LightSpeedPay - Payment Gateway Documentation

Table of Contents

Introduction

LightSpeedPay is a payment gateway that allows merchants to process payments efficiently. This documentation provides an overview of the system architecture, the key components, and instructions for setting up and maintaining the system.

The application is built using **Node.js** and **Express.js** for the server-side logic, and **MongoDB** as the database. It includes various RESTful APIs for managing merchants, transactions, wallets, banks, and pricing.

Technology Stack

- **Node.js** (v16.x+)
- **Express.js** (v4.x)
- **MongoDB** (v6.x) - for storing payment data and merchant info
- **Axios** - for handling HTTP requests
- **Mongoose** - for MongoDB object modeling
- **Redis** - used for caching and improving performance
- **Jest & Supertest** - for testing the APIs
- **Nodemailer** - for sending emails

Folder Structure

The application follows a standard Node.js and Express structure:

```
├── app.js             # Main app setup
├── server.js          # Server entry point
├── routes/            # API routes
│   ├── authRoute.js
│   ├── merchantRoute.js
│   ├── transactionRoute.js
│   ├── walletRoute.js
│   └── ...
├── controllers/       # Business logic for each route
│   ├── authController.js
│   ├── merchantController.js
│   ├── transactionController.js
│   └── ...
├── models/            # Mongoose models
│   ├── Merchant.js
│   ├── Transaction.js
│   └── ...
├── middleware/        # Middleware functions
│   ├── authMiddleware.js
│   ├── requestLogger.js
├── config/            # Database connection and environment config (not
included in current setup)
├── seeder.js          # Seeding script for importing/destroying data
├── tests/             # Test cases
│   ├── auth.test.js
│   └── ...
└── utils/             # Utility functions
```

Configuration

You need to create a `.env` file at the root of the project for environment-specific settings. The required variables are mentioned below in the **Environment Variables** section.

Running the Application

1. **Install dependencies:**

   ```
   npm install
   ```

2. **Run in development mode:**

   ```
   npm run dev
   ```

   This will start the application with **nodemon** on `PORT 5000` or the port specified in the `.env` file.

3. **Run in production mode:**

```
npm start
```

API Routes

Here is a list of the key routes available in LightSpeedPay.

- **Authentication**
  - POST `/api/v1/auth/login` - Login as merchant/admin
  - POST `/api/v1/auth/register` - Register a new merchant
- **Merchants**
  - GET `/api/v1/merchant` - Get all merchants
  - POST `/api/v1/merchant` - Create a new merchant
  - GET `/api/v1/merchant/sandbox` - Access sandbox merchants for testing
- **Transactions**
  - GET `/api/v1/transaction` - Get all transactions
  - POST `/api/v1/transaction` - Create a new transaction
  - GET `/api/v1/transaction/sandbox` - Sandbox for test transactions
- **Wallet**
  - GET `/api/v1/wallet` - Get wallet balance
  - POST `/api/v1/wallet/topup` - Top up wallet
- **Bank Accounts**
  - GET `/api/v1/bank` - Get bank accounts linked with a merchant
  - POST `/api/v1/bank` - Add a bank account
- **Admin**
  - GET `/api/v1/admin` - Admin operations
  - POST `/api/v1/adminBank` - Add bank details for admin
- **Pricing**
  - GET `/api/v1/price` - Get pricing details
  - POST `/api/v1/price` - Update pricing for transactions

Middleware

The following middleware is used in the application:

1. **authMiddleware.js**: Handles JWT authentication and protection of routes.
    - `protect` middleware is used to protect specific routes for merchants or admins.
2. **requestLogger.js**: Logs incoming requests and is used only in non-testing environments.

Environment Variables

The application requires the following environment variables to function. These should be placed in the `.env` file at the root of the project.

```
MONGO_URI="mongodb+srv://<user>:<password>@cluster0.mongodb.net/?
retryWrites=true&w=majority"
PORT=6001
JWT_SECRET="your-secret-key"
ICICI_API_URL="https://icici-uat-v2.mitrafintech.com"
IS_READONLY=0
OTP_URL="https://api.gupshup.io/wa/api/v1/template/msg"
OTP_URL_KEY="your-otp-url-key"
```

Testing

The application uses **Jest** and **Supertest** for testing. To run the tests, use the following command:

```
npm run test
```

For Windows:

```
npm run test:win
```

Database Setup

To connect to MongoDB, the application requires the `MONGO_URI` environment variable. You can use a cloud MongoDB instance or a local MongoDB setup.

- Example MongoDB connection string:

  ```
  mongodb+srv://<user>:<password>@cluster0.mongodb.net/payment-gateway
  ```

- The database connection is initiated in `server.js`:

```
const connectDB = require('./config/db');
connectDB();
```

Seeding Data

You can seed the database with initial data using the `seeder.js` script.

- **Import data:**

```
npm run data:import
```

- **Destroy data:**

```
npm run data:destroy
```

Error Handling

All unhandled routes return a 404 error:

```
app.all("*", (req, res) => {
  const err = new Error(`Can't find ${req.originalUrl} on this server!`);
  err.statusCode = 404;
  res.status(err.statusCode).json({
    status: 'fail',
    error: err.message,
  });
});
```

Global error handling middleware can be added in the future for more complex error management.

Important Notes

1. **CORS**: The application uses `cors` to allow cross-origin requests. You can configure the allowed domains in `app.js`.
2. **Read-only mode**: The application has an `IS_READONLY` flag to toggle between read-only and write modes.