FORUM PROJECT

CSCB634 Практика по програмиране и по реализация на бази данни
Петко Данчев, F068583

Съдържание

1. Въведение

Този проект представлява форумно приложение с управление на потребители, публикуване на коментари и функции за модериране. Включва ролево-базирано управление на достъпа, за да се управляват различни потребителски роли като Администратор, Модератор и Потребител.

2. Технологии

Бекенд

- Фреймуърк: ASP.NET Core
- Аутентикация: ASP.NET Core Identity
- База данни: Entity Framework Core със SQL Server

Фронтенд

- Фреймуърк: React

- Управление на състоянието: React Hooks
- HTTP клиент: Axios

Машинно Обучение

- Фреймуърк: ML.NET
- Моделиране: Sentiment Analysis

3. Инструкции за настройка

1. Инсталиране на зависимости

Уверете се, че имате инсталиран .NET Core SDK и Node.js.

- Инсталирайте необходимите зависимости за бекенда с командата:

dotnet restore

- Инсталирайте необходимите зависимости за фронтенда с командата:

npm install

2. Настройка на база данни

- Конфигурирайте връзката към базата данни в appsettings.json.
- Изпълнете миграциите на EF Core с командата:

dotnet ef database update

3. Стартиране на проекта

- Стартирайте бекенда с командата:

dotnet run

- Стартирайте фронтенда с командата:

npm start

4. Разбивка на проектите

Бекенд компоненти

1. Контролер за Аутентикация

Контролерът за аутентикация обработва регистрация на потребители, вход, изход и извличане на текущи потребителски данни. Ендпойнтите включват регистрация, вход, изход и получаване на информация за текущия потребител.

2. Контролер за Модератори

Контролерът за модератори обработва извличане на маркирани коментари и одобряването или изтриването им. Ендпойнтите включват извличане на маркирани коментари, одобряване на коментар и изтриване на коментар.

3. Контролер за Администратори

Контролерът за администратори обработва управление на потребители и присвояване на роли. Ендпойнтите включват извличане на потребители и превключване на ролята модератор за потребител.

Фронтенд компоненти

1. Основен Компонент (App Component)

Основният компонент обработва маршрутизацията и състоянието на аутентикацията. Включва логика за проверка на влизане и управление на регистрация и вход.

2. Компонент за Администраторски Панел

Компонентът за администраторски панел управлява потребители и роли и показва панела за модериране на модераторите. Включва логика за извличане на потребители и текущи потребителски данни.

3. Компонент за Модериране

Компонентът за модериране показва маркирани коментари и позволява на модераторите да ги одобряват или изтриват. Включва логика за извличане на маркирани коментари и обработка на одобрение или изтриване.

4. Компонент за Списък с Коментари

Компонентът за списък с коментари показва списък с коментари. Включва логика за извличане на коментари.

5. Компонент за Формуляр за Коментари

Компонентът за формуляр за коментари позволява на потребителите да публикуват нови коментари. Включва логика за публикуване на коментар и анализ на текста за токсичност.

6. Компонент за Вход

Компонентът за вход обработва входа на потребители. Включва логика за вход и обновяване на състоянието на аутентикацията.

7. Компонент за Регистрация

Компонентът за регистрация позволява на нови потребители да се регистрират. Включва логика за регистрация и обработка на грешки.

Компоненти на ML Модела

1. Входен и Изходен Модел

Входният и изходният модел определят схемата за вход и изход на ML модела. Входният модел включва текст и етикет, а изходният модел включва предсказан етикет и оценка.

2. Сервиз за Предсказване

Сервизът за предсказване обработва предсказанията с помощта на предварително обучен ML модел. Включва логика за зареждане на модела и предсказване на резултатите.

3. Контролер за Анализ

Контролерът за анализ предоставя функционалността за предсказвания чрез API ендпойнт. Включва логика за обработка на заявки и връщане на предсказания.

