetailsGuest = () => {
  return (
    <Box sx={{ minHeight: '100vh', display: 'flex', flexDirection:
'column' }}>
      <Header />
      <Container maxWidth="lg" sx={{ flex: 1, mt: 4 }}>
        <RenderTopicDetails />
        <RenderPost />
      </Container>
    </Box>
  );
};
```

topicReducer.js

```
import { createAction, createReducer } from '@reduxjs/toolkit';

const initialState = {
  topicsArr: [],
  filteredTopics: [],
};

export const getTopic = createAction('get');
export const addTopic = createAction('add');
export const updateTopic = createAction('update');
export const addPost = createAction('addPost');
export const filterTopics = createAction('filter');
export const deleteTopic = createAction('deleteTopic');
export const deletePost = createAction('deletePost');

export const topicReducer = createReducer(initialState, builder =>
{
  builder.addCase(addTopic, (state, action) => {
    state.topicsArr.push(action.payload);
    state.filteredTopics = state.topicsArr;
  });
  builder.addCase(updateTopic, (state, action) => {
    const index = state.topicsArr.findIndex(
      topic => topic.id === action.payload.id
    );
    if (index !== -1) {
      state.topicsArr[index] = {
        ...state.topicsArr[index],
        ...action.payload,
        post: state.topicsArr[index].post,
      };
    }
    state.filteredTopics = state.topicsArr;
  });
  builder.addCase(addPost, (state, action) => {
    const { Id, post } = action.payload;
    const topic = state.topicsArr.find(t => t.id === Id);
    if (topic) {
      topic.post.push(post);
    }
  });
  builder.addCase(filterTopics, (state, action) => {
    const { category, searchTerm } = action.payload;
    state.filteredTopics = state.topicsArr.filter(topic => {
      const matchesCategory = category ? topic.category ===
category : true;
      const matchesSearchTerm = topic.nameTopic
        .toLowerCase()
        .includes(searchTerm.toLowerCase());
```

```
      return matchesCategory && matchesSearchTerm;
    });
  });
  builder.addCase(deleteTopic, (state, action) => {
    state.topicsArr = state.topicsArr.filter(
      topic => topic.id !== action.payload
    );
    state.filteredTopics = state.topicsArr;
  });
});
```

store.js

```
import { configureStore } from '@reduxjs/toolkit';
import { persistStore, persistReducer } from 'redux-persist';
import storage from 'redux-persist/lib/storage';
import { topicReducer } from '../slice/TopicSlice';

const persistConfig = {
  key: 'root',
  storage,
};

const persistedReducer = persistReducer(persistConfig,
topicReducer);

export const store = configureStore({
  reducer: {
    topics: persistedReducer,
  },
});

export const persistor = persistStore(store);
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import { App } from 'components/App';
import './index.css';
import { BrowserRouter } from 'react-router-dom';
import { store } from './redux/store/store';
import { Provider } from 'react-redux';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <BrowserRouter basename="/forum">
      <Provider store={store}>
        <App />
      </Provider>
    </BrowserRouter>
  </React.StrictMode>
);
```