Expense Tracker App - Microservices Architecture Documentation

Overview

This document provides an in-depth overview of the Expense Tracker application built with a microservices architecture. The application is designed to efficiently manage user expenses and includes features such as wallet management, transaction logging, monthly summaries, and filtering options. The architecture leverages Kong Gateway for secure API routing, Prometheus and Grafana for monitoring, and MongoDB as the primary data store.

1 Architecture Overview

The application is structured as a set of loosely coupled microservices. Each service performs a specific function and can be scaled independently, allowing for high modularity and maintainability. The main services are:

- **Frontend (ReactJS)**: Runs on port 8080 and serves the user interface for managing expenses.
- **Backend (Node.js)**: Runs on port 8080 and provides a REST API for frontend interactions.
- **Database (MongoDB)**: Stores user data, transactions, and wallet information. Exposed on port 27017.
- **Kong Gateway**: A reverse proxy and API gateway, running on port 8443, routes requests securely.
- **Prometheus**: Collects backend metrics from the /metrics endpoint, running on port 9090.
- **Grafana**: Provides a dashboard to visualize metrics, running on port 3000.

2 Components and Responsibilities

2.1 Frontend (ReactJS)

- **Purpose**: Provides the user interface for the application.
- **Port**: 8080
- **Main Features**:
  - Wallet management
  - Transaction logging
  - Monthly summary and transaction filters
- **Docker**: Containerized, exposing port 8080.

2.2 Backend (Node.js)

- **Purpose**: Provides RESTful APIs for frontend requests.
- **Port**: 8080
- **Endpoints**:
  - `/api/transactions`: CRUD operations for transactions.
  - `/api/wallets`: CRUD operations for wallets.
  - `/metrics`: Exposes Prometheus-compatible metrics.
- **Docker**: Containerized, exposing port 8080.
- **Security**: Communicates with Kong Gateway via TLS.

2.3 Database (MongoDB)

- **Purpose**: Stores application data.
- **Port**: 27017
- **Configuration**: Hosted in a Docker container, stores JSON documents.

2.4 Kong Gateway

- **Purpose**: API gateway, securing and routing requests.
- **Port**: 8443
- **Configuration**:
  - PostgreSQL for configuration storage.
  - Routes requests securely with TLS.
- **Docker**: Configured and managed via Docker.

2.5 Prometheus

- **Purpose**: Collects metrics from backend.
- **Port**: 9090
- **Docker**: Configured in Docker.

2.6 Grafana

- **Purpose**: Visualizes data from Prometheus.
- **Port**: 3000
- **Docker**: Configured with dashboards for expense tracking.

3 Inter-Service Communication

- **Frontend to Backend**: Routed through Kong Gateway on port 8443.
- **Backend to MongoDB**: Direct connection on port 27017.
- **Prometheus to Backend**: Scrapes /metrics endpoint for data.
- **Grafana to Prometheus**: Pulls metrics data for dashboards.

4 Docker Setup

Each service is containerized. Example Dockerfiles and docker-compose configuration are below.

4.1 Dockerfile for React Frontend

```
# Dockerfile for React
FROM node:14-alpine
WORKDIR /app
COPY package.json ./
RUN npm install
COPY . .
EXPOSE 8080
CMD ["npm", "start"]
```

4.2 Dockerfile for Node.js Backend

```
# Dockerfile for Node.js backend
FROM node:14-alpine
WORKDIR /app
COPY package.json ./
RUN npm install
COPY . .
EXPOSE 8080
CMD ["node", "server.js"]
```

4.3 docker-compose.yml

```
version: '3.8'
services:
  frontend:
    build:
      context: ./frontend
    ports:
      - "8080:8080"

  backend:
    build:
      context: ./backend
    ports:
      - "8080:8080"
    environment:
      MONGODB_URI: "mongodb://database:27017/expenses"

  database:
    image: mongo
    ports:
      - "27017:27017"

  kong:
    image: kong
    ports:
      - "8443:8443"

  prometheus:
    image: prom/prometheus
```

```
    ports:
      - "9090:9090"

  grafana:
    image: grafana/grafana
    ports:
      - "3000:3000"
```

5 Kong Gateway Configuration

To configure Kong Gateway, set up routes, services, and consumers with the following commands:

```
# Configure Kong service
curl -i -X POST http://localhost:8001/services/ \
    --data "name=backend" \
    --data "url=http://backend:8080"

# Configure route for service
curl -i -X POST http://localhost:8001/services/backend/routes \
    --data "paths[]=/api" \
    --data "strip_path=true"
```

6 Prometheus and Grafana Setup

- **Prometheus Configuration**: Update the config file to scrape metrics from the backend.
- **Grafana Setup**: Add Prometheus as a data source and import a dashboard for metrics visualization.

7 Security Configuration

Enable TLS in Kong Gateway to secure communication between frontend and backend.

8 Environment Variables and Configuration

Include environment variables in a `.env` file for ease of configuration.