COLLEGE CODE:9111

COLLEGE NAME: SRM MADURAI COLLEGE FOR ENGINEERING AND TECHNOLOGY

DEPARTMENT: B.E COMPUTER SCIENCE AND ENGINEERING

STUDENT NM-ID: 451186A06133385A11F419FE498D4137

A8884C7CC5278F7B7394345D64BD84CB
1FDBD7B5DCA253284674E6DBD128FDFD
857E9787939AE7BBA024C9CE6C315408
DE8BAC54E8107FF98F34AABCF0F38D6C

ROLL NO.:911123104018
911123104020
911123104051
911123104053
911123104702

DATE:

Completed the project named as Phase II_

TECHNOLOGY PROJECT NAME: LOGIN AUTHENTICATION SYSTEM

SUBMITTED BY, NAME:
HOORUL FIRTHOUS.A
JAYASAKTHI K R
SRI LAKSHMI.S
TAAKSHINI DEVI.R
RIZWANA BARVEEN M

Project: Login Authentication System(Phase2)

---

1. Tech Stack Selection

- **Frontend**:
  - **HTML, CSS, JavaScript** (or React.js for SPA)
- **Backend**:
  - **Node.js + Express.js** (for REST API)
- **Database**:
  - **MongoDB / MySQL** (user storage)
- **Authentication**:
  - **JWT (JSON Web Tokens)** for stateless authentication
  - **bcrypt/argon2** for password hashing
- **Optional Tools**:
  - Nodemailer (for sending password reset emails)
  - Postman (for API testing)

2. UI Structure / API Schema Design

UI Structure

- **Login Page** → (email/username, password)
- **Signup Page** → (username, email, password, confirm password)
- **Forgot Password Page** → (email input, reset link/OTP)
- **Reset Password Page** → (new password input form)
- **Dashboard Page** → (welcome message, profile info, logout button)

API Schema Design

User Model (Database Schema)

```
{
  "id": "string (auto-generated)",
  "username": "string",
  "email": "string",
  "password": "hashed string",
  "createdAt": "timestamp",
  "updatedAt": "timestamp"
}
```

Auth API Schema

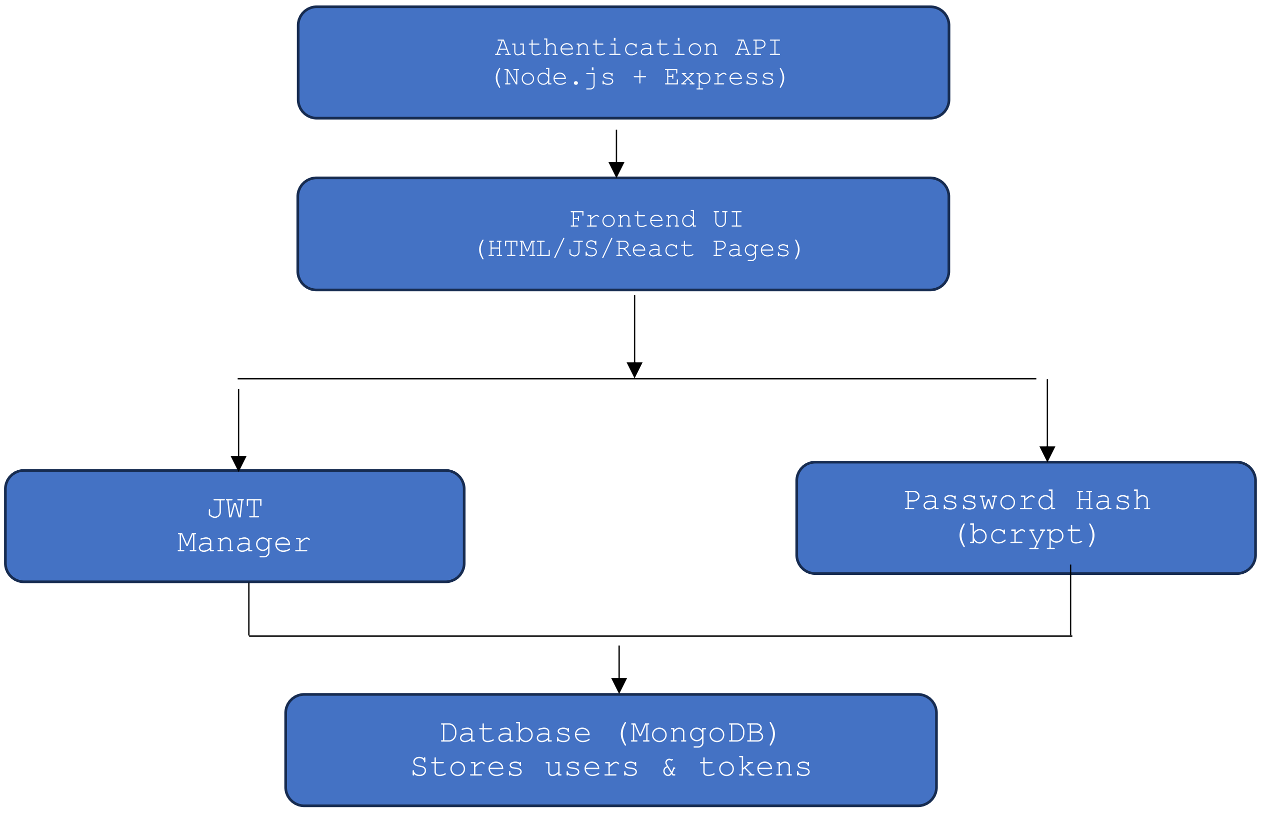
1. **Signup (POST /api/auth/signup)**
   Input: `{ username, email, password }`
   Output: `{ message: "User created successfully" }`
2. **Login (POST /api/auth/login)**
   Input: `{ email, password }`
   Output: `{ token: "JWT_TOKEN", user: {id, username, email} }`
3. **Forgot Password (POST /api/auth/forgot)**
   Input: `{ email }`
   Output: `{ message: "Reset link sent" }`
4. **Reset Password (POST /api/auth/reset)**
   Input: `{ token, newPassword }`
   Output: `{ message: "Password updated" }`

