Subject: LPI(ADBMS) Class: TE IT B

Mini Project Synopsis

Group Id: 4

GroupMembers:-

Roll No.	Name	Mobile	Email
37007	Tanmay Chandgude	8411860355	tanmaychandgude.007@gmail.com
37059	Siddhi Shinde	9021726790	siddhishinde384@gmail.com
37027	Arya Kasliwal	7720048199	arya123kasliwal@gmail.com
37004	Aaryan Bairagi	9850088559	aaryanbairagi1108@gmail.com

Project Title:- Authify: The JWT-Powered Fortress for Secure Authentication

Problem statement :

- **Project Vision**

  In the digital age, securing user data and managing access has become increasingly critical for businesses and developers alike. Authify aims to address the need for robust, secure, and seamless authentication methods by providing an easy-to-integrate authentication tool that exclusively uses JWT (JSON Web Token) for token-based authentication. Our vision is to create a lightweight, highly secure, and scalable authentication system that simplifies user management and access control for developers, ensuring that they can easily implement secure authentication in their applications.

- **Issues to be Addressed:** Traditional authentication systems are often complex and prone to security risks such as password breaches, session hijacking, and token theft. Authify aims to solve the following issues:

1. **Security Vulnerabilities:** Reducing common vulnerabilities in user authentication such as brute force attacks, password leaks, and unauthorized access.
2. **Ease of Integration:** Offering a simple solution for developers to easily integrate secure JWT-based authentication without requiring complex setups.
3. **Scalability and Performance:**Providing a scalable, high-performance solution for handling user authentication requests with minimal overhead.

- **Methodology:**

  Authify is built using the MERN stack, which consists of MongoDB, Express.js, React, and Node.js. The project uses JWT exclusively for secure token-based authentication. Each authentication token is digitally signed and transmitted securely. The following packages and tools are employed for the project's functionality:

  - **Express:** A lightweight web framework for handling server-side logic and routing.

  - **Cookie-Parser:** For parsing cookies from incoming HTTP requests to handle session data efficiently.

  - **Mailtrap:** Used to send transactional emails (e.g., account verification, password resets).

  - **Bcryptjs:** For hashing passwords and securing user credentials.

  - **Dotenv:** For loading sensitive configuration variables from .env files.

  - **JSONWebToken (JWT):** Used for creating, signing, and verifying authentication tokens to ensure a secure session mechanism.

  - **Mongoose:** To interact with MongoDB and manage user data storage securely.

  - **Crypto:** For generating cryptographically secure random numbers for OTPs used in email verification.

  - **Nodemon** (dev dependency): For automatically restarting the Node.js server during development upon code changes.

  Additional frontend technologies include:

  - **Zustand:** A state management library for managing authentication states and user sessions efficiently.

  - **Axios**: For making HTTP requests to the backend API, ensuring secure communication between the frontend and backend.

- **Algorithm:** The JWT-based authentication process follows these steps:

1. **Signup:** The user signs up with their email and password. The password is hashed using Bcryptjs before being stored in the database.
2. **Email Verification:** An OTP is generated using the Crypto library and sent to the user's email using Mailtrap. The user needs to verify their email to activate the account.
3. **Login:** Upon successful login, a JWT is created, signed, and issued to the user. The token is used for subsequent requests, allowing access to protected routes.
4. **Token Verification:** The JWT is verified with each incoming request to ensure that the user is authenticated and authorized to access the resources.
5. **Password Reset:** A user can reset their password by generating an OTP and sending it to their email. The new password is hashed before being updated in the database.

Scope of Mini project :

The scope of the Authify project includes the core features and functionalities needed to address the identified authentication challenges and provide a secure, easy-to-use JWT-based authentication tool. These features aim to solve the security, ease of integration, and performance issues highlighted in the problem statement. The mini-project will focus on the following components:

- **Features and Functionality:**

  1. User Signup and Registration:

  - Users can sign up with their email and password.

  - Passwords are securely hashed using bcryptjs before being stored in the database.

  - Integration with MongoDB (via mongoose) for storing user credentials and profile data.

  2. Email Verification:

  - After signing up, users receive an email with an OTP (One-Time Password) for verification.

  - OTP generation using crypto to ensure secure, random codes.

  - Email delivery is handled through Mailtrap.

  - Users must verify their email addresses before gaining full access to the system.

  3. JWT-Based Authentication:

  - User login generates a JWT (JSON Web Token), which is used for authentication in subsequent requests.

  - Tokens are signed with a secure secret and issued to the user upon successful login.

  - The system will store the token securely using cookies, parsed by cookie-parser.

  4. Login and Token Verification:

  - Users can log in with their credentials, and upon successful authentication, a JWT token is generated.

  - The token is passed in each request to verify if the user has permission to access certain resources (protected routes).

