TikTok Clone Web App

A simplified TikTok-style social media platform with user authentication, post creation, followers/following system, and real-time interactions using Flask, Socket.IO, and Tailwind CSS.

Table of Contents

Features

- **User Authentication:** Sign up and login with username, email, and password
- **Validation & Flash Messages:** Signup checks for existing username/email; Login flashes incorrect password or "username not found"
- **Session Management:** Redirects unauthenticated users to the home page
- **Dashboard:** Upload posts (image/video), view posts, followers, and following, search posts and users
- **Profile:** View stats (posts, followers, following) and bio
- **Follow/Unfollow:** Real-time using Socket.IO
- **Responsive Design:** Tailwind CSS for modern UI
- **3D Animation:** Homepage with Spline interactive 3D visuals
- **Database:** MySQL via XAMPP
- **Flash Messages:** Notify users for errors and success

Tech Stack

Layer	Technology
Backend	Flask, Flask-RESTful, Flask-SocketIO
Frontend	HTML, CSS, JavaScript, Tailwind CSS
Database	MySQL (XAMPP)
Real-Time	Socket.IO
Animation	Spline 3D

Installation

Follow these steps to set up the TikTok Clone Web App on your local machine:

1. Clone the Repository

```
git clone https://github.com/yourusername/tiktok-clone.git
cd tiktok-clone
```

2. Create Virtual Environment

```
python -m venv venv
# Activate environment
source venv/bin/activate  # Windows: venv\Scripts\activate
```

3. Install Dependencies

```
pip install -r requirements.txt
```

4. Configure Database

```
SQLALCHEMY_DATABASE_URI = "mysql+pymysql://username:password@localhost/
db_name"
```

5. Initialize Database

```
set FLASK_APP=app.py
flask db init
flask db migrate
flask db upgrade
```

6. Run the Application

```
python app.py
```

Project Structure

```
tiktok-clone/ │ ├ README.md ├ requirements.txt ├ .gitignore ├ backend/ │ ├ resources/ │ │
├ home.py │ │ ├ profile.py │ │ ├ upload.py │ │ └ user.py │ ├ routes/ │ │ └ routes.py │
├ app.py │ ├ config.py │ ├ extensions.py │ └ model.py ├ frontend/ │ ├ 404.html │ ├
edit.html │ ├ home.html │ ├ login.html │ ├ main.html │ ├ post.html │ ├ profile.html │ ├
register.html │ ├ upload.html │ └ userprofile.html ├ upload/ │ ├ files/ │ └ profile/ ├
document/ │ ├ follow_unfollow.png │ ├ get_post.png │ ├ home_page.png │ ├
loading_effect.png │ ├ login_page.png │ ├ profile_edit_page.png │ ├ profile+page.png │ ├
registration_page.png │ ├ search_post.png │ ├ upload_page.png │ ├ upload_sample.png │ ├
xammp.png │ ├ code.png │ └ files.png └ config.py
```

Database Tables

The application uses MySQL database with the following table structure:

Table	Columns	Description
User	id (PK), username (unique), email (unique), password	Stores user credentials
Posts	id (PK), username, title, caption, post (file path)	Stores all user posts
Profile	id (PK), username, bio, profile (image path)	Stores profile info
Followers	id (PK), username, follower	Tracks followers for each user

API Endpoints

The following RESTful API endpoints are available:

Endpoint	Method	Description
`/register`	POST	Register new user
`/login`	POST	Login existing user
`/dashboard/upload`	POST	Upload post (image/video)
`/dashboard/home`	GET	Fetch all posts for main feed
`/search?search=`	GET	Search posts or users
`/user/<search>`	GET	Fetch searched user profile
`/dashboard/logout`	GET	Logout user (destroy session)
`/dashboard/profile/edit`	POST	Edit logged-in user profile
`/dashboard/profile`	GET	Get current profile info

Authentication & Validation

Sign Up

> **Fields:** Username, Email, Password
>
> **Validation Checks:**
>
> - Username exists → Flash: `Please use another username`
> - Email exists → Flash: `Email already registered`

Login

> **Fields:** Username, Password
>
> **Validation Checks:**
>
> - Incorrect password → Flash: `Incorrect password`
> - Username not found → Flash: `No username found`

Session Management

> The application implements secure session management:
>
> - Checks for session cookie on dashboard pages
> - Redirects to home page if session is invalid
> - Session destroyed on logout

Real-Time Interactions

Follow/Unfollow functionality is handled via Socket.IO for real-time updates without page reload.

Socket.IO Implementation

```
const followBtn = document.getElementById("followBtn");
if(followBtn){
  const socket = io();
  followBtn.addEventListener("click", () => {
    socket.emit("follow", { follow: "{{ searched_user.username }}" });
  });
  socket.on("follow_status", (data) => {
    if (data.status === "followed") {
      followBtn.textContent = "Unfollow";
      followBtn.classList.replace("from-pink-500","from-gray-500");
    } else {
      followBtn.textContent = "Follow";
      followBtn.classList.replace("from-gray-500","from-pink-500");
    }
  });
}
```

Key Features:

- Real-time button state updates
- Dynamic CSS class changes
- No page refresh required
- Instant feedback to user actions

Frontend Design

The frontend is built with modern web technologies for a responsive and engaging user experience:

- **Tailwind CSS:** Utility-first CSS framework for rapid UI development
- **Spline 3D:** Interactive 3D animations on the homepage
- **Responsive Layout:** Mobile-first design approach
- **Flash Messages:** Real-time user notifications

Key Pages

- **home.html:** Landing page with 3D animations
- **login.html / register.html:** Authentication pages

main.html: Dashboard with post feed

profile.html: User profile with stats

upload.html: Post creation interface

userprofile.html: Other users' profiles

Usage

Getting Started

Start XAMPP and ensure MySQL is running

Launch the Flask application using `python app.py`

Navigate to `http://localhost:5000` in your browser

Create an account or login with existing credentials

Creating Posts

Navigate to the upload page from the dashboard

Select an image or video file

Add a title and caption

Click upload to share your post

Following Users

Search for users using the search bar

Click on a user's profile

Click the "Follow" button

The button updates to "Unfollow" in real-time

Screenshots & Demo

The `document/` folder contains screenshots of the application including:

home_page.png: Landing page with 3D animation

registration_page.png: User signup interface

login_page.png: User login interface

get_post.png: Main feed with posts

upload_page.png: Post creation interface

profile_page.png: User profile view

follow_unfollow.png: Follow/Unfollow functionality

search_post.png: Search results

loading_effect.png: Loading animations

> **Note:** For a complete demo and live screenshots, please refer to the `document/` directory in the project repository.