Build FinPay: A FinTech Solution

A PRD draft for "Build FinPay: A FinTech Solution"

MASTERINGBACKEND

Masteringbackend

Table Of Contents

Introduction

The "Build FinPay: A FinTech Solution" project involves developing a comprehensive FinTech application using any backend language and various FinTech APIs.

The application will provide users with functionalities to create and manage invoices, generate virtual cards, create virtual accounts, and facilitate payments in different currencies.

Users will have access to a dashboard to track their balances, manage transactions, create invoices, handle wallets, and set up notifications.

The platform will also allow for currency conversion, withdrawal of funds, and viewing of account statements.

This project will be a full-featured FinTech solution that caters to both business and personal financial needs.

Objectives

- Build a secure FinTech application that allows users to create and manage financial transactions and accounts.
- Integrate with external FinTech APIs to provide real-time currency conversion and virtual card generation.
- Enable comprehensive user management with authentication, registration, and profile editing capabilities.
- Implement a notification system to keep users informed about their account activities and transaction statuses.
- Provide advanced data management features like search, filter, and pagination for invoices, transactions, and cards.

Functional Requirements

User Management

- **Implement Login Feature:** Allow users to log in using email and password.
- I**mplement Register Feature:** Enable new users to create accounts by registering with their email, password, and additional details.
- **Implement Forgot Password:** Allow users to reset their password by sending a reset link to their registered email.
- **Implement Logout Feature:** Provide the ability for users to log out of the application.

Invoices

- **Create New Invoice**: Allow users to create new invoices by specifying details like amount, currency, due date, and description.
- **View All Invoices**: Retrieve and display all invoices created by the user.
- **View Draft Invoices**: Retrieve and display all draft invoices that have not been sent.
- **View Pending Invoices**: Retrieve and display invoices that are pending payment.
- **Retrieve Due Invoices**: Display invoices that are due for payment soon.
- **Retrieve Overdue Invoices**: Display invoices that are past their due date and have not been paid.
- **Filter All Invoices**: Implement filtering functionality to narrow down invoices by status, amount, or date.
- **Search All Invoices**: Provide search functionality to find invoices by keywords or other attributes.
- **Delete an Invoice**: Allow users to delete invoices that are no longer needed.

Cards

- **Create New Card**: Allow users to generate new virtual cards with specific limits and usage preferences.
- **Retrieve All Cards**: Display all virtual cards associated with the user's account.
- **Retrieve a Card**: View details of a specific card by ID.
- **Delete a Card**: Allow users to delete a virtual card that they no longer need.

Wallets

- **Retrieve Current Balance**: Display the user's current balance for each currency in their wallet.
- **Retrieve and Export Account Statement**: Allow users to retrieve and export their account statements in PDF or CSV format.
- **Switch Currencies**: Enable users to switch between different currencies in their wallet.
- **Retrieve Receiving Account Details**: Show details of the account receiving payments in different currencies.
- **Retrieve Total Incomes & Expenses**: Summarize total incomes and expenses for a given period.
- **Send Money to Recipient**: Enable users to send money to a recipient's account.
- **Convert Currencies**: Allow users to convert one currency to another based on current exchange rates.
- **Fund Your Account**: Enable users to add funds to their wallet through bank transfer or credit card.
- **Withdraw Your Fund**: Allow users to withdraw funds from their wallet to an external account.

Transactions

- **Retrieve All Transactions**: Display a list of all transactions, including payments, conversions, and receipts.
- **Search Transactions**: Provide search functionality to find transactions by keywords or attributes.
- **Filter Transactions**: Implement filters to narrow down transactions by date, amount, type, or status.
- **Paginate Transactions**: Paginate through a large list of transactions for easy viewing.
- **Delete a Transaction**: Allow users to delete a transaction if needed.
- **Retrieve a Transaction**: Retrieve details of a specific transaction by its ID.

Profile

- **Retrieve Profile Details**: Allow users to view their profile information.
- **Edit Profile Details**: Enable users to update their profile information like name, address, and contact details.
- **Add New Beneficiary**: Add a new beneficiary to the user's profile for payments or transfers.
- **Search Beneficiaries**: Provide search functionality to find beneficiaries by name or other attributes.
- **Retrieve a Single Beneficiary**: Retrieve details of a specific beneficiary by ID.
- **Delete Beneficiary**: Allow users to delete a beneficiary from their profile.
- **Activate 2FA**: Implement two-factor authentication for enhanced account security.
- **Implement Verification Process**: Include steps for verifying user identity (e.g., KYC) before they can access certain features.

Non-Functional Requirements

- **Scalability:** The application should handle a large number of users and concurrent transactions efficiently.
- **Performance:** Implement strong security measures to protect user data and financial transactions.
- **Security:** Ensure the application provides fast response times for dashboard updates and API requests.
- **Reliability:** Maintain high availability and handle failures gracefully.
- **Usability:** The API should be easy to use and well-documented.

