# Payment Gateway System

**Project Overview**

The payment gateway system is designed to facilitate online

payment processing for merchants while ensuring security

and fraud prevention. It serves as the interface between

online merchants and banks, processing various types of

transactions, including purchases and cancellations.

**Key Features**

* **Transaction Processing**: Handles various
* transaction types,
* including purchases, cancellations, and risk
* validation.
* **Fraud Prevention**: Implements checks based
* on credit card
* information and geographical locations to
* identify and block high-risk transactions.
* **Multi-Bank Integration**: Supports multiple
* banks, accommodating different API standards
* to serve merchants in various countries.
* **Reporting and Analytics**: Generates reports
* for management to assess transaction activity and risk metrics.

**Code Structure**

* **Controllers**:
  + GatewayController: Manages transaction

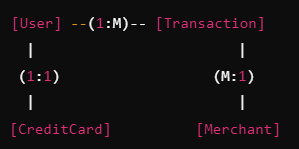
requests and responses.

* + ReportController: Handles report

generation for transaction analytics.

* **Models**:
  + Transaction
  + TransactionStatus
  + ServerURLs
  + Bank
  + Merchant
  + Payment
  + MerchantPayment
  + Country
  + User
  + Report
  + RiskManagement
  + AuditLog
* **Services**:
  + TransactionBackgroundService: Processes transactions and handles fraud checks.
  + ReportBackgroundService: Generates daily reports based on transaction data.

**ERD for Payment Gateway:**



Warning!!!

        GetServerUrlByIp can work fully on-promises with special project and local database contain IP address range for entire world country.

        Please research if we need to not use 3th party providers like "ipapi.co". What if this service stopped?

        This approach is very simply just for the tests and not for real production.

**Prerequisites**

1. .NET 8 SDK installed.
2. Install the necessary NuGet packages:
   * **EF Core**: Microsoft.EntityFrameworkCore
   * **EF Core SQL Server**: Microsoft.EntityFrameworkCore.SqlServer
   * **EF Core Tools**: Microsoft.EntityFrameworkCore.Tools

**1. Set Up the Project**

1. Create a new Web API project in .NET 8:

**API Flow**

1. **Initiate Transaction**:
   * The client calls /api/transaction/initiate and receives a GUID.
   * The transaction is stored in the database with a Pending status.
2. **Simulate Bank Response** (optional):
   * The system (e.g., a background job or periodic task) runs the SimulateBankResponse stored procedure to randomly assign statuses to transactions.
3. **Check Transaction Status**:
   * The client calls /api/transaction/status/{transactionGuid} to check the transaction’s current status (e.g., "Approved", "Insufficient Funds", etc.).

#### System Interface

The request and response formats (in JSON) for the payment gateway:

* **Request**:

json

Copy code

{

"merchant\_id": "12345",

"amount": 100.00,

"currency": "USD",

"credit\_card": {

"number": "4111111111111111",

"expiry": "12/2025",

"cvv": "123"

},

"client\_info": {

"client\_id": "98765",

"email": "user@example.com"

}

}

* **Response**:

json

Copy code

{

"transaction\_id": "ABC123",

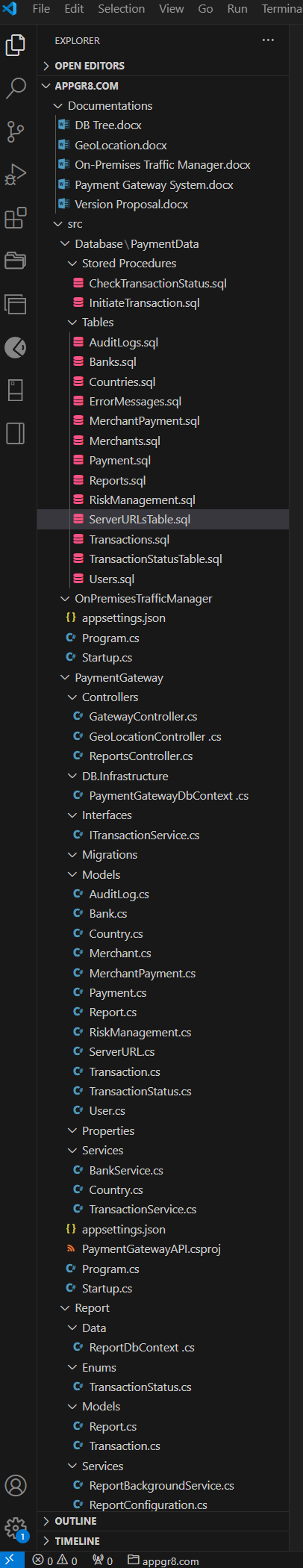
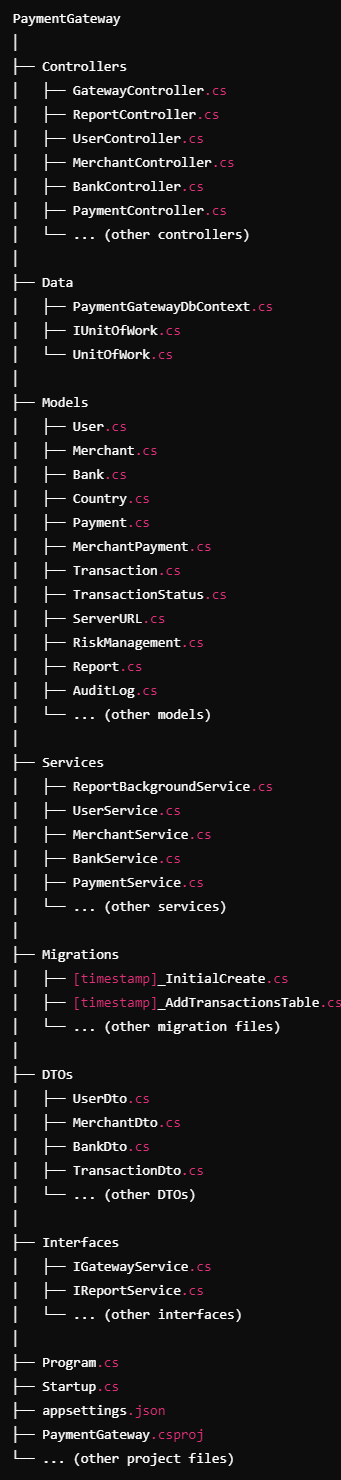
"status": "approved",

"message": "Transaction approved",

"fee": 0.5

}

The system co

**Database Structure**

* **Tables**:
  + Transactions: Stores transaction details (including AccountCurrency).
  + TransactionStatus: Contains possible transaction statuses (e.g., successful, failed).
  + ServerURLs: Maps server URLs based on geographical location.
  + Banks: Contains information about integrated banks.
  + Merchants: Stores merchant details.
  + Payment: Manages payment information.
  + MerchantPayment: Links merchants to payment methods.
  + Countries: Holds country data for risk analysis.
  + Users: Stores user information.
  + Reports: Keeps records of reports generated.
  + RiskManagement: Contains risk-related information.
  + AuditLogs: Records audit trails for transactions.

**Technical Documentation**

* **ERD (Entity-Relationship Diagram)**: Visual representation of the database schema and relationships between tables.
* **API Specifications**: Detailed endpoints for transaction processing, risk validation, and report generation, including request and response formats in JSON.