Expense Application

Prerequisites

- .NET SDK 8 installed on your machine.
- Docker/Docker Compose installed on your machine.
- Git installed on your machine.

Clone the Repository

```
git clone https://github.com/yourusername/ExpenseApplication.git
cd ExpenseApplication
```

Docker Configuration

1. Restart Docker Containers

Execute the following command to bring down existing Docker containers, volumes, and remove orphaned containers, then bring the Docker environment back up:

```
docker-compose down -v --remove-orphans && docker-compose up
```

2. Apply Migrations

Run the following commands to apply Entity Framework migrations and update the database:

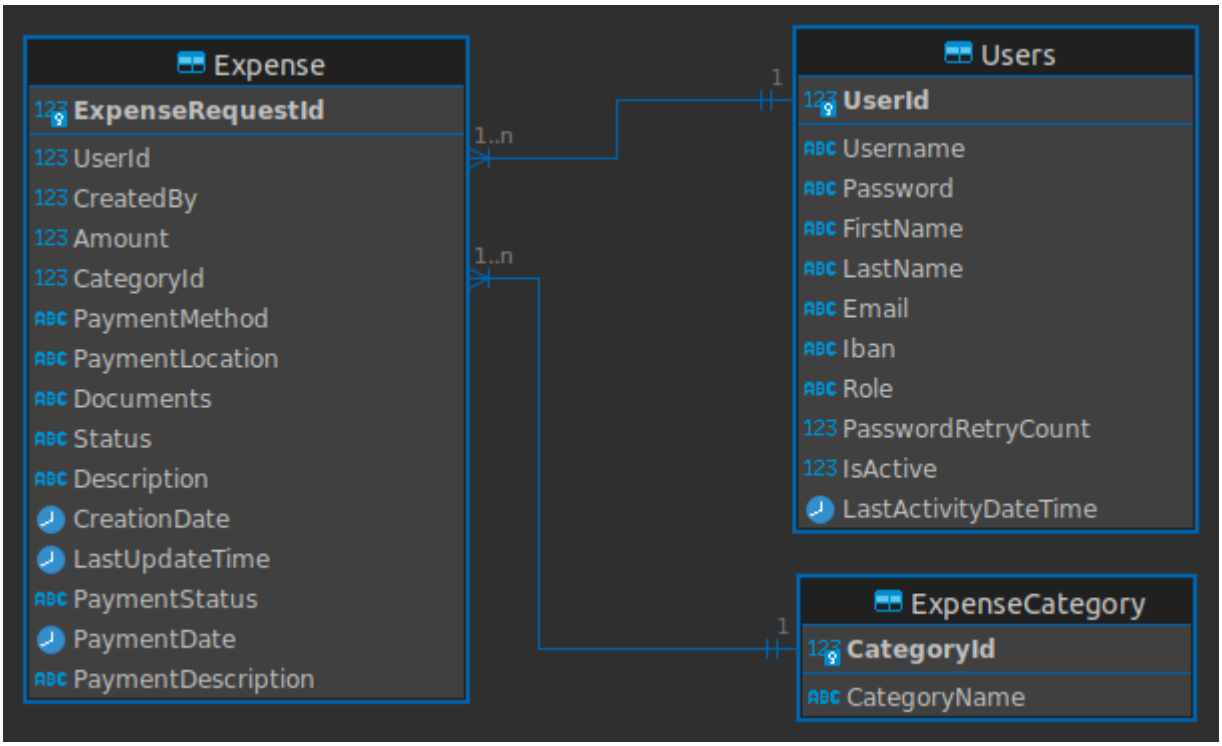
```
cd ExpenseApplication
dotnet ef migrations add InitialCreate --project Infrastructure --startup-
project Api
dotnet ef database update --project Infrastructure --startup-project Api
```

If you already have migrations, you can run the following command to drop the database and re-create it:

```
cd ExpenseApplication
dotnet ef database update --project Infrastructure --startup-project Api
```

The application will be accessible at `http://localhost:5245` by default.