4. Обучение и Преобучение на Модела

Функционалността за обучение и преобучение на ML модела включва логика за зареждане на данни, създаване на тръбопровод за обучение и запазване на модела.

5. Примерни Данни

Примерните данни се използват за обучение и тестване на модела. Включват текстови примери с етикети за токсичност.

5. API endpoints

Контролер за Аутентикация

- POST /api/auth/register: Регистрация на нов потребител.
- POST /api/auth/login: Вход на потребител.
- POST /api/auth/logout: Изход на потребител.
- GET /api/auth/me: Получаване на информация за вписания потребител.

Контролер за Модератори

- GET /api/moderator/flagged: Получаване на маркирани коментари.

- POST /api/moderator/approve/{id}: Одобрение на маркиран коментар.
- POST /api/moderator/delete/{id}: Изтриване на маркиран коментар.

Контролер за Администратори

- GET /api/admin/users: Получаване на всички потребители.
- POST /api/admin/toggleModerator/{id}: Превключване на ролята модератор за потребител.

ML API

- POST /analyze: Анализ на текст за токсичност.

6. Примерен код:

Контролер за Аутентикация

```
[HttpPost(template: "register")]
0 references | Petko Dapchev, 2 days ago | 1 author, 2 changes
public async Task<IActionResult> Register([FromBody] RegisterModel model)
{
    if (!ModelState.IsValid)
        return BadRequest(ModelState);

    var user = new ApplicationUser
    {
        UserName = model.Username,
        Email = model.Email
    };

    var result = await _userManager.CreateAsync(user, model.Password);

    if (result.Succeeded)
    {
        var role:string = string.IsNullOrWhiteSpace(model.Role) ? "User" : model.Role;
        await _userManager.AddToRoleAsync(user, role);
        await _signInManager.SignInAsync(user, isPersistent: false);
        return Ok(new { message = "Registration successful" });
    }

    foreach (var error in result.Errors)
        ModelState.AddModelError(key:string.Empty, error.Description);

    return BadRequest(ModelState);
}

[HttpPost(template: "login")]
0 references | Petko Dapchev, 2 days ago | 1 author, 2 changes
public async Task<IActionResult> Login([FromBody] LoginModel model)
{
    if (!ModelState.IsValid)
        return BadRequest(ModelState);

    var result = await _signInManager.PasswordSignInAsync(model.Username, model.Password, isPersistent:false, lockoutOnFailure:false);

    if (result.Succeeded)
        return Ok(new { message = "Login successful" });

    return Unauthorized(new { message = "Invalid login attempt" });
}

[HttpPost(template: "logout")]
0 references | Petko Dapchev, 2 days ago | 1 author, 2 changes
public async Task<IActionResult> Logout()
{
    await _signInManager.SignOutAsync();
    return Ok(new { message = "Logged out" });
}

[HttpGet(template: "me")]
0 references | Petko Dapchev, 2 days ago | 1 author, 2 changes
public IActionResult Me()
{
    if (!User.Identity.IsAuthenticated)
        return Unauthorized();

    var roles:List<string> = User.FindAll(type:ClaimTypes.Role).Select(role:Claim => role.Value).ToList();

    return Ok(new
    {
        Username = User.Identity.Name,
        Roles = roles
    });
```