  - Tokens are verified in real-time with each incoming request to the backend using json web token.

  5. Forgot Password and Password Reset:

  - Users can request a password reset, which triggers an OTP email for password recovery.

  - After verifying the OTP, users can set a new password.

6. Dashboard Access (Authenticated Routes):

- A user-specific dashboard is accessible only to authenticated users.

- JWT tokens are validated to ensure only verified users can access these protected routes.

7. Environment Configuration:

- Sensitive information such as database credentials, JWT secret keys, and mail service credentials are securely stored and managed using dotenv.

8. Frontend Integration:

- A simple, user-friendly React-based frontend built using React.

- Axios is used to handle all API requests between the frontend and the backend.

- The frontend includes login, signup, email verification, forgot password, and reset password pages.

- State management using Zustand to handle user session states across the application.

- UI elements include dynamic icons provided by Lucide-React to enhance the frontend experience.

9. Real-Time Development Environment:

- The server is monitored by nodemon, a development dependency that automatically restarts the application when changes are made, improving the development experience.

- **Expected Outcome:**

- The mini-project will fully address core authentication needs such as user sign-up, login, email verification, and token-based access control.

- It will implement secure methodologies like password hashing and JWT-based session management.

- The backend and frontend will work in harmony to deliver a seamless authentication experience, with built-in email verification and password reset mechanisms.

Hardware and Software Requirements

To successfully execute the Authify project, the following hardware and software components are required:

Hardware Requirements:

1. Processor:

- Minimum: Dual-core processor (e.g., Intel i3 or AMD equivalent)

- Recommended: Quad-core processor or higher for smoother performance (e.g., Intel i5 or AMD Ryzen)

2. Memory (RAM):

- Minimum: 4 GB

- Recommended: 8 GB or higher for handling development tasks and running local servers smoothly.

3. Storage:

- Minimum: 20 GB free disk space

- Recommended: 50 GB or higher to accommodate additional software, dependencies, and project files.

4. Internet Connection:

- Required for installing dependencies, sending emails (e.g., Mailtrap), and accessing cloud-based services (if applicable).

Software Requirements:

1. Operating System:

- Minimum: Windows 10, macOS 10.13 (High Sierra), or any recent Linux distribution (Ubuntu 18.04+).

- Recommended: Windows 11, macOS 11 (Big Sur), or Ubuntu 20.04+ for a stable and updated development environment.

2. Node.js:

- Minimum Version: 16.0.0 or higher

- Required for running the backend server and managing packages (via npm).

3. NPM (Node Package Manager):

- Installed with Node.js (version 6 or higher recommended) to handle project dependencies like Express, bcryptjs, JWT, etc.

4. Database:

- MongoDB (v4.4 or higher):

  - Required for storing user data, hashed passwords, email verification tokens, etc.

  - Can be run locally or through MongoDB Atlas (cloud-based).

5. Code Editor:

- VS Code (recommended): A popular code editor with support for Node.js, React, and JavaScript development.

- Alternative: Sublime Text, Atom, WebStorm.

6. Frontend Requirements:

- React: For building the user interface. Node.js will install it as part of the project setup.

- Axios: To handle HTTP requests between frontend and backend

- Lucide-React: For adding icons to the frontend.

- Zustand: For state management across React components.

7. Backend Frameworks and Libraries:

- Express.js (v4.19.2 or higher): The web framework to manage backend routes and APIs.

- Mongoose (v8.5.2 or higher): For MongoDB object modeling and database operations.

- Cookie-Parser: To parse and manage cookies.

- Bcryptjs: For hashing passwords.

- JSONWebToken (JWT): For handling authentication tokens.

- Mailtrap: For sending verification and password reset emails.

- Crypto: For generating cryptographically secure OTPs.

8. Environment Configuration:

- Dotenv: For managing sensitive environment variables such as JWT secrets and database credentials in a .env file.

9. Development Tools:

- Nodemon: Installed as a dev dependency to automatically restart the server during development on file changes.

References:

1. Express and Node.js: - [Learn Server-Side Development with Express and Node.js](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs)

2. React: [Getting Started with Client-side JavaScript Frameworks: React](https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started)

3. JavaScriptDocumentation- (https://developer.mozilla.org/en-US/docs/Web/JavaScript)

4. Axios: - [Axios Documentation: HTTP Client for JavaScript](https://axios-http.com/docs/intro)

5. Mailtrap API: - [Mailtrap API Documentation](https://api-docs.mailtrap.io/)

6. MongoDB: - [MongoDB Documentation](https://www.mongodb.com/docs/)

7. Mongoose:- [Mongoose Documentation](https://mongoosejs.com/docs/)

Subject Incharge

(Dr.Mrs.S.S. Bhavsar)