High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components of **SafePass - Simple Auth API**. SafePass is a backend authentication API designed for educational purposes, providing user registration, secure password management, JWT-based login, and protected routes using Node.js, Express, and MongoDB.

1. System Architecture Overview

Architecture Description:
SafePass follows a modular, RESTful backend architecture. The system consists of an API server (Node.js/Express) interfacing with a MongoDB database. Authentication is handled via JWT tokens, and password security is ensured through hashing.

Component	Description
API Server	Handles HTTP requests, routes, and business logic (Node.js/Express)
Auth Module	Manages registration, login, password hashing, and JWT issuance
Protected Routes	Endpoints requiring valid JWT for access
Database Layer	Stores user data securely (MongoDB)

2. Component Interactions

Sequence Step	Interaction Description
1. Client Request	Client sends HTTP request (register/login/protected resource)
2. API Routing	Express routes request to appropriate controller/module
3. Auth Processing	Auth module handles registration/login, hashes passwords, issues JWT
4. DB Operations	Database layer reads/writes user data as needed
5. JWT Verification	Middleware checks JWT for protected routes
6. Response	API server returns response to client

3. Data Flow Overview

Flow Step	Data Involved	Direction
User Registration	Username, password	Client → API → DB
Password Hashing	Plain password → Hashed password	API (internal)

Login	Username, password	Client → API → DB
JWT Issuance	User ID, token	API → Client
Protected Resource Access	JWT token	Client → API (verify)

---

4. Technology Stack

Layer/Function	**Technology/Framework**
API Server	Node.js, Express
Database	MongoDB (Mongoose ODM)
Authentication	bcrypt (password hashing), jsonwebtoken (JWT)
API Testing	Postman (suggested)

---

5. Scalability, Reliability & Security

- **Scalability:** Stateless JWT authentication enables horizontal scaling of API servers. MongoDB supports sharding for larger datasets.
- **Reliability:** Error handling and input validation ensure API robustness. MongoDB provides data persistence.
- **Security:** Passwords are hashed with bcrypt. JWTs are signed and verified. Protected routes enforce authentication.

End of Document