Course Project

February 2023

You work for Met@ wants to diversity its portfolio besides the met@verse, because the investors are getting mad. So, they hire you and your team to develop a social network with a specific target in mind. By February 27 you must develop an instance of an MVP (Minimum Viable Product) for Mark Zucabyte to test.

Here are the in-depth technical requirements for our platform:

- Customizable Profiles: Users can create and customize their profiles with personalized header images, background colors, and font styles, allowing them to showcase their sarcastic flair.
- User Feed: A real-time feed of all the latest sarcastic posts, featuring an infinite scroll for endless hours of mock-worthy content.
- User Search: An search engine that allows users to find other users and their profiles.
- Commenting System: A commenting system that allows users to engage in sarcastic banter with one another, complete with real-time notifications, emoji reactions, and the ability to upvote or downvote comments.
- Deployment: The system must be deployed for users to be able to use it.
- Security: The platform must be secure, protecting user data with encryption, and ensuring the privacy of their sarcastic musings.

Your main page must have a slogan and the name of the social network.

Remember that there are competing teams since Mark can only invest in a few ideas. So you must strive for a robust and understandable code that will push production and maintenance costs and delays down. However, you want your database to scale well to many users, you must should not use ORM, since the the time it takes to do the mapping is not negligible. Take that into consideration while developing the app.

Each page in your project should have at least one test.

1 Bad news?

Since then, the Met@ stocks went down again and your boss is thinking about replacing you with ChatGTP. So he asked it to do your work. However, this is what ChatGPT came up with:

1.1 Home page

```
<!DOCTYPE html>
<html>
<head>
  <title>Sarcastic Social Network</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/
  4.0.0/css/bootstrap.min.css" integrity=
  "sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
  crossorigin="anonymous">
  <style>
    .container {
      margin-top: 50px;
      text-align: center;
    }
    h1 {
      font-weight: bold;
      font-size: 36px;
      margin-bottom: 20px;
    }
    p {
      font-size: 18px;
      margin-bottom: 40px;
    }
    .btn-primary, .btn-secondary {
      margin-right: 20px;
      font-size: 18px;
      padding: 10px 20px;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Welcome to the Sarcastic Social Network</h1>
    <p>Join the community and share your sarcastic wit with the world.</p>
    {% if not current_user.is_authenticated %}
      <a href="{{ url_for('login') }}" class="btn btn-primary">Login</a>
      <a href="{{ url_for('register') }}" class="btn btn-secondary">Register</a>
    {% else %}
      <p>Hello {{ current_user.username }}!</p>
```

```
      <a href="{{ url_for('logout') }}" class="btn btn-secondary">Logout</a>
    {% endif %}
  </div>
</body>
</html>
```

1.2 User's profile page

```
<!DOCTYPE html>
<html>
<head>
  <title>{{ user.username }}'s Profile</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/
  4.0.0/css/bootstrap.min.css" integrity=
  "sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
  crossorigin="anonymous">
  <style>
    .container {
      margin-top: 50px;
      text-align: center;
    }
    h1 {
      font-weight: bold;
      font-size: 36px;
      margin-bottom: 20px;
    }
    p {
      font-size: 18px;
      margin-bottom: 40px;
    }
    h2 {
      font-weight: bold;
      font-size: 24px
      margin-top: 40px;
    }
    .post {
      border: 1px solid gray;
      padding: 20px;
      margin-bottom: 20px;
    }
    .post h3 {
      font-weight: bold;
      font-size: 18px;
      margin-bottom: 10px;
    }
```

```
    .post p {
      font-size: 16px;
      margin-bottom: 10px;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>{{ user.username }}'s Profile</h1>
    {% if user == current_user %}
      <a href="{{ url_for('edit_profile') }}" class="btn btn-secondary">Edit Profile</a>
    {% endif %}
    <h2>Posts</h2>
    {% for post in posts %}
      <div class="post">
        <h3>{{ post.title }}</h3>
        <p>{{ post.body }}</p>
      </div>
    {% endfor %}
  </div>
</body>
</html>
```

1.3 New post page

```
<!DOCTYPE html>
<html>
<head>
  <title>Create a New Post</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/
  4.0.0/css/bootstrap.min.css" integrity=
  "sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
  crossorigin="anonymous">
  <style>
    .container {
      margin-top: 50px;
      text-align: center;
    }
    h1 {
      font-weight: bold;
      font-size: 36px;
      margin-bottom: 20px;
    }
    form {
```

```
      margin-top: 40px;
      text-align: left;
      display: inline-block;
      width: 60%;
    }
    label {
      font-weight: bold;
      font-size: 18px;
      margin-bottom: 10px;
    }
    input[type="text"], textarea {
      width: 100%;
      padding: 10px;
      font-size: 16px;
      margin-bottom: 20px;
    }
    input[type="submit"] {
      padding: 10px 20px;
      font-size: 18px;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Create a New Post</h1>
    <form action="{{ url_for('create_post') }}" method="post">
      <label for="title">Title:</label>
      <input type="text" id="title" name="title" required>
      <label for="body">Body:
      <textarea id="body" name="body" rows="10" required></textarea>
      <input type="submit" value="Create Post">
    </form>
  </div>
</body>
</html>
```

1.4 User profile page and the new post page

```
from flask import Flask, render_template, request, redirect, url_for
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
```

```
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///social_network.db'
db = SQLAlchemy(app)

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    posts = db.relationship('Post', backref='author', lazy=True)

class Post(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(80), nullable=False)
    body = db.Column(db.Text, nullable=False)
    author_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)

@app.route('/')
def index():
    return redirect(url_for('all_users'))

@app.route('/users')
def all_users():
    users = User.query.all()
    return render_template('all_users.html', users=users)

@app.route('/users/<int:user_id>')
def user_profile(user_id):
    user = User.query.get(user_id)
    posts = Post.query.filter_by(author_id=user_id).all()
    return render_template('user_profile.html', user=user, posts=posts)

@app.route('/posts/new', methods=['GET', 'POST'])
def new_post():
    if request.method == 'POST':
        title = request.form['title']
        body = request.form['body']
        author_id = current_user.id
        post = Post(title=title, body=body, author_id=author_id)
        db.session.add(post)
        db.session.commit()
        return redirect(url_for('user_profile', user_id=author_id))
    return render_template('new_post.html')

if __name__ == '__main__':
    db.create_all()
    app.run(debug=True)
```

Based on what you have learned, can you fix this code?

2 Bonus

2.1 1V

To help you with the project management you asked ChatGPT to help you with the generation of tasks. This is what he gave you:

"Here are the steps you can follow:

- Set up the Flask framework: You can install Flask using pip and create a new Flask application with a basic file structure.
- Create a database: You can use SQLite as the database for your social network. You can create tables to store user information, posts, comments, etc.
- Create models: You can use Flask-SQLAlchemy to create models for your database tables. You can define the fields, relationships, and methods for your models.
- Create views: You can create views to handle user requests and render templates. You can use flask's @app.route() decorator to define URL routes for each view.
- Create templates: You can create HTML templates to render the views. You can use Jinja2 templates to insert dynamic content from your views.
- Add functionality: You can add functionality for features such as user authentication, creating posts, commenting, searching, etc.
- Test and Deploy: Finally, you can test your application locally and then deploy it to a hosting service."

Can you spot if there is anything wrong?

2.2 0,5V

Bonus: How would you prevent SQL injections in you application?

You can write the answers to these questions in a "answers.txt" file at the root of your repository.