SecureAccess Global Password Manager - Project Plan

Project Overview

Objective: Build a secure, global user account information application that stores usernames and passwords for any client or web application with robust security, JWT authentication, and comprehensive audit capabilities.

Database: MySQL ('secureaccess') **Architecture:** Multi-tier application with strong security focus

---

Phase 1: Foundation & Database Setup (Week 1-2)

Database Implementation

- [x] Database schema design completed
- [ ] Execute SQL table creation scripts
- [ ] Set up database indexes and constraints
- [ ] Configure database user accounts and permissions
- [ ] Implement database backup and recovery procedures
- [ ] Set up database connection pooling

Security Infrastructure

- [ ] Design encryption key management system
- [ ] Implement master password hashing strategy (bcrypt/Argon2)
- [ ] Set up JWT secret key management
- [ ] Design password encryption/decryption algorithms
- [ ] Implement secure random salt generation

Phase 2: Core Backend Development (Week 3-6)

Authentication System

- [ ] User registration API with validation
- [ ] Master password authentication
- [ ] JWT token generation and validation
- [ ] Refresh token rotation mechanism
- [ ] Session management system
- [ ] Password reset functionality with security questions

User Management APIs

- [ ] User CRUD operations
- [ ] Profile management (first name, last name updates)
- [ ] Account status management (active/inactive/locked)
- [ ] Two-factor authentication integration
- [ ] User preferences management

Password Storage System

- [ ] Stored accounts CRUD operations
- [ ] Password encryption/decryption services
- [ ] Application management system
- [ ] Category and folder organization
- [ ] Bulk import/export functionality
- [ ] Password sharing capabilities (if required)

Phase 3: Security & Audit Features (Week 7-8)

Security Implementation

- [ ] Encryption key rotation system
- [ ] JWT blacklist management
- [ ] Rate limiting and brute force protection
- [ ] Input validation and sanitization
- [ ] SQL injection prevention
- [ ] XSS protection measures

Audit & Logging

- [ ] Comprehensive audit logging system
- [ ] User activity tracking
- [ ] Failed login attempt monitoring
- [ ] Password access logging
- [ ] Security event notifications
- [ ] Log retention and cleanup procedures

Phase 4: Frontend Development (Week 9-12)

User Interface

- [ ] User registration and login forms
- [ ] Dashboard with password overview

- [ ] Password management interface
- [ ] Application/service management
- [ ] Search and filter functionality
- [ ] Password generator tool

Security Features UI

- [ ] Two-factor authentication setup
- [ ] Security settings panel
- [ ] Audit log viewer
- [ ] Session management interface
- [ ] Password strength indicators
- [ ] Secure password sharing interface

Phase 5: Testing & Quality Assurance (Week 13-14)

Security Testing

- [ ] Penetration testing
- [ ] Vulnerability assessment
- [ ] Encryption validation testing
- [ ] JWT security testing
- [ ] Session management testing
- [ ] Input validation testing

Functional Testing

- [ ] Unit testing for all APIs
- [ ] Integration testing
- [ ] User acceptance testing
- [ ] Performance testing
- [ ] Load testing
- [ ] Cross-browser compatibility testing

Phase 6: Deployment & Production Setup (Week 15-16)

Infrastructure

- [ ] Production server setup
- [ ] SSL/TLS certificate installation
- [ ] Database production configuration

- [ ] Backup and disaster recovery setup
- [ ] Monitoring and alerting system
- [ ] Security hardening procedures

Go-Live Preparation

- [ ] Data migration procedures (if applicable)
- [ ] User training documentation
- [ ] Production deployment
- [ ] Post-deployment verification
- [ ] Performance monitoring setup
- [ ] Security monitoring implementation

Technical Stack Considerations

Backend Options

- **API Framework:** Node.js (Express), Python (FastAPI/Django), Java (Spring Boot), or C# (.NET)
- **Authentication:** JWT with refresh token rotation
- **Encryption:** AES-256 for password storage, bcrypt/Argon2 for master passwords
- **Database:** MySQL (already established)

Frontend Options

- **Web Application:** React, Vue.js, or Angular
- **Mobile Apps:** React Native, Flutter, or native development
- **Desktop Apps:** Electron, Tauri, or native applications

Security Standards

- **Compliance:** Consider GDPR, CCPA, SOC 2 requirements
- **Encryption:** End-to-end encryption for sensitive data
- **Authentication:** Multi-factor authentication support
- **Auditing:** Comprehensive logging and monitoring

Risk Assessment & Mitigation

High-Risk Areas

1. **Data Breaches:** Implement zero-knowledge architecture

2. **Password Exposure:** Use client-side encryption
3. **Authentication Bypass:** Multi-layered security validation
4. **Database Compromise:** Encrypt sensitive data at rest
5. **Session Hijacking:** Secure JWT implementation with short expiration

Mitigation Strategies

- Regular security audits and penetration testing
- Implement security headers and HTTPS everywhere
- Use prepared statements to prevent SQL injection
- Implement proper error handling without information disclosure
- Regular backup testing and disaster recovery drills

Success Metrics

Security Metrics

- Zero successful unauthorized access attempts
- 100% password encryption coverage
- Complete audit trail for all sensitive operations
- Sub-second authentication response times
- 99.9% uptime availability

User Experience Metrics

- User registration completion rate > 95%
- Password retrieval success rate > 99%
- Average response time < 200ms
- User satisfaction score > 4.5/5
- Support ticket volume < 1% of active users

Budget & Resource Considerations

Development Team

- **Backend Developer(s):** 2-3 developers
- **Frontend Developer(s):** 2-3 developers

- **Security Specialist:** 1 consultant/developer
- **DevOps Engineer:** 1 engineer
- **QA Tester:** 1-2 testers

Infrastructure Costs

- Database hosting and backup solutions
- SSL certificates and security tools
- Monitoring and logging services
- Development and staging environments
- Third-party security services (penetration testing)

Next Immediate Steps

1. **Execute database table creation** using the provided SQL scripts
2. **Choose technology stack** for backend and frontend development
3. **Set up development environment** with proper security configurations
4. **Begin user authentication system** development
5. **Establish security protocols** and coding standards
6. **Create project repository** with proper branch protection and code review processes

This plan provides a comprehensive roadmap for building a secure, enterprise-grade password management system with all the features indicated by your database design.