C++ Based Social Media App - Project Report

# Introduction

Project Title: "C++ Based Social Media App with Minimal JavaScript"  
  
Objective:  
The objective of this project is to build a minimal yet functional social media platform using C++ as the backend engine, HTML and CSS for the frontend, and minimal JavaScript only for necessary asynchronous operations. This app supports core features like user registration, login, post creation, commenting, friend connections, and notification handling.  
  
Motivation:  
Many web applications rely heavily on JavaScript. This project explores the feasibility of implementing web application backend logic using an object-oriented C++ approach integrated with HTML/CSS through CGI scripts, while minimizing client-side scripting.

# Features and Functionality

1. User Registration and Login  
 - Handled by register.cgi and login.cgi  
 - User credentials are stored in a plain text database (users.txt)  
 - Passwords stored as plain strings (not hashed, for simplicity)  
  
2. Post Creation and Viewing  
 - Users can create posts  
 - Posts are only visible to friends  
 - Stored in a posts.txt database and in memory using std::list<Post>  
  
3. Friendship System  
 - Users can send friend requests and manage connections  
 - Managed via Relationship class  
 - Only friends can view each other’s posts  
  
4. Commenting System  
 - Posts can have associated comments  
 - Managed through Comment class  
  
5. Notification System  
 - When a user posts something, friends are notified  
 - Managed via Notification class  
  
6. Frontend Integration  
 - Separate HTML pages for login, register, home, and profile  
 - JavaScript only used to handle form submissions and asynchronous post loading

# OOP Concepts Used

1. Encapsulation  
 - All data (e.g., user credentials, posts) is encapsulated within classes  
 - Public interfaces expose necessary functions, while private variables ensure internal data safety  
  
2. Abstraction  
 - High-level interfaces provided for user operations, post management, and notification handling

# File Structure and Integration

Folder Structure:  
  
/project-root  
│  
├── cgi-bin/  
│ ├── login.cgi  
│ ├── register.cgi  
│ └── getposts.cgi  
│  
├── src/  
│ ├── main.cpp  
│ ├── User.cpp  
│ ├── Post.cpp  
│ ├── Comment.cpp  
│ ├── Notification.cpp  
│ └── Relationship.cpp  
│  
├── headers/  
│ ├── User.h  
│ ├── Post.h  
│ ├── Comment.h  
│ ├── Notification.h  
│ └── Relationship.h  
│  
├── html/  
│ ├── register.html  
│ ├── login.html  
│ ├── home.html  
│ └── profile.html  
│  
├── js/  
│ ├── register.js  
│ ├── login.js  
│ └── loadposts.js  
│  
├── database/  
│ ├── users.txt  
│ └── posts.txt

# Challenges and Future Improvements

Challenges Faced:  
- C++ is not typically used in full-stack web development, so Crow was manually implemented  
- Managing synchronous user state without sessions or cookies  
- Handling file I/O securely and efficiently  
  
Future Improvements:  
- Add password hashing using sha256 or bcrypt  
- Improve frontend with AJAX for better user experience  
- Integrate SQLite or a JSON-based local DB for more scalable storage  
- Add session and cookie support for secure logins  
- Unit tests for all classes and integration testing for CGI