Exercise 1 - Ticket Categorization

1. **Query 1**:
   Category: **Bug**
   Response:
   Hi,
   I understand you're experiencing an issue with the 'Add Product' button being grayed out. This seems like a bug. Our team will look into it. In the meantime, could you try clearing your browser cache or using a different browser to see if that helps? Please let us know if the issue persists.

2. **Query 2**:
   Category: **Feature Request**
   Response:
   Hi,
   Thank you for your suggestion! Currently, Zenskar does not support automated monthly summaries of invoices. However, we've noted this as a feature request and will evaluate it for future releases. Let us know if you have any other questions.

3. **Query 3**:
   Category: **Customer Training**
   Response:
   Hi,
   I'd be happy to help you with configuring the usage-based billing schema. Here's a guide on how to set it up: [Insert link to guide]. If you need further assistance, feel free to reach out, and we can schedule a training session.

Exercise 2 - Onboarding Plan

1. **Onboarding Plan**:
   Step 1: Configure the client's Zenskar account.
   - Provide login credentials and guide them through setting up their account.
   - Set up payment methods, billing cycle, and any necessary integrations.

   Step 2: Upload the client's first set of usage data.

   - Provide a step-by-step guide on how to upload data to the platform.
   - Offer assistance during the first upload to ensure smooth execution.

   Step 3: Generate the client's first invoice.

  - Guide them through creating an invoice based on the uploaded data.
  - Offer troubleshooting support if needed.

2. **Resources**:
   - Welcome email template with account setup instructions.
   - User manual for uploading usage data.
   - Step-by-step guide for generating invoices.
   - Video tutorials for onboarding.

Exercise 3 - Customer Feedback Analysis

1. **Action Items**:
   - **UI Redesign**: Prioritize a UI overhaul to make the platform more user-friendly. This can include simplifying the layout, improving the navigation, and modernizing the design.
   - **Improve Support Response Time**: Ensure faster response times by increasing staffing during peak hours, introducing automated responses for common issues, and tracking response time metrics.

2. **Email Draft**:
   "Dear [Client's Name],
   Thank you for your valuable feedback. We're thrilled to hear that you're satisfied with Zenskar's functionality.
   Regarding your concerns:
   - We're currently working on a UI update to improve the user experience.
   - We've also taken steps to enhance our support team's responsiveness to ensure timely assistance.
     We appreciate your patience as we continue to improve our services. If you have any other questions or suggestions, feel free to reach out.
     Best regards,
     [Your Name]

Exercise 4 - Understanding the Platform

Zenskar API Integration Documentation

1. Environment Setup

To set up the environment for the project, follow these steps:

1.1 Install Python

Ensure you have Python 3.x installed on your machine. You can download the latest version of Python from Python's official website.

1.2 Install Required Libraries

Make sure you have `requests` and `python-dotenv` installed. These libraries are required for making HTTP requests and loading environment variables.

Run the following command in your terminal to install them:

```
bash
Copy code
py -m pip install requests python-dotenv
```

1.3 Set Up Environment Variables

Create a `.env` file in your project's root directory and add the following:

```
env
Copy code
BASE_URL=https://api.zenskar.com
API_KEY=your_api_key_here
ORG_ID=your_org_id_here
```

Make sure to replace `your_api_key_here` and `your_org_id_here` with your actual API key and organization ID.

---

2. API Endpoints Used

2.1 Create Customer

- **Endpoint:** `POST /customers`
- **Description:** Creates a customer in the system with relevant details.
- **Request Data:**
  - `external_id`: Unique external identifier for the customer (e.g., `"236862834426"`).
  - `customer_name`: Full name of the customer (e.g., `"New Customer5"`).
  - `email`: Customer's email address (e.g., `"ayush25@gmail.com"`).
  - `phone_number`: Customer's phone number (e.g., `"+919811333910"`).