Use Cases

- **Create Invoice:** A user creates an invoice for a client, specifying the amount, currency, and due date.
- **Generate Virtual Card:** A user generates a virtual card for online payments with spending limits.
- **Retrieve Transaction History:** A user views their transaction history, applies filters, and exports statements.

User Stories

- **As a user, I want to register and create an account** so that I can start using the platform to manage my financial operations.

- **As a user, I want to log in to my account** so that I can access my dashboard and view my financial data.

- **As a user, I want to create and send invoices** so that I can bill my clients and track payments.

- **As a user, I want to view and manage my invoices** so that I can keep track of paid, pending, and overdue invoices.

- **As a user, I want to create and manage virtual cards** so that I can use them for online payments and set spending limits.

- **As a user, I want to create a virtual account** so that I can receive payments in different currencies.

- **As a user, I want to retrieve and view my current account balance** so that I know how much money I have in my wallet.

- **As a user, I want to convert currencies** so that I can have the desired currency available for transactions.

- **As a user, I want to send money to recipients** so that I can pay vendors or transfer funds to beneficiaries.

- **As a user, I want to retrieve and export my account statements** so that I can have a record of my financial activities.
- **As a user, I want to search and filter transactions** so that I can quickly find specific transactions or view historical data.
- **As a user, I want to view current exchange rates** so that I can decide when to convert currencies or make international payments.
- **As a user, I want to retrieve my profile details and update my information** so that I can keep my account information up-to-date.
- **As a user, I want to set up two-factor authentication (2FA)** so that I can add an extra layer of security to my account.
- **As a user, I want to receive notifications and alerts** so that I can stay informed about important activities like incoming payments or security warnings.
- **As an admin, I want to view detailed analytics and reports** so that I can track platform performance, user engagement, and financial metrics.

Technical Requirements

- **Programming Language:** Use a backend language like Node.js with Express.js for handling API requests and managing server logic.
- **Database:** Use a relational database like PostgreSQL or MySQL to store user data, transaction history, invoices, and card details.
- **Cache Storage:** Utilize caching mechanisms like Redis to speed up frequent queries and reduce database load for data like exchange rates or user session data.
- **Payment Gateway:** Integrate a payment processing service like Stripe or PayPal to handle payments for invoices and adding funds to wallets.
- **Currency Conversion API:** Use a third-party currency conversion API (e.g., OpenExchangeRates) to get real-time exchange rates for multi-currency support.
- **Cloud Services:**Use cloud services like AWS or Google Cloud for video storage, compute power, and scalable storage solutions.
- **API Documentation:** Use Swagger or Postman to document the API, making it easy for developers to understand and integrate the endpoints.

API Endpoints

User Management

- **POST /signup:** Register a new user.
- **POST /login:** Authenticate a user.
- **GET /profile:** Get user profile details.
- **PUT /profile:** Update user profile.

Dashboard

- **GET / api/dashboard/balances:** Retrieve All Balances.
- **GET /api/dashboard/receiving-account** Get Receiving Payment Account.

Invoices

- **POST /api/invoices:** Create Invoice.
- **GET /invoices/{id}:** Get an invoice.

Cards

- **POST /cards:** Create Virtual Card.
- **GET /cards:** Retrieve all cards.

Wallets

- **POST /wallets/fund:** Fund your wallet.

Security

- **Data Encryption:** Ensure that sensitive data such as user passwords and card information is encrypted in transit and at rest.
- **Authentication**: Use token-based authentication (JWT) for all API requests to ensure only authorized users can access their data.
- **Role-based Access Control**: Implement role-based access control to restrict access to admin, user, and guest functions as appropriate.
- **Input Validation:** Validate all user inputs to prevent SQL injection, cross-site scripting (XSS), and other forms of attack.

Performance

- **API Rate Limiting**: Implement rate limiting to prevent abuse and ensure fair use of the API.
- **Caching**: Use caching for frequently accessed data like exchange rates and profile information to reduce server load.
- **Database Indexing**: Index database tables on frequently queried fields to improve read performance.

Documentation

- Provide API documentation using tools like Swagger to help developers integrate the app with third-party services or expand on the platform.
- Include detailed request and response examples, error codes, and descriptions for each endpoint.

Glossary

- **Virtual Card:** A digital card that functions like a physical debit or credit card but exists only electronically.
- **Invoice:** A document requesting payment for services or goods provided, including details like amount, due date, and description.
- **Currency Conversion:** The process of converting an amount of money from one currency to another based on the current exchange rate.
- **2FA (Two-Factor Authentication):** An additional layer of security that requires not only a password and username but also something that only the user has on them (e.g., a smartphone app) before they can access their account.

Appendix

Include flowcharts or diagrams showing the user flow through various functionalities, such as creating invoices, managing virtual cards, and handling transactions.

Want more backend projects?

https://projects.masteringbackend.com

Join our community

Need to show-off or ask doubts? Join our Slack Community. Ask questions, help others and learn in public to make the best use of MBProject.

https://masteringbackend.com/community