**Assumptions Made in the Code**

### Email Domain Validation

This application currently only accepts email addresses ending with @dso.org.sg or @gmail.com for OTP verification, as configured in the GenerateOtpEmail method. If necessary, other domains can easily be added to the list. Additionally, the IsValidEmail method specifically checks for the @dso.org.sg domain pattern, assuming that this is the main domain used by authorized users.

### Simulated Email Sending

The SendEmailAsync method is set up to simulate email delivery by using a Task.Delay of 500 milliseconds and printing a console log. This setup is intended for demonstration purposes. In a production setting, an actual email service, such as SMTP or a third-party provider like SendGrid, would replace this simulation.

### In-Memory OTP Storage

OTPs are temporarily stored in memory using a ConcurrentDictionary where each OTP is linked to an email address and an expiration timestamp. This setup assumes that OTPs don’t need to persist after server restarts and that memory storage can handle the application’s load. For scaled or distributed deployments, a shared cache or database would be more suitable.

### 1-Minute OTP Expiry

OTPs are valid for a short period of 1 minute, as specified by the \_otpExpiryDuration setting. This duration works well for standard OTP use cases but may need adjustments for scenarios requiring different time limits.

### 6-Digit OTP Generation

The OTP is a random 6-digit number generated using Random.Next(100000, 999999), which provides enough uniqueness and security for demonstration purposes. If higher security is needed, the OTP length can be extended.

### Single OTP Per User

Each email address can only have one active OTP at any given time. When a new OTP is generated, it automatically replaces any previously stored OTP for that user.

### Form-Based OTP Verification

OTP verification is assumed to take place via form submission on a single webpage (Index.cshtml). While this method works well for basic form-based verification, it can be modified to support AJAX or API-based requests for a more seamless user experience.

### Simple Status Messages

Service methods return status messages like STATUS\_EMAIL\_OK and STATUS\_OTP\_FAIL as simple strings. This approach keeps development straightforward, but for more complex applications, these status messages might need to be standardized or expanded.

**Testing the Module**

To ensure the EmailOTPModule functions correctly, testing can be broken down into unit tests and integration tests. Here's a structured approach to testing:

**Testing EmailOtpService**

* **OTP Generation**: Test that GenerateOtp produces a six-digit OTP as expected.
* **Email Validation**: Test GenerateOtpEmail to accept emails ending in @dso.org.sg or @gmail.com and reject others, returning "STATUS\_EMAIL\_INVALID" for invalid domains.
* **OTP Expiry**: Confirm OTPs expire after one minute, returning "STATUS\_OTP\_TIMEOUT" if checked after this duration.
* **Successful OTP Verification**: Generate and immediately verify an OTP, ensuring CheckOtp returns "STATUS\_OTP\_OK" for a correct OTP.
* **Incorrect OTP Verification**: Generate an OTP and test with an incorrect value, expecting "STATUS\_OTP\_FAIL."
* **Simulated Email Sending**: Ensure SendEmailAsync simulates sending by returning true after a delay.

**Testing OtpController**

* **Send OTP Request**: Mock EmailOtpService and submit a valid email to SendOtp, verifying a successful response with "STATUS\_EMAIL\_OK."
* **Invalid Email Submission**: Test SendOtp with an invalid email, expecting "STATUS\_EMAIL\_INVALID."
* **Correct OTP Verification**: Mock CheckOtp to return "STATUS\_OTP\_OK" for the VerifyOtp action with a correct OTP, confirming success.
* **Incorrect OTP Verification**: Mock CheckOtp to return "STATUS\_OTP\_FAIL" when tested with an incorrect OTP.

**Integration Tests for Application**

* **Homepage Load**: Ensure the homepage (/) loads correctly, validating the default route and Index action.
* **Send OTP Route**: POST to /send-otp with a valid email, checking for a "STATUS\_EMAIL\_OK" response.
* **Verify OTP Route**: After OTP generation, POST to /verify-otp with the OTP, confirming a "STATUS\_OTP\_OK" response.