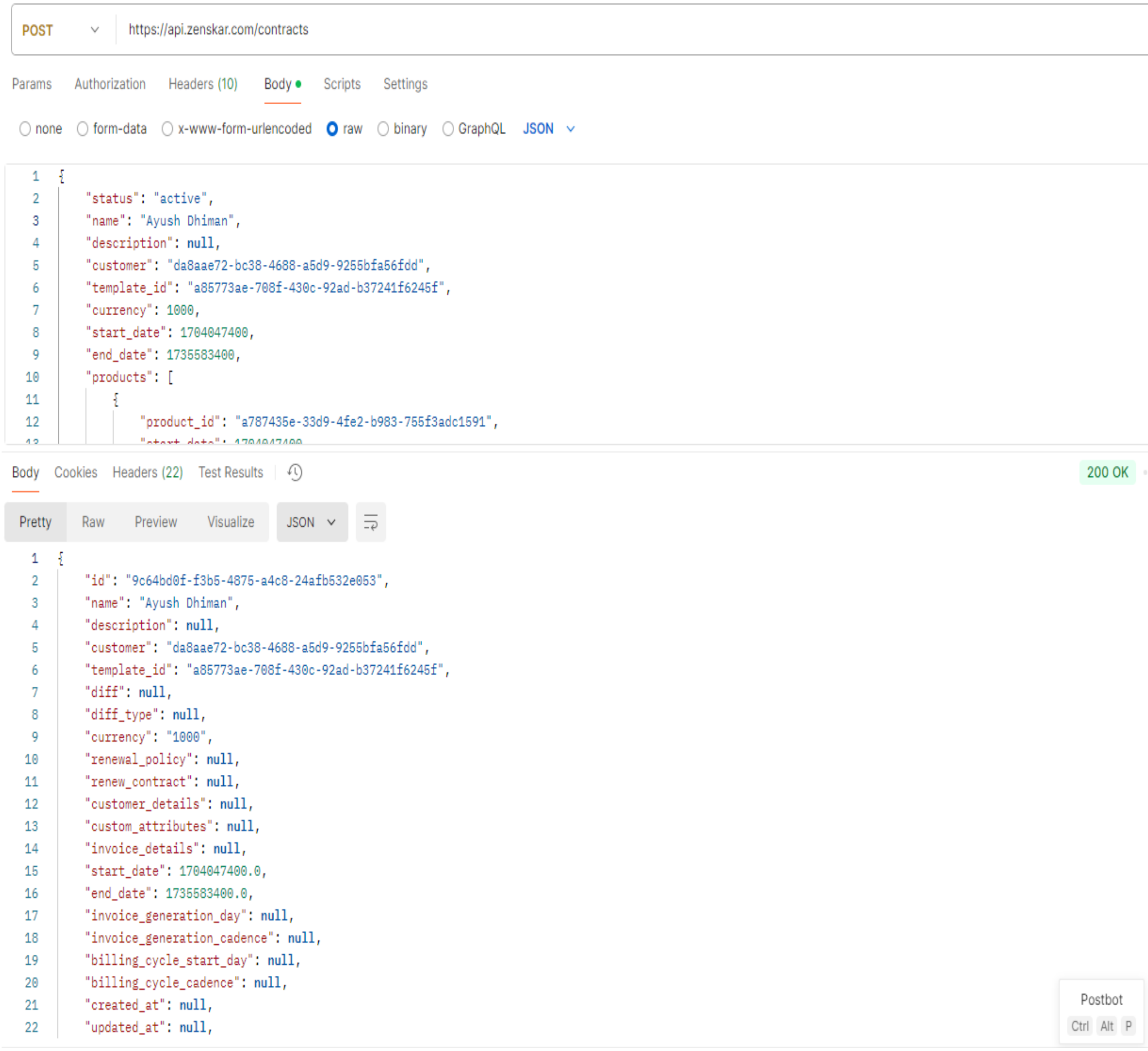
2.2 Create Products

- **Endpoint**: `POST /products`
- **Description**: This endpoint creates products like One Time Fee, Monthly Platform Fee, and Monthly User Fee.
- **Request Data**:
  - `name`: Name of the product.
  - `type`: Type of the product (e.g., "group", "product").
  - `billing`: Billing type (e.g., "prepaid", "postpaid").
  - `frequency`: Payment frequency (e.g., "one_time", "monthly").
  - `price`: The price of the product.

2.3 Create Contract

- **Endpoint**: `POST /contracts`
- **Description**: This endpoint creates a contract for the customer, associating them with products and a template.
- **Request Data**:
  - `status`: Status of the contract (e.g., "active").
  - `name`: Name of the contract.
  - `customer`: Customer ID.
  - `template_id`: Template ID.
  - `currency`: Currency associated with the contract.
  - `start_date`: Start date in Unix timestamp.
  - `end_date`: End date in Unix timestamp.

