# CMPE 327 – Assignment 05

Department of Electrical and Computer Engineering
Queen's University

Composed By
Nicholas Seegobin (20246787)
Samhith Sripada (20232740)
Rodrigo Del Aguila (20275528)

Date of Submission
Tuesday, November 25th, 2024

*"We hereby attest that the work submitted is our own individual work and that no portion of this submission has been copied in whole or in part from another source, with the possible exception of properly referenced material."*

# Table of Contents



# Table of Figures

# 1. Integration Test Summary

## 1.1. Login and Registration Integration Tests

The integration test script, *test_integration.py*, verifies key features of the RentEase system, including user registration, user login, Google authentication, and password recovery. It simulates real user actions, such as creating accounts and logging in, to ensure the system works as expected. Each test checks how the frontend, backend, and database interact, identifying any issues in how these components work together.

### 1.1.1. Import Statements

The script begins by importing necessary libraries and modules:

- **pytest:** A testing framework for writing and executing test cases.
- **random:** Used to generate random usernames for test users.
- **User:** The user model class from the backend.
- **getUserData:** A function to retrieve user data from the database for validation.

These imports allow the script to simulate user interactions and verify database operations.

### 1.1.2. Test User Initialization

Two test user profiles are created:

- **testUser:** A generic test user with a random username.
- **googleTestUser:** A simulated user for testing Google authentication.

Both users are initialized with attributes such as username, email, password, and user_type.

### 1.1.3. test_register Function

- **Purpose:** Verifies that a new user can register successfully.
- **Functionality Tested:** Simulates user registration, ensures the backend saves user data, and checks if the data persists in the database.

### 1.1.4. test_invalid_register Function

- **Purpose:** Validates that invalid registration inputs are rejected.
- **Functionality Tested:** Tests input validation logic and ensures no invalid data is saved.

### 1.1.5. test_login Function

- **Purpose:** Confirms that existing users can log in with valid credentials.
- **Functionality Tested:** Verifies login functionality through the backend.

### 1.1.6. test_google_signup Function

- **Purpose:** Tests Google-based user signup.
- **Functionality Tested:** Simulates Google user registration and ensures proper database integration.

### 1.1.7. test_google_signin Function

- **Purpose:** Ensures returning users can sign in using Google credentials.
- **Functionality Tested:** Verifies backend Google login functionality.

### 1.1.8. test_forgot_password Function

- **Purpose:** Simulates the password recovery process.
- **Functionality Tested:** Tests if the backend handles password reset requests correctly.

### 1.2. Tenant Integration Tests

The integration test script, *tenants.test.js*, ensures the smooth functionality of the tenants table and the registration form. It validates user interactions and UI responsiveness by simulating various scenarios. These tests verify that the frontend behaves as intended when interacting with dropdown menus, modals, sorting mechanisms, and date parsing functionalities.

#### 1.2.1. Registration Form Test

The registration form functionality ensures that users can interact with the role selection form accurately. It validates the user input by displaying an error message if no role is selected, preventing form submission. Once a valid role, such as "Landlord," is selected, the error message is removed, and the form allows submission. This ensures that the form is intuitive and prevents invalid inputs from being processed.

#### 1.2.2. Tenant Table Test

The tenant table functionality focuses on interactive elements like dropdown menus, modals, table sorting, and date parsing. Users can toggle action dropdowns for individual tenants, with all open menus closing when clicking outside. The table includes modals for actions like viewing tenant profiles, sending messages, and checking message histories, each of which can be opened and closed as expected. Sending a message displays a success toast notification that disappears after a brief delay. The table also supports sorting by name and "Last Contacted" date columns in ascending and descending order, with dates accurately parsed into JavaScript Date objects for proper chronological sorting. These features ensure a smooth and user-friendly interface for managing tenant information.