3. Data Handling Approach

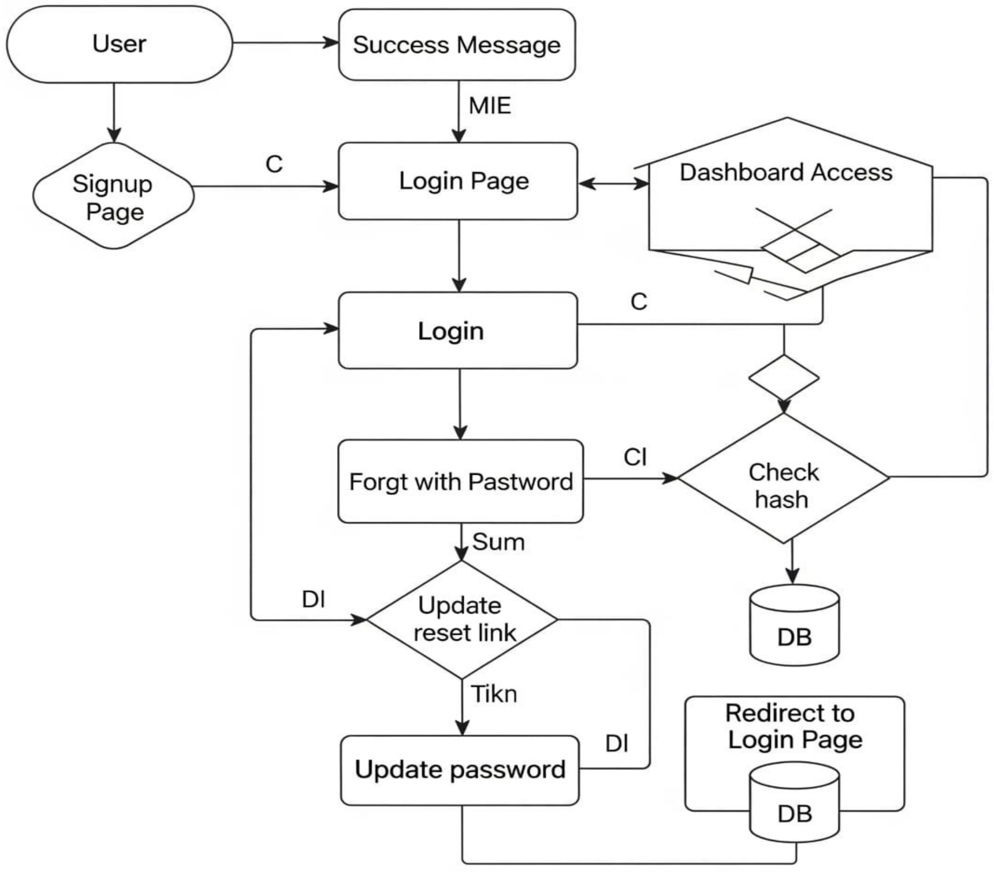
- **User Registration**:
  - Store **hashed password** in DB using bcrypt.
- **Login Authentication**:
  - Compare entered password with **stored hash**.
  - Generate **JWT token** for valid login.
- **Session Management**:
  - Token stored in browser localStorage or cookies.
- **Password Reset**:
  - Generate **reset token**, send via email.
  - User clicks reset link → verifies token → updates password.
- **Data Security**:
  - Enforce HTTPS.
  - Validate inputs (prevent SQL Injection / NoSQL Injection).

- Implement rate limiting for login attempts (prevent brute force).

4. Component / Module Diagram



5. Basic Flow Diagram


User
Success Message
MIE
Signup Page
C
Login Page
Dashboard Access
Login
C
Forgt with Pastword
Cl
Check hash
Sum
DI
Update reset link
DB
Tikn
DI
Update password
Redirect to Login Page
DB