Entity Relationship Diagram (ERD)



CreatedBy: Admin UserId
Status: Only admin can approve or reject expense requests.
Description: User add description for expense request.
PaymentDescription: Admin add description for expense request. Or error code from background job.

Expense

Create Expense

Method	Path
POST	/api/Expense/

Description:

Create an expense record. Authorized users: **[Admin, Personnel]**

Get All Expenses

Method	Path
GET	/api/Expense/

Description:

Retrieve expense records. Authorized users: **[Admin]**

Update Expense

Method	Path

Method	Path
PUT	/api/Expense/{expenseRequestId}

Description:

Update an expense record by expense request ID. Authorized users: **[Admin]**.
Updating expense as approved do not fire background job. Only approve endpoint fires background job. This endpoint can be used if expens is paid manually.

Delete Expense

Method	Path
DELETE	/api/Expense/{expenseRequestId}

Description:

Delete an expense record by expense request ID. Authorized users: **[Admin]**

Get Expense by ID

Method	Path
GET	/api/Expense/{expenseRequestId}

Description:

Retrieve an expense record by expense request ID. Authorized users: **[Admin]**

Approve Expense

Method	Path
PATCH	/api/Expense/Approve/{expenseRequestId}

Description:

Approve an expense record by expense request ID. Authorized users: **[Admin]** Admin users approve expense records by giving expense request id. This endpoint fires background job to update expense record.

Reject Expense

Method	Path
PATCH	/api/Expense/Reject/{expenseRequestId}

Description:

Reject an expense record by expense request ID. Authorized users: **[Admin]** Reject endpoint do not fire background job. Updates expense record immediately. And relevant fields are updated. Payment description, status, last updated date...

Get Expenses by User

Method	Path
GET	/api/Expense/ByUser

Description:

Retrieve expense records for the authenticated user. Authorized users: **[Admin, Personnel]**.
Admin users can give null user id to retrieve all expenses. Personnel can only retrieve their own expenses. Personnel can filter expenses by status and date range. Also Admin can filter expenses. This api is queryable.

ExpenseCategory

Create Expense Category

Method	Path
POST	/api/ExpenseCategory

Description:

Create a new expense category. Authorized users: **[Admin]**

Get All Expense Categories

Method	Path
GET	/api/ExpenseCategory

Description:

Retrieve all expense categories. Authorized users: **[Admin]**

Update Expense Category

Method	Path
PUT	/api/ExpenseCategory/{expenseCategoryId}

Description:

Update an expense category by expense category ID. Authorized users: **[Admin]**

Delete Expense Category

Method	Path
DELETE	/api/ExpenseCategory/{expenseCategoryId}

Description:

Delete an expense category by expense category ID. Authorized users: **[Admin]** Category can not be deleted if category has pending expenses.

Get All Expense Categories

Method	Path
GET	/api/ExpenseCategory/GetAllExpenseCategory

Description:

Retrieve all expense categories. Authorized users: **[Admin]**

PaymentSimulator

Process Payment

Method	Path
POST	/api/PaymentSimulator/ProcessPayment

Description:

Simulate the process of making a payment. Authorized users: **[Anonymous]** Hangfire is used to simulate payment process. Hangfire is a background job library. It is used to simulate payment process.

Report

Approved Payment Frequency Report

Method	Path
POST	/api/Report/ApprovedPaymentFrequencyReport

Description:

Generate a report on the frequency of approved payments. Authorized users: **[Admin]**.
Example response:

```
{
  "type": "monthly",
  "startDate": "01/01/2024 00:00:00",
  "endDate": "31/01/2024 23:59:59",
  "approvedCount": 16,
  "approvedSum": 14400,
  "averageApprovedAmount": 900
}
```


Rejected Payment Frequency Report

Method	Path
POST	/api/Report/RejectedPaymentFrequencyReport

Description:

Generate a report on the frequency of rejected payments. Authorized users: **[Admin]**.
Example response:


```
{
  "type": "weekly",
  "startDate": "22/01/2024 00:00:00",
  "endDate": "28/01/2024 23:59:59",
  "rejectedCount": 0,
  "rejectedSum": 0,
  "averageRejectedAmount": 0
}
```