Контролер за Модератори

```
[Authorize(Roles = "Moderator, Admin")]
[ApiController]
[Route(template: "api/[controller]")]
1 reference | Petko Dapchev, 1 day ago | 1 author, 3 changes
public class ModeratorController : ControllerBase
{
    private readonly ApplicationDbContext _context;

    0 references | Petko Dapchev, 78 days ago | 1 author, 1 change
    public ModeratorController(ApplicationDbContext context)
    {
        _context = context;
    }

    [HttpGet(template: "flagged")]
    0 references | Petko Dapchev, 10 days ago | 1 author, 1 change
    public async Task<IActionResult> GetFlaggedComments()
    {
        var comments :List<Comment> = await _context.Comments // DbSet<Comment>
            .Where(c :Comment => c.IsFlagged && !c.IsApproved) // IQueryable<Comment>
            .Include(navigationPropertyPath: c :Comment => c.User) // IIncludableQueryable<Comment,ApplicationUser>
            .ToListAsync(); // Task<List<_>>

        return Ok(comments);
    }

    [HttpPost(template: "approve/{id}")]
    0 references | Petko Dapchev, 10 days ago | 1 author, 1 change
    public async Task<IActionResult> ApproveComment(int id)
    {
        var comment = await _context.Comments.FindAsync(id);
        if (comment == null)
            return NotFound();

        comment.IsApproved = true;
        comment.IsFlagged = false;
        await _context.SaveChangesAsync();

        return Ok(new { message = "Comment approved and published." });
    }

    [HttpPost(template: "delete/{id}")]
    0 references | Petko Dapchev, 10 days ago | 1 author, 1 change
    public async Task<IActionResult> DeleteComment(int id)
    {
        var comment = await _context.Comments.FindAsync(id);
        if (comment == null)
            return NotFound();

        _context.Comments.Remove(comment);
        await _context.SaveChangesAsync();

        return Ok(new { message = "Comment deleted." });
    }
}
```

