**Project Documentation: FinMate360 - Unified Fintech SaaS Platform**

**1. Project Overview**

**Project Name:** FinMate360  
**Objective:** To develop a unified financial management platform integrating personal budgeting, group expense tracking, invoicing, micro-investments, and credit tracking services into a single scalable SaaS product.  
**Scope:**

* Centralize multiple financial utilities
* Leverage AI for smarter insights
* Scalable for individual users, small businesses, and enterprises

**Technology Stack:**

* Backend: ASP.NET Core (.NET 8)
* Database: SQL Server
* ORM: Entity Framework Core
* API: RESTful APIs secured with JWT

**2. Product Modules & Functionalities**

**2.1 Budget & Goal Manager (BudgetPal)**

* Link UPI/Banks to auto-fetch transactions
* Auto-generate and customize budget
* Set and monitor financial goals
* Analyze actuals vs planned expenditure
* AI-driven financial saving suggestions *(Future)*

**2.2 Group Expense Splitter (BillSplitz)**

* Create and manage groups
* Add expenses, calculate splits
* UPI-based settlement
* Voice input & reminders *(Future)*
* Subscription expense manager *(Future)*

**2.3 Invoice & GST Management (GSTMate)**

* Freelancers/SMEs invoice generation
* GST calculation & report generation
* Share via email/PDF/WhatsApp *(Future)*
* CA portal access *(Future)*

**2.4 Micro-Investment Engine (MicroFund India)**

* Round-off and invest spare change
* Track performance (Mutual Funds, Gold)
* Auto-SIP setup *(Future)*
* RBI-compliant wallet integration *(Future)*

**2.5 Credit Score Manager (CreditBoost)**

* Track credit score from CIBIL/Experian
* Credit health dashboard
* Improvement suggestions
* Credit card/loan recommendations *(Future)*
* AI credit bot *(Future)*

**3. Architecture**

**3.1 Clean Architecture Layers**

FinMate360.sln

├── API Layer (Controllers, Filters)

├── Application Layer (Services, Use Cases, DTOs)

├── Domain Layer (Entities, Interfaces, Enums)

├── Infrastructure Layer (Repositories, EF Core DB Context)

├── Shared Layer (Constants, Utilities, Helpers)

**3.2 Authentication & Authorization**

* JWT Auth
* RBAC (Role Based Access Control)
* 2FA support via Email/SMS
* Refresh tokens
* Secure password hashing (BCrypt)

**4. Database Design**

**4.1 Core Tables (120+ Estimated)**

|  |  |
| --- | --- |
| **Table Name** | **Description** |
| Users | User info & roles |
| Transactions | Linked bank/UPI transactions |
| Budgets | Auto-generated/user budget plans |
| Goals | Savings goals data |
| Groups | Expense split groups |
| GroupExpenses | Expenses within a group |
| Invoices | Freelance/SME invoice data |
| InvoiceItems | Line items per invoice |
| GSTReports | Tax summaries per period |
| Investments | Micro investment logs |
| CreditScores | History of pulled scores |
| Notifications | Alerts and system notifications |

**5. API Endpoints (Sample)**

|  |  |  |  |
| --- | --- | --- | --- |
| Module | Endpoint | Method | Description |
| Auth | /api/auth/login | POST | Login user |
| Budget | /api/budget/create | POST | Create budget |
| Group | /api/group/create | POST | Create group |
| Invoice | /api/invoice/generate | POST | Generate invoice |
| GST | /api/gst/report/{month} | GET | View GST report |
| Investment | /api/investment/roundup | POST | Micro-invest investment |
| Credit | /api/credit/score | GET | Get credit score info |

**6. Development Phases**

**Phase 1: Core Platform**

* Auth
* User Profile
* Transaction Import
* Budget Module

**Phase 2: Expense Split & Credit Score**

* Group Expense Management
* Credit Score Tracking

**Phase 3: Invoice & Tax Management**

* Invoicing System
* GST Calculation and Reports

**Phase 4: Investment Platform**

* Spare Change Rounding
* SIP, Wallet, Dashboard

**7. External Integrations (Planned)**

* UPI APIs (PhonePe/Paytm/GPay)
* CIBIL/Experian API
* Mutual Fund APIs (Groww/Zerodha)
* WhatsApp Business API
* AI API (OpenAI, Local ML)

**8. DevOps & Deployment**

* CI/CD: GitHub Actions or Azure DevOps
* Containerization: Docker
* Hosting: Azure App Services / AWS ECS
* Database: Azure SQL / On-prem SQL Server

**9. Logging & Monitoring**

* Logging: Serilog with SQL Sink
* Error Handling: Global Exception Middleware
* Health Checks: ASP.NET Core HealthChecks
* Background Services: Cron Jobs for Scheduled Reports, AI Bot Updates

**10. Naming Conventions**

* **Table Names:** PascalCase
* **Column Names:** camelCase
* **Services/Interfaces:** I + ServiceName (e.g. IBudgetService)
* **Controllers:** [Module]Controller

**11. Next Steps**

1. Prepare detailed ER diagram (Phase 1)
2. Set up SQL Server with seed schema
3. Start .NET Core API Project
4. Implement modules incrementally with unit tests

**12. Conclusion**

This document serves as the **complete professional documentation** of FinMate360 - a powerful, scalable, and future-proof Fintech platform. The phased approach allows for MVP creation with expandability into full-scale SaaS product targeting millions of users.