Personnel Expense Frequency Report

Method	Path
POST	/api/Report/PersonnelExpenseFrequencyReport

Description:

Generate a report on the frequency of personnel expenses. Authorized users: **[Admin]**.
Example response:


```
{
  "type": "monthly",
  "startDate": "01/01/2024 00:00:00",
  "endDate": "31/01/2024 23:59:59",
  "totalPendingCount": 16,
  "totalPendingSum": 6040,
```

```
  "averagePendingAmount": 377.5,
  "personnelExpenseFrequencies": [
    {
      "userId": 1,
      "fullName": "Admin 1",
      "pendingCount": 4,
      "pendingSum": 1400,
      "averagePendingAmount": 350
    },
    {
      "userId": 2,
      "fullName": "Admin 2"
    }...
```


Personnel Summary Report

Method	Path
POST	/api/Report/PersonnelSummaryReport

Description:

Generate a summary report on personnel expenses. Authorized users: **[Admin, Personnel]**.
Personnel can only retrieve their own expenses.
Admin can retrieve all expenses if userId is null.
Example response:


```
{
  "userId": 3,
  "totalCount": 10,
  "approvedCount": 4,
  "rejectedCount": 2,
  "pendingCount": 4,
  "approvedPercentage": "40%",
  "approvedSum": 6500,
  "rejectedSum": 6500,
  "pendingSum": 6500,
  "expenses": [
    {
      "expenseRequestId": 3,
      "userId": 0,
      "categoryId": 3,
      "expenseStatus": "Approved",
      "paymentStatus": "Completed",
      "paymentDescription": "Payment Not Made",
      "amount": 2500,
      "paymentMethod": "Cash",
      "paymentLocation": "Office",
      "documents": "PayrollDocs",
      "description": "Personel maaşları ödendi.",
```

```
    "creationDate": "09/01/2024 00:00:00",
    "lastUpdateTime": "09/01/2024 00:00:00"
  }...
  {
```

Token

Generate Token

Method	Path
POST	/api/Token

Description:

Generate a new authentication token. Authorized users: **[Anonymous]**.
JWT is used for authentication.
Roles: Admin, Personnel\

User

Create User

Method	Path
POST	/api/User

Description:

Create a new user. Authorized users: **[Admin]**.

Get All Users

Method	Path
GET	/api/User

Description:

Retrieve all user records. Authorized users: **[Admin]**.

Update User

Method	Path
PUT	/api/User/{UserId}

Description:

Update a user by user ID. Authorized users: **[Admin]**.

Delete User

Method	Path
DELETE	/api/User/{UserId}

Description:

Delete a user by user ID. Authorized users: **[Admin]**.
User can not be deleted if user has pending expenses.

Activate User

Method	Path
PATCH	/api/User/ActivateUser

Description:

Activate a user account. Authorized users: **[Admin]**.
If a user try to login with deactivated account, user will get error message. Users are blocked if they have 3 failed login attempts.
Only admin can activate user.

Deactivate User

Method	Path
PATCH	/api/User/DeactivateUser

Description:

Deactivate a user account. Authorized users: **[Admin]**.
If a user try to login with deactivated account, user will get error message.
Blocked users can not login. So they can not be created expenses.\

Get All Users

Method	Path
GET	/api/User/GetAllUser

Description:

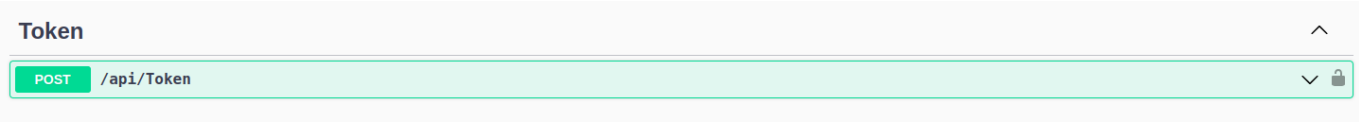
Retrieve all user records. Authorized users: **[Admin]**.

