Product Requirements & Specification Document

Project Name

SafePass - Simple Auth API

Description

SafePass is a backend authentication API designed for educational purposes. It provides user registration, secure password hashing, JWT-based login, and protected routes. The API is built using Node.js, Express, and MongoDB.

1. Goals & Objectives

Goal	Description
User Registration	Allow users to register with email and password
Secure Authentication	Implement password hashing and JWT-based login
Protected Endpoints	Restrict access to certain routes to authenticated users
Educational Simplicity	Keep implementation clear and easy to understand

2. Functional Requirements

ID	Requirement
FR1	Users can register with email and password
FR2	Passwords are securely hashed before storage
FR3	Users can log in with valid credentials and receive a JWT
FR4	JWT is required to access protected routes
FR5	API returns appropriate error messages for invalid actions
FR6	User data is stored in MongoDB

3. Non-Functional Requirements

ID	Requirement
NFR1	Use Node.js (LTS), Express, and MongoDB
NFR2	Codebase is modular and well-documented
NFR3	API follows RESTful conventions
NFR4	Passwords are hashed using bcrypt

NFR5	JWTs are signed with a secure secret
NFR6	No frontend; API only

4. API Endpoints

Method	Endpoint	Auth Required	Description
POST	/api/register	No	Register new user
POST	/api/login	No	Authenticate user, return JWT
GET	/api/protected	Yes	Example protected route

5. Data Model

User

Field	Type	Constraints
_id	ObjectId	Auto-generated
email	String	Required, unique, valid
password	String	Required, hashed
createdAt	Date	Auto-generated

6. Security Specifications

- Passwords hashed with bcrypt (min. 10 salt rounds)
- JWTs signed with environment-stored secret
- JWT expiry: 1 hour
- No sensitive data in JWT payload

7. Error Handling

Scenario	Response Code	Message Example
Invalid credentials	401	“Invalid email or password”
Email already registered	409	“Email already in use”
Missing/invalid JWT	401	“Authentication required”
Validation errors	400	“Invalid input”

8. Implementation Outline

```
// Registration
POST /api/register
  - Validate input
  - Hash password
  - Store user in MongoDB

// Login
POST /api/login
  - Validate input
  - Compare password hash
  - Issue JWT

// Protected Route
GET /api/protected
  - Verify JWT
  - Return protected data
```

9. Environment & Dependencies

Dependency	Version/Notes
Node.js	LTS
Express	Latest stable
MongoDB	Atlas/local
bcrypt	For password hashing
jsonwebtoken	For JWT handling
dotenv	For env variables

10. Out of Scope

- No frontend/UI
- No OAuth/social login
- No email verification
- No password reset

11. Success Criteria

- All endpoints function as specified
- Passwords never stored in plain text
- JWT authentication works as intended
- Code is clear, modular, and documented

12. Milestones

Milestone	Description
Setup & Boilerplate	Project structure, dependencies
User Registration	/api/register endpoint
User Login & JWT	/api/login endpoint, JWT handling
Protected Route	/api/protected endpoint
Documentation & Testing	API docs, basic tests

End of Document