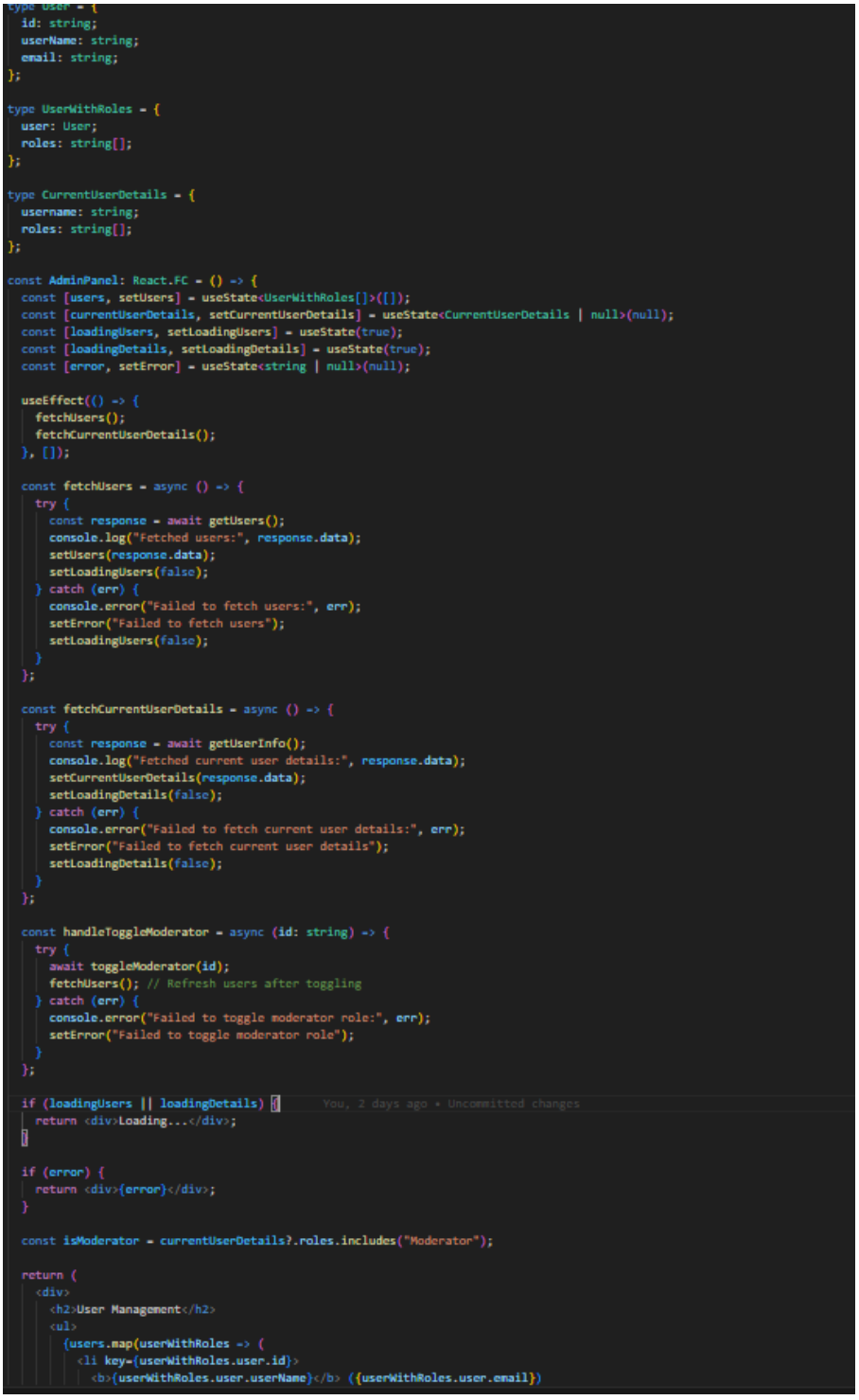
Контролер за Администратори

```
[ApiController]
[Route(template: "api/[controller]")]
1 reference | Petko Dapchev, 1 day ago | 1 author, 3 changes
public class AdminController : ControllerBase
{
    private readonly UserManager<ApplicationUser> _userManager;
    private readonly RoleManager<IdentityRole> _roleManager;

    0 references | Petko Dapchev, 78 days ago | 1 author, 1 change
    public AdminController(UserManager<ApplicationUser> userManager, RoleManager<IdentityRole> roleManager)
    {
        _userManager = userManager;
        _roleManager = roleManager;
    }

    [HttpGet(template: "users")]
    0 references | Petko Dapchev, 1 day ago | 1 author, 3 changes
    public async Task<IActionResult> Users()
    {
        var users :List<ApplicationUser> = _userManager.Users.ToList();
        var userRoles = new List<UserWithRolesViewModel>();

        foreach (var user in users)
        {
            var roles :IList<string> = await _userManager.GetRolesAsync(user);
            userRoles.Add(new UserWithRolesViewModel
            {
                User = user,
                Roles = roles
            });
        }

        return Ok(userRoles);
    }

    [HttpPost(template: "toggleModerator/{id}")]
    0 references | Petko Dapchev, 1 day ago | 1 author, 3 changes
    public async Task<IActionResult> ToggleModerator(string id)
    {
        var user = await _userManager.FindByIdAsync(id);
        if (user == null)
            return NotFound("User not found");

        // Ensure the Moderator role exists
        if (!await _roleManager.RoleExistsAsync(roleName: "Moderator"))
        {
            var result = await _roleManager.CreateAsync(new IdentityRole("Moderator"));
            if (!result.Succeeded)
                return BadRequest(error: "Could not create Moderator role");
        }

        var isModerator = await _userManager.IsInRoleAsync(user, role: "Moderator");

        IdentityResult roleResult;
        if (isModerator)
            roleResult = await _userManager.RemoveFromRoleAsync(user, role: "Moderator");
        else
            roleResult = await _userManager.AddToRoleAsync(user, role: "Moderator");

        if (!roleResult.Succeeded)
            return BadRequest(roleResult.Errors);

        return Ok(new { message = "Role updated successfully" });
    }
}
```

