**Dev 1: builds user authentication system and error handling/input validation authentication and validation error handling**

**Registration (/register)**

When a new customer tries to create an account, the code does these steps:

1. **Collect Input**  
   It takes values entered in the form:
   * email
   * first\_name
   * last\_name
   * phone
   * password

The input is trimmed (spaces removed) and converted to lowercase (for email).

1. **Validate Email**
   * Uses a **regular expression (EMAIL\_RE)** to check format.
   * Example of valid: john.doe@example.com
   * Example of invalid: john.doe@com or john@@example
   * If invalid display error *“Please enter a valid email.”*
2. **Validate Phone Number (Optional)**
   * If a phone is provided, it must match the **US phone number regex (PHONE\_RE)**.
   * Examples of valid:
     + 123-456-7890
     + (123) 456-7890
     + +1 123-456-7890
   * If invalid display error *“Please enter a valid phone number (e.g., 123-456-7890).”*
3. **Validate Password Strength**  
   The password must meet all of these:
   * At least **8 characters long**
   * At least **1 number**
   * At least **1 uppercase letter**
   * At least **1 lowercase letter**
   * Example of valid: Moffat2024
   * Example of invalid: password (too weak)
   * If invalid display error *“Password must be 8+ chars and include upper, lower, and a number.”*
4. **Hash Password**
   * Uses **bcrypt** to hash the password before saving to the database.
   * This means the actual password is never stored in plain text (for security).
5. **Insert into Database**
   * Creates a new account number like CUST1004.
   * Saves the customer’s details (name, email, phone, password hash) in the Customers table.
   * If successful display success *“Registration successful. Please log in.”*
   * If an error occurs (e.g., duplicate email) → Rollback and show error.

**Login (/login)**

When someone tries to log in:

1. **Collect Input**
   * Takes email and password from the form.
2. **Check Customers Table**
   * Looks up the email in the Customers table.
   * If not found → Checks the Employees table.
   * Sets the role (customer or Staff).
3. **Handle Invalid Login**
   * If email is not found display error *“No account found with this email. Please register first.”*
4. **Validate Password**
   * Uses **bcrypt.checkpw()** to compare the entered password with the stored hash.
   * If wrong password display error *“Incorrect password, please try again.”*
   * If hash is corrupted (invalid salt) → Shows the same friendly error message (instead of crashing).
5. **Start Session**
   * If login succeeds → Saves user data in session (like user\_id, email, and role).
   * Flash success *“Welcome back Moffat Bay Lodge!”*

**Logout (/logout)**

* Clears all session data (logs user out).
* Flash message *“You have been logged out.”*
* Redirects to home page (index).

In summary:

* **Email validation** ensures users can only register with a properly formatted email.
* **Phone validation** ensures correct US-style phone numbers if provided.
* **Password validation** enforces security by requiring a strong password.
* **Password hashing (bcrypt)** ensures passwords are never stored as plain text.
* **Error handling** provides user-friendly messages instead of raw exceptions.