API Documentation and Deployment Guide

API Documentation

1. Overview

The Staff API allows for the management of staff members, including registration, retrieval, and updates. It is designed to be secure with API key-based authentication.

2. Authentication

The API requires an API key to access its endpoints. The key should be passed in the `Authorization` header as follows:
Authorization: Bearer <your_api_key>

3. Endpoints

Base URL: `/api/staff`

1. Staff Registration

Endpoint: `POST /register`

Request Body:

```
{
  "Surname": "string",
  "OtherNames": "string",
  "DateOfBirth": "YYYY-MM-DD",
  "IDPhoto": "base64string",
  "UniqueCode": "string"
}
```

Responses:

- **200 OK**
- {
- "StatusCode": 200,
- "Message": "Registration successful",
- "EmployeeNumber": "string"
  }

- **400 Bad Request**
- {
- "StatusCode": 400,
- "Message": "Invalid data."

```
}
```

- **401 Unauthorized**
- {
- "StatusCode": 401,
- "Message": "Invalid unique code./Missing Authorization Header/Invalid API Server Key"
}

- **500 Internal Server Error**
- {
- "StatusCode": 500,
- "Message": "An error occurred while registering staff.",
- "Error": "string"
}

2. Staff Retrieval

Endpoint: `GET /`

Query Parameters: `employeeNumber` (optional)

Responses:

- **200 OK**
- {
- "StatusCode": 200,
- "Data": [/* Array of staff objects */]
}

- **404 Not Found**
- {
- "StatusCode": 404,
- "Message": "Staff not found."
}

3. Staff Update

Endpoint: `PUT /update`

Request Body:

```
{
  "EmpNo": "string",
  "DateOfBirth": "YYYY-MM-DD",
```

```
  "IDPhoto": "base64string"
}
```

Responses:

- **200 OK**
- {
- "StatusCode": 200,
- "Message": "Staff details updated successfully."
  }

- **400 Bad Request**
- {
- "StatusCode": 400,
- "Message": "Invalid data."
  }

- **404 Not Found**
- {
- "StatusCode": 404,
- "Message": "Staff not found."
  }

Deployment Guide

1. **Prerequisites:**
   - .NET SDK (version 6.0 or later)
   - A database (e.g., SQL Server) configured and accessible
   - Entity Framework Core installed in your project
2. **Setup:**

   Clone the repository or copy the project files to your local machine. Navigate to the project folder in your terminal.

3. **Database Configuration:**

   Update the `appsettings.json` file with your database connection string.

4. **Migrations:**

   Run the following command to create the database schema:

   ```
   dotnet ef migrations add InitialCreate
   dotnet ef database update
   ```

5. **Running the Application:**

Start the application using:

```
dotnet run
```

The API will be accessible at `http://localhost:5000/api/staff`.

6. **Testing the API:**

   Use tools like Postman or curl to test the endpoints as described in the API documentation.

7. **Logging:**

   Ensure that your logging mechanism is set up to capture API requests and responses as shown in the `LogApiRequest` method.

Deployment Guide for Client Application

1. Prerequisites:

- Flutter SDK installed on your machine.
- An IDE such as Visual Studio Code or Android Studio.
- Access to the .NET API with the bearer token configured in `appsettings.json`.

2. Create a Flutter Project:

1. Open your terminal and run:

   ```
   flutter create staff_app
   ```

2. Navigate into your project directory:

   ```
   cd staff_app
   ```

3. Add Dependencies:

Open `pubspec.yaml` and add the following dependencies:

```
dependencies:
http: ^0.14.0
flutter:
  sdk: flutter
```

Run the following command to install the new dependencies:

```
flutter pub get
```

4. Set API Service:

Open file `api_service.dart` in the `lib` directory and configure the url and serverKey.

Example implementation:

```
class ApiService {
final String baseUrl = 'http://localhost:5000/api; //
Change to your API URL
final String serverKey = 'your_server_key'; // Set your
server key here
```

5. Testing the Application:

Run the application on an emulator or physical device:

```
flutter run
```

Ensure your API is running and accessible from the device/emulator.

7. Handling Authentication:

Make sure the `serverKey` in your Flutter application matches the key set in the `appsettings.json` of your .NET API.

You can securely manage the server key by using environment variables or Flutter's secure storage package.

8. Deployment:

1. Build the application for release:
2.
```
flutter build apk  # For Android
flutter build ios  # For iOS
```

3. Follow the respective platform guidelines for deploying the app (e.g., Google Play Store for Android, App Store for iOS).

Notes:

Ensure that the API URL in your Flutter app points to the correct server where your .NET API is hosted.

Test thoroughly to ensure that the authentication mechanism works as expected.

Troubleshooting

- **API fails to start**: Check the logs for any configuration errors.
- **Database connection issues**: Ensure the connection string in `appsettings.json` is correct.
- **Authentication errors**: Verify that the API Server key is correctly set in the requests and matches the maintained in `appsettings.json`.