SYNOPSIS OF LAB ORIENTED PROJECT (LoP)

ON

PeepPost

A Social Media Platform

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Parwinder Singh - 2110996028

Samyak Jain – 2110992476

Aditya Sharma – 2110992410

Raghav Purohit – 2110992352

Kshama Sharma - 2110996041

Supervised By:

Dr. Gifty Gupta

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CHITKARA UNIVERSITY INSTITUTE OF ENGINEERING AND TECHNOLOGY
CHITKARA UNIVERSITY, PUNJAB, INDIA

CONTENTS

Software Requirements Specification (SRS) Document for PeepPost App

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to provide a detailed description of the PeepPost application. PeepPost is a microblogging platform similar to Twitter, enabling users to post short messages, follow other users, and interact with posts. This document outlines the functional and non-functional requirements, system architecture, and design specifications necessary for the development of the PeepPost app.

1.2 Scope

PeepPost will provide a platform for users to sign up, log in, post messages (referred to as "peeps"), and view posts from other users. Key features include secure user authentication with hashed passwords, OTP-based verification via email, and a basic relational model with two core tables: Users and Posts. Future enhancements will include functionality to reply to posts.

1.3 Definitions, Acronyms, and Abbreviations

- PeepPost: The name of the microblogging application.
- User: An individual who has an account on PeepPost.
- Peeps: Messages or posts made by users.
- OTP: One-Time Password.
- Nodemailer: A module for sending emails in Node.js applications.
- Express: A web application framework for Node.js.
- MongoDB: A NoSQL database used for storing user and post data.

2. Overall Description

2.1 Product Perspective

PeepPost will be built as a web application using the Express framework for the backend and MongoDB for data storage. The application will have a straightforward interface with basic functionalities essential for a microblogging site.

2.2 Product Functions

The primary functions of PeepPost include:
- User Registration and Authentication
- Posting Messages (Peeps)
- Viewing Peeps from other users
- Email-based OTP verification

2.3 User Classes and Characteristics
1. Registered User: Has an account with PeepPost and can post messages, view posts, and follow other users.
2. Guest User: Can view posts but must register to post messages and interact with other users.

2.4 Operating Environment
- Server: Node.js with Express
- Database: MongoDB
- Client: Web browsers (Chrome, Firefox, Safari, etc.)

2.5 Design and Implementation Constraints
- Security: Passwords will be hashed using a secure hashing algorithm.
- Performance: The application should handle a moderate number of concurrent users and posts efficiently.
- Scalability: Future enhancements, such as replying to posts, should be supported.

3. Functional Requirements
3.1 User Registration
- Input: User's email address, username, password.
- Processing:
  - Validate the input data.
  - Hash the password before storing it in the database.
  - Send an OTP to the user's email for verification.
- Output:
  - Confirmation of successful registration.
  - Email with OTP for account verification.

3.2 User Login

- Input: Username or email address, password.
- Processing:
  - Verify the user credentials.
  - Check the hashed password against the stored hash.
- Output:
  - Authentication token for logged-in users.

3.3 Posting Messages
- Input: Text content of the peep.
- Processing:
  - Validate the peep content.
  - Store the peep in the database associated with the user.
- Output:
  - Confirmation of successful post creation.

3.4 Viewing Posts
- Input: Request to view posts.
- Processing:
  - Fetch posts from the database.
  - Display posts in chronological order.
- Output:
  - List of posts from followed users or public posts.

3.5 OTP-Based Email Verification
- Input: OTP sent to the user's email.
- Processing:
  - Validate the OTP.
  - Activate the user's account upon successful OTP verification.
- Output:
  - Confirmation of account activation.

3.6 Future Scope: Reply to Posts
- Input: Reply content, target post ID.
- Processing:
  - Validate the reply content.
  - Associate the reply with the target post in the database.

- Output:
  - Confirmation of successful reply creation.

4. Non-Functional Requirements

4.1 Performance
- The application should respond to user requests within 2 seconds.
- It should handle up to 1000 concurrent users.

4.2 Security
- Passwords must be hashed using a secure algorithm (e.g., bcrypt).
- OTPs must be securely generated and validated.

4.3 Usability
- The user interface should be intuitive and easy to navigate.
- Provide clear error messages and feedback for user actions.

4.4 Reliability
- The application should have an uptime of 99.5%.
- Regular backups of the database should be performed.

4.5 Scalability
- The system should be able to scale to accommodate increased user load and additional features (e.g., replying to posts).

5. System Architecture
5.1 Overview
The PeepPost app will use a client-server architecture with the following components:
- Client: Web interface for users to interact with the application.
- Server: Node.js server running Express to handle HTTP requests and interact with the MongoDB database.
- Database: MongoDB for storing user and post data.

5.2 Components

- Frontend: HTML, CSS, JavaScript (React or similar library).
- Backend: Node.js with Express.
- Database: MongoDB.
- Email Service: Nodemailer for sending OTP emails.

5.3 Data Model

- Users Table:
  - `userId` (String, unique, primary key)
  - `email` (String, unique)
  - `username` (String, unique)
  - `hashedPassword` (String)
  - `createdAt` (Date)
  - `updatedAt` (Date)

- Posts Table:
  - `postId` (String, unique, primary key)
  - `userId` (String, foreign key referencing Users)
  - `content` (String)
  - `createdAt` (Date)
  - `updatedAt` (Date)

6. Glossary

- Hashing: A process of transforming a password into a fixed-size string of characters, which is typically a hash code.
- OTP: A one-time password used for verifying the user's identity.

7. Appendices

7.1 Assumptions and Dependencies

- The application assumes a stable internet connection for both server and client.
- Dependency on external libraries such as Express, MongoDB, and Nodemailer.

7.2 Future Enhancements

- Implement the functionality to reply to posts.
- Add features for user profiles and following/unfollowing other users.