Task 1-2: Case Study

Build a Simple Billing App for a SaaS Platform Using Cloudflare Workers

Problem Statement:

Your task is to design and implement a billing app for a SaaS platform **that fully utilizes Cloudflare Workers**. The app should support multiple subscription tiers and handle recurring billing **using TypeScript**. Cloudflare Workers will provide a serverless environment that is highly performant, scalable, and cost-effective for handling all backend processes, including API requests, data storage, and business logic execution.

Requirements:

Core Features:

1. **Subscription Management**:
   - Create and manage subscription plans with different pricing and billing cycles.
   - Assign subscription plans to customers and manage their subscription status.
2. **Billing Engine**:
   - Automatically generate invoices at the end of each billing cycle based on the customer's subscription plan.
   - Handle prorated billing for mid-cycle upgrades or downgrades.
3. **Payment Processing**:
   - Record payments made by customers and update invoice status accordingly.
   - Handle failed payments and implement retry logic using Cloudflare Workers' scheduled cron triggers.
4. **Notifications**:
   - Send email notifications to customers when an invoice is generated, when a payment is successful, or when a payment fails using an integrated email service like SendGrid or Mailgun.

Additional Features:

- **Invoice Generation Function**:
  - Implement the logic to generate invoices using Cloudflare Workers as a serverless function.
  - Trigger the invoice generation function via HTTP requests or Cloudflare Workers' scheduled events (cron jobs).
  - The function should calculate the invoice amount based on a customer's current subscription plan and update the invoice status to "generated."
- **Data Storage**:
  - Use Cloudflare Workers KV or Durable Objects for storing subscription data, customer information, invoices, and payments.
  - Leverage Durable Objects to maintain stateful data for billing cycles and customer sessions.
- **API Endpoints**:
  - Provide the required endpoints to handle actions such as creating a subscription plan, assigning a subscription plan to a customer, generating an invoice, processing a payment, listing invoices for a customer, and retrieving subscription details. Implement these endpoints using TypeScript and Cloudflare Workers' serverless environment.

Entities:

The billing app will consist of several key entities, each with specific fields that can be modified or expanded based on implementation needs.

1. **Customer**:
   - `id`: Unique identifier for the customer.
   - `name`: Customer's name.
   - `email`: Customer's email address.
   - `subscription_plan_id`: The current subscription plan the customer is on.
   - `subscription_status`: Current status of the subscription (e.g., active, cancelled).
2. **SubscriptionPlan**:
   - `id`: Unique identifier for the subscription plan.
   - `name`: Name of the plan (e.g., Basic, Pro, Enterprise).
   - `billing_cycle`: The billing cycle of the plan (e.g., monthly, yearly).
   - `price`: Price of the plan.
   - `status`: Status of the plan (active/inactive).
3. **Invoice**:
   - `id`: Unique identifier for the invoice.
   - `customer_id`: The customer associated with the invoice.
   - `amount`: Total amount due.
   - `due_date`: The date the payment is due.
   - `payment_status`: Status of the payment (e.g., pending, paid, failed).
   - `payment_date`: The date when the payment was received.
4. **Payment**:
   - `id`: Unique identifier for the payment.
   - `invoice_id`: The invoice associated with the payment.
   - `amount`: Amount paid.
   - `payment_method`: Method of payment (e.g., credit card, PayPal).
   - `payment_date`: Date when the payment was made.

Evaluation Criteria:

- Code quality and structure.
- Effective use of Cloudflare Workers for serverless computing.
- Understanding of subscription management and billing processes.
- Handling of edge cases such as failed payments and mid-cycle plan changes.
- Basic documentation and testing, including unit tests for key functions.

Deliverables:

1. A GitHub repository with the TypeScript code for the Cloudflare Worker-based billing app.
2. A README file explaining how to deploy and run the project on Cloudflare Workers, along with a description of the design approach.
3. Basic API documentation outlining the available endpoints, expected inputs, outputs, and error handling strategies.

Task 2-2 : Code Refactoring

For the following task, identify at least 3 problems in the code.

Problematic Code:

```javascript
app.get('/product/:productId', (req, res) => {
    db.query(`SELECT * FROM products WHERE id=${req.params.productId}`, (err, result) => {
        if (err) throw err;
        res.send(result);
    });
});
```