Additional Information

- Handlervalidator.cs and fluentvalidation is used for validation.
- Entities have unique index fields so additional validations are added for those like CategoryName, UserName, Email.
- Dependencies injected, code is decoupled. Avoid code duplication. Code can extended easily.
- 2 Admin, 2 Personnel users are created in migration. 42 expenses are created.
- Further Improvements: ApiResponse Class, I did not used apiresponse class so as to return appropriate status code and message. But it can be improved. Handling enums, ClassLevelCascadeMode etc.

Runtime Screenshots(Projenin Çalışma Örneği)

Get token



Get JWT Token and use it in Authorization header.

After Authorization, you can use api endpoints. Here a test token you can also use it.

```
eyJhbGciOiJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGRzaWctbW9yZSNobWFjLXNoYTI
1NiIsInR5cCI6IkpXVCJ9.eyJJZCI6IjEiLCJFbWFpbCI6ImFkbWluMUBleGFtcGxlLmNvbSIsI
lVzZXJuYW1lIjoiYWRtaW4xIiwiaHR0cDovL3NjaGVtYXMubWljcm9zb2Z0LmNvbS93cy8yMDA4
LzA2L2lkZW50aXR5L2NsYWltcy9yb2xlIjoiQWRtaW4iLCJleHAiOjE2NzA1ODkzNDg5LCJpc3M
iOiJFeHBlbnNlQXBwbGljYXRpb24iLCJhdWQiOiJFeHBlbnNlQXBwbGljYXRpb24ifQ.iyL7pHs
WY8q-O_gRqCIMctZtrSfK1rZ8vFYASk02utQ
```

ExpenseRequestId 42 is pending. Admin can approve it.

```
{
  "expenseRequestId": 42,
  "userId": 1,
  "categoryId": 10,
```

```
    "expenseStatus": "Pending",
    "paymentStatus": "Pending",
    "paymentDescription": "Payment Not Made",
    "amount": 300,
    "paymentMethod": "Cash",
    "paymentLocation": "Office",
    "documents": "EducationReceipts",
    "description": "Eğitim masrafları ödendi.",
    "creationDate": "13/01/2024 00:00:00",
    "lastUpdateTime": "06/01/2024 00:00:00"
  }
```


Approve Expense \

Response body


```
{
  "expenseRequestId": 42,
  "userId": 1,
  "categoryId": 10,
  "expenseStatus": "Approved",
  "paymentStatus": "OnProcess",
  "paymentDescription": "Payment Not Made",
  "amount": 300,
  "paymentMethod": "Cash",
  "paymentLocation": "Office",
  "documents": "EducationReceipts",
  "description": "Eğitim masrafları ödendi.",
  "creationDate": "13/01/2024 00:00:00",
  "lastUpdateTime": "22/01/2024 06:21:42"
}
```


Response headers

```
content-type: application/json; charset=utf-8
date: Mon,22 Jan 2024 03:21:42 GMT
server: Kestrel
transfer-encoding: chunked
```

Payment simulator sleeps 6 seconds. And probabilisticly approves(90%) or rejects(10%) expense.\

Response body


```
{
  "expenseRequestId": 42,
  "userId": 1,
  "categoryId": 10,
  "expenseStatus": "Approved",
  "paymentStatus": "Completed",
  "paymentDescription": "Payment successful. From IBAN: TR760009901234567800100001, To IBAN: TR760009901234567800100001, Amount: 300",
  "amount": 300,
  "paymentMethod": "Cash",
  "paymentLocation": "Office",
  "documents": "EducationReceipts",
  "description": "Eğitim masrafları ödendi.",
  "creationDate": "13/01/2024 00:00:00",
  "lastUpdateTime": "22/01/2024 06:21:49"
}
```


Response headers

```
content-type: application/json; charset=utf-8
date: Mon,22 Jan 2024 03:22:30 GMT
server: Kestrel
transfer-encoding: chunked
```

Further information can be found in the link. Also in documentation folder there is a postman collection. You can import it to postman and test the api.