Assignment 9: Sprint 3 Plan (for Milestone #3)

1) Microservice B: getUpcomingPayments()

The getUpcomingPayments() API call is designed to retrieve a list of payments that are scheduled for future dates beyond the current day. This service is particularly useful for users who want to keep track of their upcoming financial obligations and ensure they are prepared for any pending payments. When a GET request is made to the /paymentManagement/upcomingPayments endpoint, the server responds with a JSON array containing details of each upcoming payment, including the vendor name, amount due, due date, and payment status. This allows users to have a clear and organized view of all their upcoming payments, helping them manage their finances more effectively and avoid missing any due dates.

This service retrieves the list of upcoming payments that are due after today's date.

Request

Method: GET

URL: `/paymentManagement/upcomingPayments`

Sample Response

```
Upcoming Payments: [
  {
    id: 'uuid-2',
    vendor: 'Electricity',
    amount: 150,
    dueDate: '2024-06-05',
    status: 'Pending'
  },
  {
    id: 'uuid-3',
    vendor: 'Internet',
    amount: 60,
    dueDate: '2024-06-10',
    status: 'Pending'
  },
  {
    id: 'uuid-6',
    vendor: 'Car Loan',
    amount: 400,
    dueDate: '2024-06-25',
    status: 'Pending'
  },
  {
    id: 'uuid-8',
    vendor: 'Insurance',
    amount: 100,
    dueDate: '2024-07-01',
    status: 'Pending'
  },
  {
    id: 'uuid-9',
    vendor: 'Netflix',
    amount: 15,
    dueDate: '2024-07-05',
    status: 'Pending'
  },
  {
```

2) Microservice B: getCurrentPayments()

The getCurrentPayments() API call is designed to retrieve a list of payments that are due on the current date. This service is particularly useful for users who need to stay on top of their immediate financial obligations and ensure that they do not miss any payments due today. When a GET request is made to the /paymentManagement/currentPayments endpoint, the server responds with a JSON array that includes detailed information about each payment due today, such as the vendor name, payment amount, due date, and payment status. This real-time overview allows users to quickly identify and address their current financial responsibilities, promoting timely payments and helping to avoid late fees or service interruptions.

This service retrieves the list of payments that are due today.

Request

Method: GET

URL: `/paymentManagement/currentPayments`

Sample Response

```
Current Payments: [
  {
    id: 'uuid-1',
    vendor: 'Rent',
    amount: 1200,
    dueDate: '2024-06-01',
    status: 'Pending'
  },
  {
    id: 'uuid-4',
    vendor: 'Water',
    amount: 45,
    dueDate: '2024-05-15',
    status: 'Pending'
  },
  {
    id: 'uuid-5',
    vendor: 'Gym',
    amount: 30,
    dueDate: '2024-05-20',
    status: 'Pending'
  },
  {
    id: 'uuid-7',
    vendor: 'Credit Card',
    amount: 200,
    dueDate: '2024-05-30',
    status: 'Pending'
  }
]
```

3) Microservice C: getPaymentHistory()

The getPaymentHistory() API call provides users with a comprehensive record of their past payments that have been completed and marked as paid. By making a GET request to the /paymentManagement/paymentHistory endpoint, users receive a detailed JSON array of historical payment transactions. Each entry in the response includes critical information such as the vendor name, payment amount, due date, actual payment date, and the payment status. This service is essential for users who need to review their payment history for budgeting, financial tracking, or reconciliation purposes. By offering a clear and accessible log of past payments, the getPaymentHistory() API call helps users maintain financial transparency and accountability.

This service retrieves the list of past payments that have been marked as paid.

Request

Method: GET

URL: `/paymentManagement/paymentHistory`

Sample Response

4) Microservice C: searchPaymentsByVendor()

The searchPaymentsByVendor() API call allows users to efficiently search for payments associated with a specific vendor. By making a GET request to the /paymentManagement/searchPaymentsByVendor endpoint with a vendor query parameter, users can retrieve a filtered list of payments related to the specified vendor. This functionality is particularly valuable for users who need to review all transactions involving a particular vendor, whether for budgeting, auditing, or tracking purposes. The server responds with a JSON array containing detailed information about each relevant payment, including the vendor name, amount, due date, and payment status. This targeted search capability enhances financial management by enabling users to quickly access and review payments specific to any vendor, ensuring better control and oversight of their expenses.

This service searches for payments by the vendor name.

Request

Method: GET

URL: `/paymentManagement/searchPaymentsByVendor?vendor=Spotify`

Query Parameter: `vendor` (required) - The name of the vendor to search for.

Sample Response


```
Search Payments: [
  {
    id: 'uuid-110',
    vendor: 'Spotify',
    amount: 10,
    dueDate: '2024-02-10',
    paymentDate: '2024-02-09',
    status: 'Paid'
  },
  {
    id: 'uuid-111',
    vendor: 'Spotify',
    amount: 11,
    dueDate: '2024-03-10',
    paymentDate: '2024-03-09',
    status: 'Paid'
  },
  {
    id: 'uuid-112',
    vendor: 'Spotify',
    amount: 12,
    dueDate: '2024-04-10',
    paymentDate: '2024-04-09',
    status: 'Paid'
  }
]
```

5) Microservice D: registerUser()

The registerUser() API call facilitates the registration of new users by securely capturing their credentials. When a POST request is made to the /paymentManagement/register endpoint with a JSON body containing a username and password, the server processes this information to create a new user account. The password is hashed using bcrypt before being stored, ensuring that user data is protected against unauthorized access. Upon successful registration, the server responds with a confirmation message indicating that the user has been registered successfully. This service is a critical entry point for users to access the system, enabling them to create an account and start managing their payments through the platform's secure and user-friendly interface.

This service registers a new user by accepting a username and password.

Request

Method: POST

URL: `/paymentManagement/register`

Body: (application/json)

```
{
                "username": "testuser",
                "password": "testpassword"
                }
```

Sample Response

```
User registered successfully: { message: 'User registered successfully' }
```

6) Microservice D: authenticateUser()

The authenticateUser() API call is essential for verifying user credentials and granting access to the system. When a POST request is made to the /paymentManagement/login endpoint with a JSON body containing the username and password, the server validates the provided credentials against the stored user data. If the authentication is successful, the server generates a JWT (JSON Web Token) and returns it in the response, along with a message indicating successful authentication. This token can then be used for subsequent requests to authenticate the user and authorize access to protected resources within the application. By securely validating user credentials and issuing a token, the authenticateUser() API call ensures that only authorized users can access and manage their financial information, maintaining the integrity and security of the system.

This service authenticates a user by accepting a username and password, and returns a JWT token if successful.

Request

Method: POST

URL: `/paymentManagement/login`

Body: (application/json)

```
{
                "username": "testuser",
                "password": "testpassword"
                }
```

Sample Response

```
Error authenticating user: { error: 'Invalid username or password' }
```