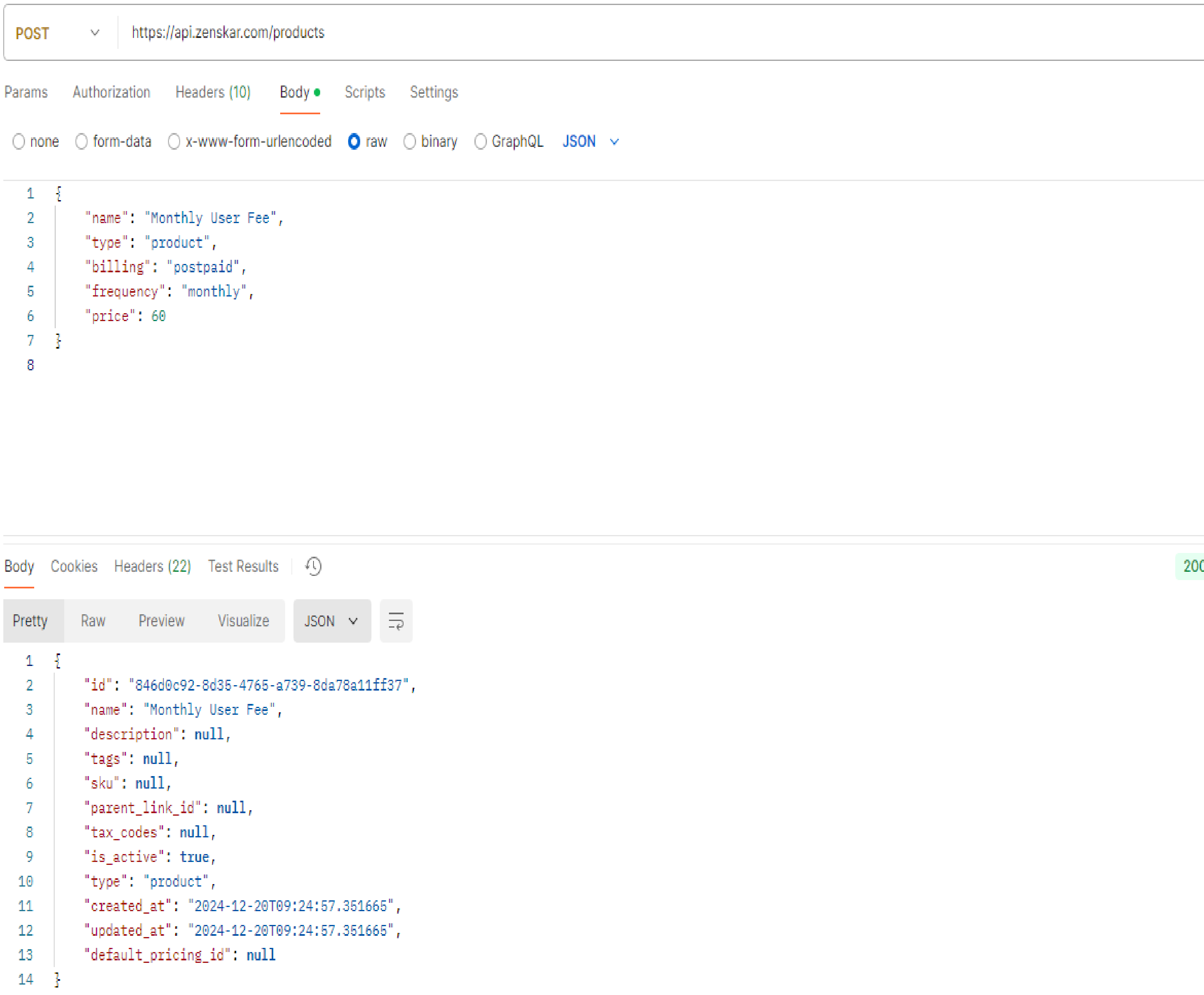
- `products`: List of product IDs associated with the contract.