## Step 02: Test Script Outputs

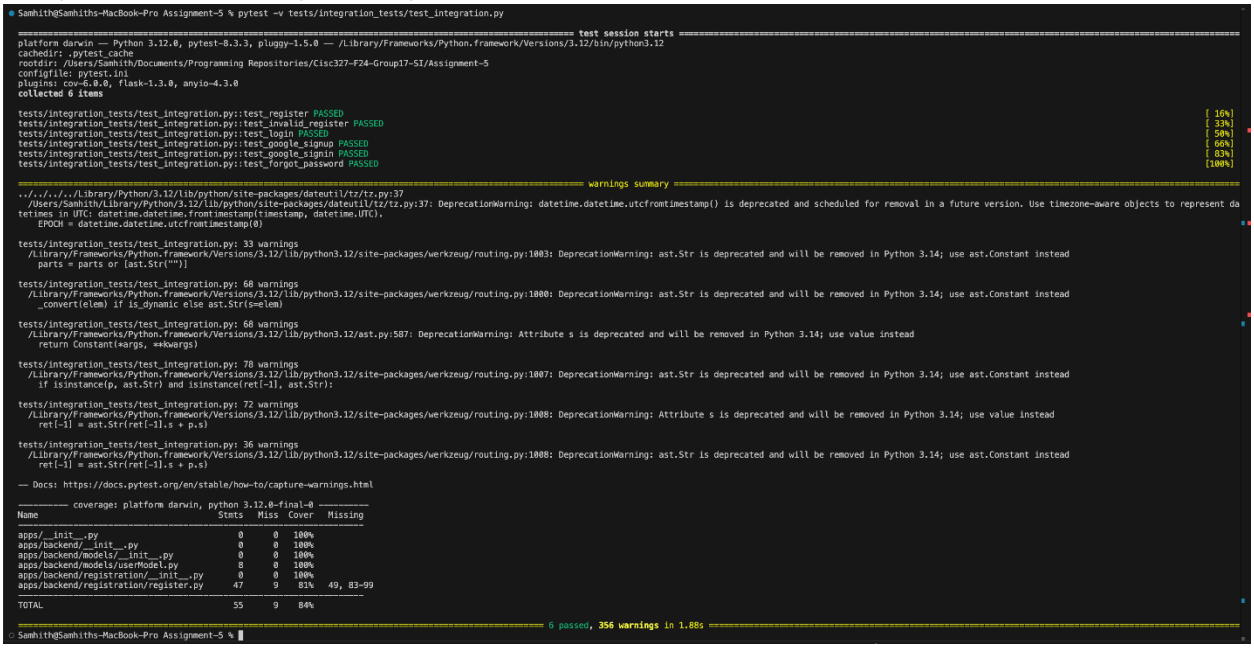


*Figure 1: A snippet of the result of running the test_integration.py script.*

```
collected 6 items

tests/integration_tests/test_integration.py::test_register PASSED
tests/integration_tests/test_integration.py::test_invalid_register PASSED
tests/integration_tests/test_integration.py::test_login PASSED
tests/integration_tests/test_integration.py::test_google_signup PASSED
tests/integration_tests/test_integration.py::test_google_signin PASSED
tests/integration_tests/test_integration.py::test_forgot_password PASSED
```

*Figure 2: A close-up of the passed test cases for the test_integration.py script.*

```
(base) roddbeatz@rods-macbook-pro tenant_tests % npm run coverage

> tenant-tests@1.0.0 coverage
> nyc --reporter=text-summary --reporter=html mocha



  Integration Tests
    Register Form
Validation function is running
      ✔ should show error when no role is selected
Validation function is running
      ✔ should hide error and allow submission when landlord is selected
    Tenants Table
      Dropdown Functionality
        ✔ should toggle dropdown menu when clicking the toggle button
        ✔ should close all dropdowns when clicking outside
      Modal Functionality
        ✔ should open and close view profile modal
        ✔ should handle send message modal and show success toast
        ✔ should open and close message history modal
      Table Sorting
        ✔ should sort table by name column
        ✔ should sort table by date column
      Date Parsing
        ✔ should correctly parse dates in the expected format


  10 passing (212ms)


=============================== Coverage summary ===============================
Statements   : 100% ( 8/8 )
Branches     : 100% ( 0/0 )
Functions    : 100% ( 0/0 )
Lines        : 100% ( 8/8 )
================================================================================
(base) roddbeatz@rods-macbook-pro tenant_tests %
```

*Figure 3: A snippet of the result of running the tenants.test.js script.*

# 3. Test Execution Instructions

## 3.1 Python Integration Tests

1. Navigate to the project root directory: **cd Assignment-5**
2. Run the integration test script using pytest: **pytest -v tests/integration_tests/test_integration.py**

### 3.2. Javascript Integration Tests

1. Navigate to the tenant_tests directory: **cd Assignment-5/tests/tenant_tests**
2. Install the required npm packages: **npm install**
3. Run the integration tests with coverage: **npm run coverage**

## 4. Task Distribution

| Name | Contributions |
|---|---|
| Samhith Sripada (20232740) | 1. Implemented integration tests for Python routes.<br>2. Updated README. |
| Nicholas Seegobin (20246787) | 1. Created a report to document the coverage test analysis.<br>2. Updated README. |
| Rodrigo Del Aguila Velarde (20275528) | 1. Implemented integration tests for Javascript features.<br>2. Updated README. |