Name:

Vigyat Padiya(92300527222),

Dhruv Makwana(92300527203)

Subject:

Mini Project

Topic:

Youtube Video Downloader

Table of Contents

1. Introduction

- The YouTube Video Downloader is a secure and user-friendly web application that enables users to download YouTube videos in various formats and resolutions.
- It is built with Flask (Python) for backend processing, MySQL for data storage, and yt-dlp for video extraction.
- The system integrates user authentication, download history tracking, and an administrative dashboard.
- The application recommends VLC Media Player for optimal playback due to its wide codec support.

2. Features

- Video Downloading: Download YouTube videos in multiple resolutions (360p–8K) in MP4 format
- User Authentication: Secure registration/login system with bcrypt password hashing
- Download History: Tracks downloads for registered users with timestamps
- Admin Dashboard: Allows administrators to manage users and view download statistics
- Responsive UI: Modern glassmorphism-based design with Tailwind CSS
- Format Detection: Automatically detects available resolutions and formats
- Clipboard Integration: One-click paste button for URLs
- Session Management: Secure session handling with Flask sessions
- Playback Recommendation: Suggests VLC Media Player for smooth video playback

3. System Architecture

- Frontend: HTML, Tailwind CSS, JavaScript
- Backend: Flask (Python) for request handling and processing
- Database: MySQL for persistent data storage
- Video Processing: yt-dlp for video extraction and FFmpeg for format handling
- File Handling: Temporary file system for processing and delivery

4. Installation Guide

Prerequisites:

- Python 3.7+
- MySQL 5.7+
- yt-dlp library
- FFmpeg (required for merging video/audio streams)
- VLC Media Player (recommended for playback)

Step-by-Step Setup:

1. Environment Setup: pip install flask yt-dlp mysql-connector-python bcrypt werkzeug

    - Install MySQL server
    - Install FFmpeg
    - (Optional) Install VLC Media Player for playback

2. Database Configuration:

    - Run setup_database.py to create required tables.

    Default credentials:

    - Admin → admin/admin123
    - User → user/user123

3. Application Configuration:

- DB_HOST → MySQL host (default: localhost)
- DB_USER → MySQL user (default: root)
- DB_PASSWORD → MySQL password
- DB_NAME → youtube_downloader
- FLASK_SECRET_KEY → secret key for sessions

4. Run Application: python app.py

    Access at: http://localhost:5000

5. Technical Implementation

File Structure:

- app.py → main Flask application
- setup_database.py → database initialization script
- templates/ → HTML files with responsive UI

Database Schema:

- Users Table: credentials, roles (admin/user)
- Downloads Table: tracks video downloads with timestamps

Authentication System:

- Passwords hashed with bcrypt
- Flask sessions for login persistence
- Role-based access control (Admin/User)

Video Processing:

- yt-dlp extracts video/audio streams
- FFmpeg merges best video + audio
- Files stored temporarily, then deleted after serving

Security Measures:

- Input validation for URLs and forms
- Parameterized queries to prevent SQL injection
- Secure file handling and sanitization
- CSRF protection via Flask sessions

Key Functions:

- is_valid_youtube_url(): ensures input is a valid YouTube link
- Format Selection: chooses best resolution and audio merge
- User Management: registration, login, role assignment
- Download Tracking: logs user downloads with metadata

6. Testing

Functional Testing:

- URL validation across multiple formats
- User registration/login/logout
- Downloads at various resolutions
- Admin role verification (user management, statistics)
- VLC compatibility testing

Performance Testing:

- Response time for video metadata retrieval
- Download speed benchmarks
- Multi-user concurrency testing
- Video playback verification with VLC

Security Testing:

- SQL injection resistance
- XSS prevention checks
- Session hijacking/bypass attempts
- Authentication robustness

7. Limitations

- Files are stored temporarily and deleted after serving
- Dependency on YouTube's API structure—breaks possible if YouTube updates
- Single-server setup; no distributed load balancing
- No batch/playlist downloads
- Legal issues: downloading YouTube videos may violate ToS
- Playback recommendation requires external media player (VLC)

8. Future Enhancements

- Batch processing (playlists/multiple URLs)
- Support for additional formats (WebM, audio-only, etc.)
- Cloud storage integration (Google Drive, Dropbox)
- Basic video editing features (trimming, conversion)
- REST API for third-party integration
- Progressive Web App (PWA) version
- Download scheduling support
- User-specific preferences for default formats
- Built-in media player
- Direct integration with VLC for seamless playback

9. Conclusion

The YouTube Video Downloader provides a secure, reliable, and efficient solution for downloading YouTube videos.
It integrates Flask, MySQL, yt-dlp, and FFmpeg for robustness and usability.
Key strengths: user management, download history tracking, and admin features.
Limitations: no batch downloads, dependency on YouTube API updates.
Future: cloud integration, playlist downloads, VLC integration → towards a complete media management platform.