Основен Компонент (App Component)

```tsx
export default function App() {
  const [loggedIn, setLoggedIn] = useState(false);
  const [roles, setRoles] = useState<string[]>([]);
  const [refresh, setRefresh] = useState(0);
  const [showRegister, setShowRegister] = useState(false);
  const [showModeration, setShowModeration] = useState(false);

  useEffect(() => {
    fetchUserInfo(); // Fetch user info on initial load
  }, []);

  const fetchUserInfo = async () => {
    try {
      const response = await api.get("auth/me");
      setLoggedIn(true);
      setRoles(response.data.roles); // Set roles from response
    } catch (error) {
      console.error("Error fetching user info:", error);
      setLoggedIn(false);
    }
  };

  const handleLogin = () => {
    fetchUserInfo(); // Fetch user info after login
  };

  const isModerator = roles.includes("Moderator");
  const isAdmin = roles.includes("Admin");
      You, last week • Demo version …
  if (!loggedIn) {
    return showRegister ? (
      <Register onRegister={() => setShowRegister(false)} />
    ) : (
      <>
        <Login onLogin={handleLogin} />
        <button onClick={() => setShowRegister(true)}>Register</button>
      </>
    );
  }

  return (
    <div>
      <h2>Comments</h2>
      <CommentForm onPosted={() => setRefresh(x => x + 1)} />
      <CommentList key={refresh} />
      <button onClick={() => {
        logout().then(() => setLoggedIn(false));
      }}>Logout</button>
      {isModerator && (
        <div>
          <button onClick={() => setShowModeration(!showModeration)}>
            {showModeration ? "Hide Moderation" : "Show Moderation"}
          </button>
          {showModeration && <Moderation />}
        </div>
      )}
      {isAdmin && <AdminPanel />}
    </div>
  );
}
```

Компонент за Администраторски Панел



```
type User = {
  id: string;
  userName: string;
  email: string;
};

type UserWithRoles = {
  user: User;
  roles: string[];
};

type CurrentUserDetails = {
  username: string;
  roles: string[];
};

const AdminPanel: React.FC = () => {
  const [users, setUsers] = useState<UserWithRoles[]>([]);
  const [currentUserDetails, setCurrentUserDetails] = useState<CurrentUserDetails | null>(null);
  const [loadingUsers, setLoadingUsers] = useState(true);
  const [loadingDetails, setLoadingDetails] = useState(true);
  const [error, setError] = useState<string | null>(null);

  useEffect(() => {
    fetchUsers();
    fetchCurrentUserDetails();
  }, []);

  const fetchUsers = async () => {
    try {
      const response = await getUsers();
      console.log("Fetched users:", response.data);
      setUsers(response.data);
      setLoadingUsers(false);
    } catch (err) {
      console.error("Failed to fetch users:", err);
      setError("Failed to fetch users");
      setLoadingUsers(false);
    }
  };

  const fetchCurrentUserDetails = async () => {
    try {
      const response = await getUserInfo();
      console.log("Fetched current user details:", response.data);
      setCurrentUserDetails(response.data);
      setLoadingDetails(false);
    } catch (err) {
      console.error("Failed to fetch current user details:", err);
      setError("Failed to fetch current user details");
      setLoadingDetails(false);
    }
  };

  const handleToggleModerator = async (id: string) => {
    try {
      await toggleModerator(id);
      fetchUsers(); // Refresh users after toggling
    } catch (err) {
      console.error("Failed to toggle moderator role:", err);
      setError("Failed to toggle moderator role");
    }
  };

  if (loadingUsers || loadingDetails) {      You, 2 days ago • Uncommitted changes
    return <div>Loading...</div>;
  }

  if (error) {
    return <div>{error}</div>;
  }

  const isModerator = currentUserDetails?.roles.includes("Moderator");

  return (
    <div>
      <h2>User Management</h2>
      <ul>
        {users.map(userWithRoles => (
          <li key={userWithRoles.user.id}>
            <b>{userWithRoles.user.userName}</b> ({userWithRoles.user.email})
```

