**Warning: do not test the One-Time Password feature using ETSU Guest Wifi. Check the README for more information.**

**Front-end Website:**

1. **LoginPage.html and LoginPage.js:**
   * Responsible for user interaction, specifically handling login functionalities.
   * On request for OTP, it communicates with the Back-end Microservice to generate and send OTP via API.
   * On OTP verification, it again communicates with the Back-end Microservice to validate the OTP and receive a JWT token.
2. **LandingPage.html and LandingPage.js:**
   * Displays the main landing page content.
   * Uses the JWT token to personalize the user experience by fetching user information from the token.

**Back-end Microservice (app.js):**

1. **OTP Generation and Email Sending:**
   * Exposes an API endpoint to generate OTP and send it to the user's email.
   * Utilizes Nodemailer for email communication (sending email from one account to another).
2. **OTP Verification:**
   * Exposes an API endpoint to verify OTP entered by the user.
3. **JWT Token Generation:**
   * Generates a JWT token upon successful OTP verification.
   * Sends the token back to the Front-end for storage and later use.

**Workflow:**

1. **User Login:**
   * User initiates the login process by requesting an OTP.
   * Front-end sends a request to the Back-end's **/generateOTP** endpoint.
   * Back-end generates and sends the OTP to the user's email.
2. **OTP Verification:**
   * User enters the received OTP and verifies it.
   * Front-end sends a request to the Back-end's **/verifyOTP** endpoint.
   * Upon successful verification, the Back-end generates a JWT token.
3. **Token Usage:**
   * The JWT token is stored in the Front-end for future use.
   * LandingPage.js retrieves and decodes the token to personalize the user's landing page. (It updates the header’s title to the user’s email address)

**Benefits:**

* **Scalability:** Each microservice can be independently deployed and scaled as needed.
* **Isolation:** Front-end and Back-end components operate independently, reducing dependencies.
* **Flexibility:** Easier to update and maintain individual services without affecting the entire system.