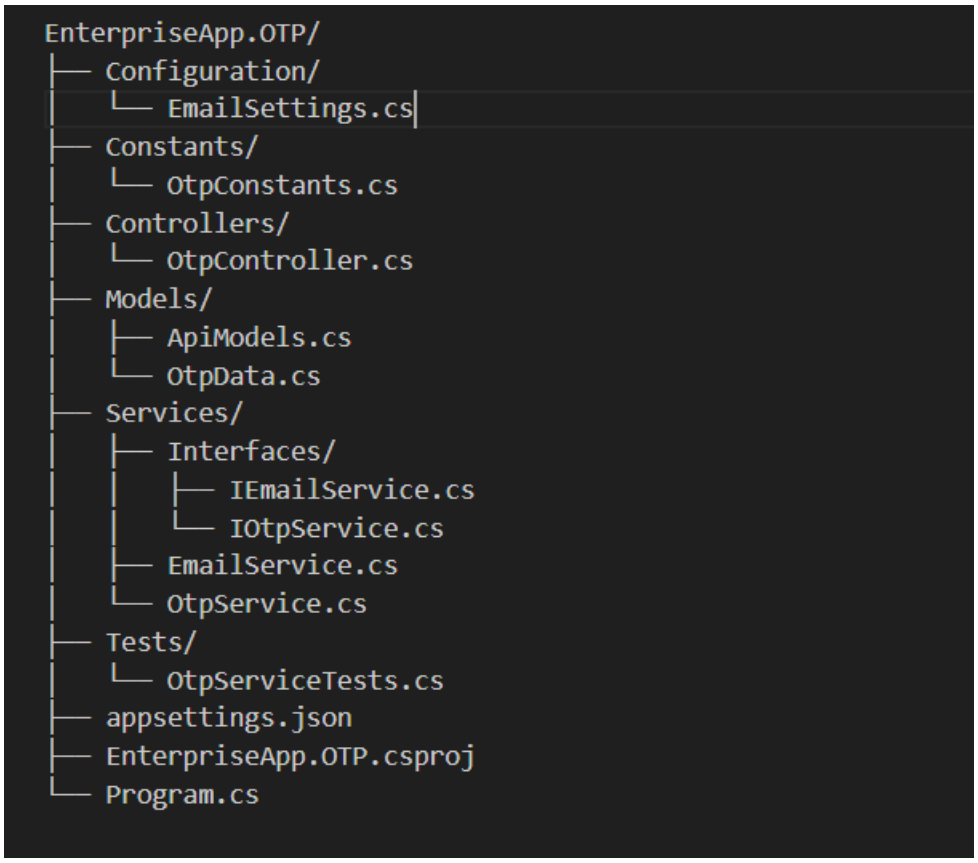
1.1 Project Structure



Key Features

1. **Layered Architecture**
    - Clear separation of concerns with controllers, services, and models
    - Dependency injection for loose coupling and better testability
    - Interface-based design for all services
2. **Email Implementation with MimeKit**
    - Robust email sending using MimeKit and MailKit
    - HTML-formatted emails for better user experience

- Configurable SMTP settings via appsettings.json

3. **Memory Cache Implementation**
   - Efficient in-memory caching for OTP storage
   - Automatic expiration after configured time (1 minute)
   - Thread-safe operations for concurrent requests
4. **Security Features**
   - Domain restriction (.dso.org.sg only)
   - Email validation
   - Limited attempts with tracking
   - OTP expiration
   - OTP removal after successful verification
5. **API Design**
   - RESTful API with proper HTTP status codes
   - Consistent response format
   - Input validation
   - Swagger documentation
6. **Logging**
   - Comprehensive logging throughout the application
   - Error tracking for failed operations
   - Security-related logging for audit purposes
7. **Unit Tests**
   - Test coverage for the OTP service
   - Mocking of dependencies

How to Use

1. **Setup:**

- Create a new ASP.NET Core Web API project
- Add the required NuGet packages (MailKit, MimeKit, etc.)
- Copy the files to their respective directories

2. **Configuration:**
   - Update the SMTP settings in appsettings.json with your email provider details
   - Adjust the allowed domain and OTP parameters as needed
3. **API Endpoints:**
   - POST /api/otp/generate - Generate and send an OTP
   - 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.