POST https://api.zenskar.com/contracts

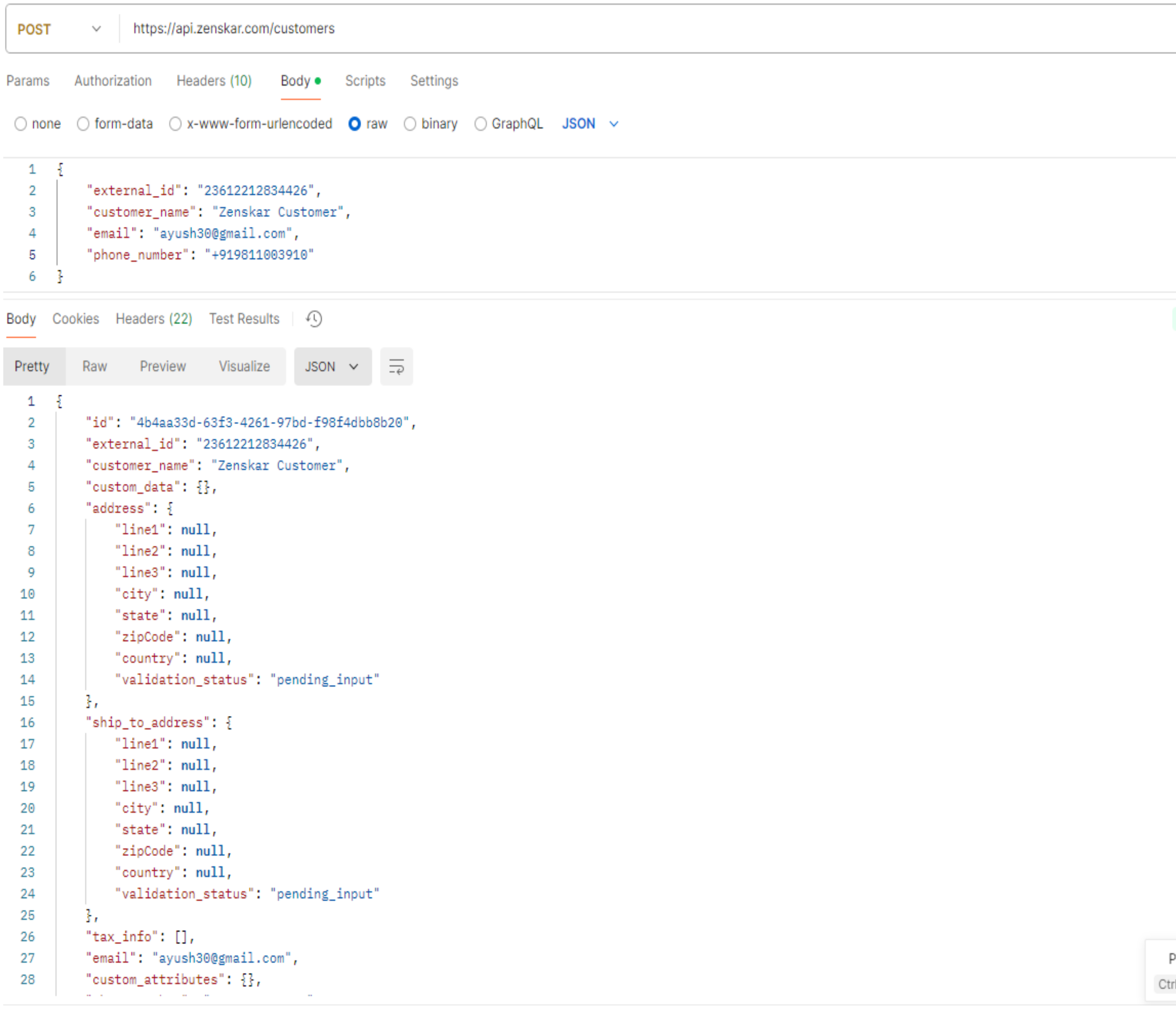
```
{
    "status": "active",
    "name": "Ayush Dhiman",
    "description": null,
    "customer": "da8aae72-bc38-4688-a5d9-9255bfa56fdd",
    "template_id": "a85773ae-708f-430c-92ad-b37241f6245f",
    "currency": 1000,
    "start_date": 1704047400,
    "end_date": 1735583400,
    "products": [
        {
            "product_id": "a787435e-33d9-4fe2-b983-755f3adc1591",
            "start_date": 1704047400
```

```
{
    "id": "9c64bd0f-f3b5-4875-a4c8-24afb532e053",
    "name": "Ayush Dhiman",
    "description": null,
    "customer": "da8aae72-bc38-4688-a5d9-9255bfa56fdd",
    "template_id": "a85773ae-708f-430c-92ad-b37241f6245f",
    "diff": null,
    "diff_type": null,
    "currency": "1000",
    "renewal_policy": null,
    "renew_contract": null,
    "customer_details": null,
    "custom_attributes": null,
    "invoice_details": null,
    "start_date": 1704047400.0,
    "end_date": 1735583400.0,
    "invoice_generation_day": null,
    "invoice_generation_cadence": null,
    "billing_cycle_start_day": null,
    "billing_cycle_cadence": null,
    "created_at": null,
    "updated_at": null,
```



```
{
    "name": "Monthly User Fee",
    "type": "product",
    "billing": "postpaid",
    "frequency": "monthly",
    "price": 60
}
```

```
{
    "id": "846d0c92-8d35-4765-a739-8da78a11ff37",
    "name": "Monthly User Fee",
    "description": null,
    "tags": null,
    "sku": null,
    "parent_link_id": null,
    "tax_codes": null,
    "is_active": true,
    "type": "product",
    "created_at": "2024-12-20T09:24:57.351665",
    "updated_at": "2024-12-20T09:24:57.351665",
    "default_pricing_id": null
}
```



