1.1 Project Structure

```
EnterpriseApp.OTP/
├── Configuration/
│   └── EmailSettings.cs
├── Constants/
│   └── OtpConstants.cs
├── Controllers/
│   └── OtpController.cs
├── Models/
│   ├── ApiModels.cs
│   └── OtpData.cs
├── Services/
│   ├── Interfaces/
│   │   ├── IEmailService.cs
│   │   └── IOtpService.cs
│   ├── EmailService.cs
│   └── OtpService.cs
├── Tests/
│   └── OtpServiceTests.cs
├── appsettings.json
├── EnterpriseApp.OTP.csproj
└── Program.cs
```

Key Features

1. **Layered Architecture**

   o Clear separation of concerns with controllers, services, and models

   o Dependency injection for loose coupling and better testability

   o Interface-based design for all services

2. **Email Implementation with MimeKit**

   o Robust email sending using MimeKit and MailKit

   o HTML-formatted emails for better user experience

o Configurable SMTP settings via appsettings.json

3. **Memory Cache Implementation**

   o Efficient in-memory caching for OTP storage

   o Automatic expiration after configured time (1 minute)

   o Thread-safe operations for concurrent requests

4. **Security Features**

   o Domain restriction (.dso.org.sg only)

   o Email validation

   o Limited attempts with tracking

   o OTP expiration

   o OTP removal after successful verification

5. **API Design**

   o RESTful API with proper HTTP status codes

   o Consistent response format

   o Input validation

   o Swagger documentation

6. **Logging**

   o Comprehensive logging throughout the application

   o Error tracking for failed operations

   o Security-related logging for audit purposes

7. **Unit Tests**

   o Test coverage for the OTP service

   o Mocking of dependencies

How to Use

1. **Setup:**

o Create a new ASP.NET Core Web API project

o Add the required NuGet packages (MailKit, MimeKit, etc.)

o Copy the files to their respective directories

2. **Configuration:**

o Update the SMTP settings in appsettings.json with your email provider details

o Adjust the allowed domain and OTP parameters as needed

3. **API Endpoints:**

o POST /api/otp/generate - Generate and send an OTP

o POST /api/otp/verify - Verify an OTP

This implementation follows enterprise-level design patterns and best practices, making it suitable for production use in a scalable application.