Компонент за Модериране

```tsx
type Comment = {
  id: number;
  content: string;
  user: { userName: string };
  createdAt: string;
};

const Moderation: React.FC = () => {
  const [flaggedComments, setFlaggedComments] = useState<Comment[]>([]);
  const [loading, setLoading] = useState(false);

  useEffect(() => {
    fetchFlaggedComments();
  }, []);

  const fetchFlaggedComments = async () => {
    setLoading(true);
    const response = await getFlaggedComments();      You, last week • Demo version
    setFlaggedComments(response.data);
    setLoading(false);
  };

  const handleApprove = async (id: number) => {
    setLoading(true);
    await approveComment(id);
    fetchFlaggedComments();
  };

  const handleDelete = async (id: number) => {
    setLoading(true);
    await deleteComment(id);
    fetchFlaggedComments();
  };

  return (
    <div>
      <h2>Flagged Comments</h2>
      {loading && <p>Loading...</p>}
      <ul>
        {flaggedComments.map(comment => (
          <li key={comment.id}>
            <b>{comment.user.userName}</b>: {comment.content}
            <div style={{ fontSize: "smaller" }}>{new Date(comment.createdAt).toLocaleString()}</div>
            <button onClick={() => handleApprove(comment.id)}>Approve</button>
            <button onClick={() => handleDelete(comment.id)}>Delete</button>
          </li>
        ))}
      </ul>
    </div>
  );
};

export default Moderation;
```

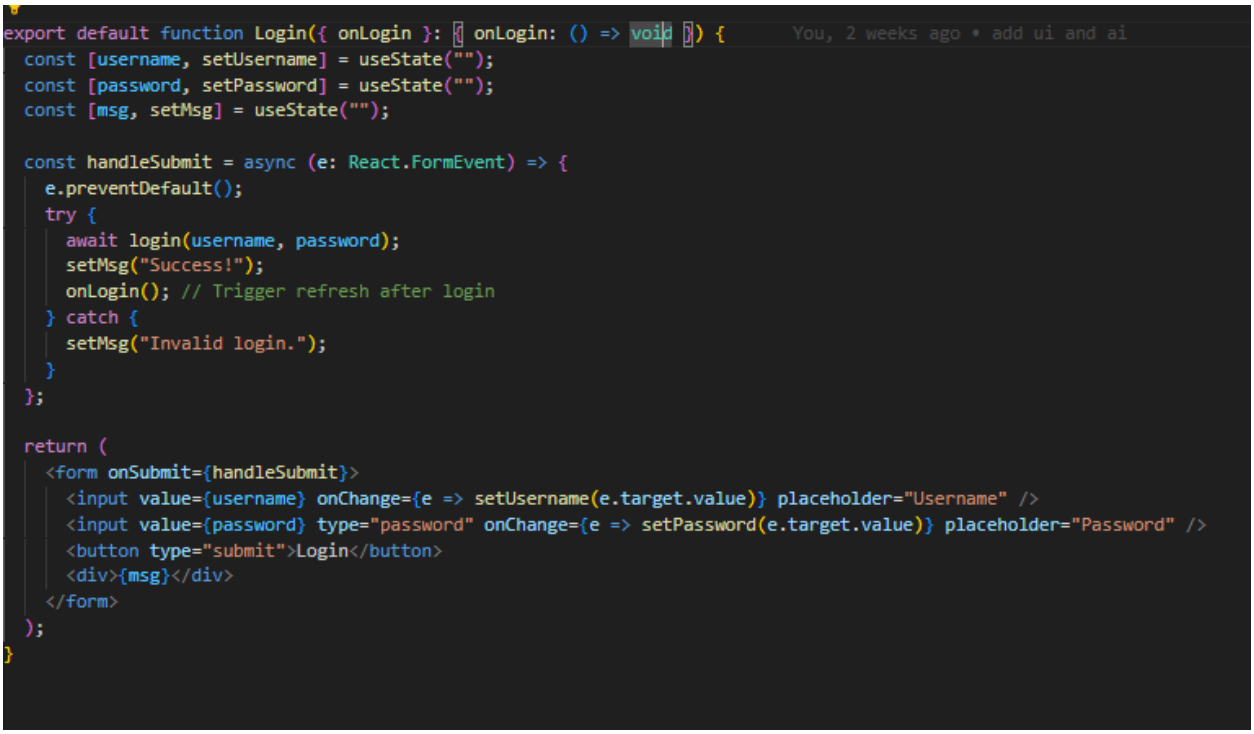
Компонент за Списък с Коментари

```tsx
type Comment = {
  id: number;
  content: string;
  createdAt: string;
  user: string;
};

export default function CommentList() {
  const [comments, setComments] = useState<Comment[]>([]);

  useEffect(() => {
    getComments().then(res => setComments(res.data));      You, 2 weeks ago • add ui and ai
  }, []);

  return (
    <ul>
      {comments.map(c => (
        <li key={c.id}>
          <b>{c.user}:</b> {c.content}
          <div style={{ fontSize: "smaller" }}>{new Date(c.createdAt).toLocaleString()}</div>
        </li>
      ))}
    </ul>
  );
}
```