https://api.zenskar.com/customers

Body: ( ) none ( ) form-data ( ) x-www-form-urlencoded (●) raw ( ) binary ( ) GraphQL JSON

```
{
    "external_id": "23612212834426",
    "customer_name": "Zenskar Customer",
    "email": "ayush30@gmail.com",
    "phone_number": "+919811003910"
}
```

```
{
    "id": "4b4aa33d-63f3-4261-97bd-f98f4dbb8b20",
    "external_id": "23612212834426",
    "customer_name": "Zenskar Customer",
    "custom_data": {},
    "address": {
        "line1": null,
        "line2": null,
        "line3": null,
        "city": null,
        "state": null,
        "zipCode": null,
        "country": null,
        "validation_status": "pending_input"
    },
    "ship_to_address": {
        "line1": null,
        "line2": null,
        "line3": null,
        "city": null,
        "state": null,
        "zipCode": null,
        "country": null,
        "validation_status": "pending_input"
    },
    "tax_info": [],
    "email": "ayush30@gmail.com",
    "custom_attributes": {},
```

3. Python Script Explanation

3.1 Creating Customer

The `create_customer()` function sends a POST request to the `/customers` endpoint to create a customer with details like external ID, name, email, and phone number.

3.2 Creating Products

The `create_product()` function sends a `POST` request to create a product. The data is formatted as JSON and sent to the `/products` endpoint. Each product includes details such as the name, type, billing method, frequency, and price.

3.3 Creating Contract

The `create_contract()` function is responsible for creating a contract. It sends a `POST` request to the `/contracts` endpoint with the contract details, including the customer ID, template ID, and associated products. The product start and end dates are converted into Unix timestamps.

3.3 Helper Functions

- `convert_to_unix_timestamp()`: Converts an ISO 8601 date string into a Unix timestamp.
- `create_all_products()`: Calls the `create_product()` function for all products (One Time Fee, Monthly Platform Fee, and Monthly User Fee).

3.4 Run the Script

To test individual functionalities, save each function in separate Python files and run them as needed:

- **Create Customer**
  Save the create_customer function in `create_customer.py` and run:

  py create_customer.py

- **Create Products**
  Save the `create_ product` function in `create_product.py` and run:

  py create_product.py

Create Contract
Save the `create_contract` function in `create_contract.py` and run:

```
py create_contract.py
```

Or else Run: py main.py

5. Challenges Faced and Resolutions

5.1 Issue: Product Duplication

- **Problem**: The product creation API would fail when attempting to create a product that already existed.
- **Solution**: Ensured proper checks are in place to verify product IDs before creation, and implemented error handling to catch duplicate creation attempts.

5.2 Issue: Date Conversion

- **Problem**: Converting ISO 8601 date format to Unix timestamp caused issues with leap years and different time zones.
- **Solution**: Used Python's `datetime.fromisoformat()` method to reliably convert date strings into Unix timestamps, considering time zone differences.

---

6. Conclusion

This integration with the Zenskar API involves creating products and contracts through their respective endpoints. Proper error handling and logging have been implemented to monitor the status of requests. The use of Python's `requests` library allows smooth interaction with the API, and environment variables ensure sensitive data (like the API key) are securely managed.