Компонент за Формуляр за Коментари

```tsx
export default function CommentForm({ onPosted }: { onPosted: () => void }) {
  const [content, setContent] = useState("");
  const [msg, setMsg] = useState("");

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    try {
      const res = await postComment(content);
      setMsg(res.data.message);
      setContent("");
      onPosted(); // refresh comments
    } catch {
      setMsg("Failed to post comment.");
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <textarea value={content} onChange={e => setContent(e.target.value)} />
      <button type="submit">Post</button>
      <div>{msg}</div>
    </form>
  );
```

Компонент за Вход



```tsx
export default function Login({ onLogin }: { onLogin: () => void }) {      You, 2 weeks ago • add ui and ai
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [msg, setMsg] = useState("");

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    try {
      await login(username, password);
      setMsg("Success!");
      onLogin(); // Trigger refresh after login
    } catch {
      setMsg("Invalid login.");
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <input value={username} onChange={e => setUsername(e.target.value)} placeholder="Username" />
      <input value={password} type="password" onChange={e => setPassword(e.target.value)} placeholder="Password" />
      <button type="submit">Login</button>
      <div>{msg}</div>
    </form>
  );
}
```

Компонент за Регистрация

```tsx
export default function Register({ onRegister }: { onRegister: () => void }) {
  const [username, setUsername] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [msg, setMsg] = useState("");

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    try {
      await register(username, email, password, "User"); // Добавяне на Role с стойност "User"
      setMsg("Registered! Please log in.");
      onRegister();
    } catch (err: any) {
      if (err.response && err.response.data) {
        setMsg(JSON.stringify(err.response.data));
      } else {
        setMsg("Failed to register.");
      }
    }       You, 2 weeks ago • add ui and ai
  };

  return (
    <form onSubmit={handleSubmit}>
      <input value={username} onChange={e => setUsername(e.target.value)} placeholder="Username" />
      <input value={email} type="email" onChange={e => setEmail(e.target.value)} placeholder="Email" />
      <input value={password} type="password" onChange={e => setPassword(e.target.value)} placeholder="Password" />
      <button type="submit">Register</button>
      <div>{msg}</div>
    </form>
  );
}
```

Входен и Изходен Модел

```
public class InputModel
{
    // The text input for sentiment/toxicity analysis
    [LoadColumn(fieldIndex: 0)]
    2 references | Petko Dapchev, 14 days ago | 1 author, 1 change
    public string Text { get; set; }

    [LoadColumn(fieldIndex: 1)]
    0 references | Petko Dapchev, 14 days ago | 1 author, 2 changes
    public string Label { get; set; }
}
```

```
6 references | Petko Dapchev, 14 days ago | 1 author, 2 changes
public class OutputModel
{
    [ColumnName("PredictedLabel")]
    1 reference | Petko Dapchev, 14 days ago | 1 author, 2 changes
    public string PredictedLabel { get; set; }

    [ColumnName("Score")]
    1 reference | Petko Dapchev, 14 days ago | 1 author, 1 change
    public float[] Score { get; set; }
}
```

Сервиз за анализиране на коментари

```
public class SentimentPredictionService
{
    private readonly PredictionEngine<InputModel, OutputModel> _predictionEngine;

    0 references | Petko Dapchev, 22 days ago | 1 author, 1 change
    public SentimentPredictionService()
    {
        var mlContext = new MLContext();
        var model :ITransformer? = mlContext.Model.Load( filePath: "MLModel.zip", out var modelInputSchema);
        _predictionEngine = mlContext.Model.CreatePredictionEngine<InputModel, OutputModel>(model);
    }

    0 references | Petko Dapchev, 22 days ago | 1 author, 1 change
    public OutputModel Predict(string text)
    {
        var input = new InputModel { Text = text };
        return _predictionEngine.Predict(input);
    }
}
```

Контролер за Анализ

```
[ApiController]
[Route( template: "[controller]")]
1 reference | Petko Dapchev, 10 days ago | 1 author, 4 changes
public class AnalyzeController : ControllerBase
{
    private static readonly object Lock = new object();
    private static PredictionEngine<InputModel, OutputModel> _engine;
    private static readonly string MlNetModelPath = Path.GetFullPath("MLModel.mlnet");

    0 references | Petko Dapchev, 10 days ago | 1 author, 4 changes
    static AnalyzeController()
    {
        var mlContext = new MLContext();
        lock (Lock)
        {
            var model :ITransformer? = mlContext.Model.Load(MlNetModelPath, out _);
            _engine = mlContext.Model.CreatePredictionEngine<InputModel, OutputModel>(model);
        }
    }

    [HttpPost]
    0 references | Petko Dapchev, 10 days ago | 1 author, 2 changes
    public ActionResult<OutputModel> Post([FromBody] InputModel input)
    {
        if (string.IsNullOrWhiteSpace(input.Text))
            return BadRequest();

        var prediction :OutputModel? = _engine.Predict(input);

        var result :{PredictedLabel,Scores} = new
        {
            prediction.PredictedLabel,
            Scores = prediction.Score
        };

        return Ok(result);
    }
}
```

Примерни Данни

```
Text,Label
"I hate this forum, its the worst!",1
"This is a very helpful post, thanks!",0
You are so dumb and annoying.,1
I totally agree with your point!,0
Why do you even exist?,1
"Great answer, really insightful.",0
You're such an idiot.,1
"Nice work, this really helped me.",0
This is stupid and a waste of time.,1
Fantastic explanation. Very clear.,0
You should just shut up already.,1
Thank you for taking the time to explain.,0
Nobody cares about your opinion.,1
This post saved me hours of work!,0
You're completely useless.,1
Very professional and respectful reply.,0
What a pile of garbage.,1
I'm impressed by your perspective.,0
You always ruin everything.,1
Good clarification. Appreciate it.,0
Stop spamming your nonsense here.,1
That actually makes sense now. Thanks!,0
I can't believe how dumb you are.,1
You're very articulate. I like your style.,0
This thread is trash just like the OP.,1
Thanks for helping out. Much appreciated.,0
You don't belong here. Leave.,1
Great example. That helped me understand.,0
You're pathetic and a joke.,1
Excellent response. Totally agree.,0
```

7. Заключение

Документацията предоставя пълен преглед на форумното приложение, включително компонентите на бекенда за аутентикация, модериране и управление на потребители, както и интеграцията на машинно обучение за анализ на настроението. Компонентите на фронтенда и бекенда са детайлно описани, показвайки как взаимодействат с едно с друго и ML модела, за да покрият изискванията. Документацията включва и примерни данни и конфигурацията, необходима за настройка и стартиране на приложението, както и частти от имплементацията за по-голямо разбиране.