Final Project Phase#3:

1. Project Identification:

Topic: End-to-End Testing for a Hotel Booking App with Selenium and TestNG

2. Groups:

Sr No	Name	Email
1	Sanika Hirave	sanikahirave@oakland.edu

3. Project specifications

The **primary goal** of this project is to:

1. Develop a Hotel Management System which will have booking and cancellation feature for multiple hotels and the history of all transactions with an interactive user interface.
2. Test the developed system by performing end-to-end testing using various testing techniques like Selenium and TestNG, pytest.

Aims:

1. Create the hotel management app.
2. Verify that the booking, and cancellation features work as expected.
3. Test the Flask API responses.
4. Check UI developed using react.
5. Write and automate test cases.

Intended Outcomes:

1. Ensure a Functional System: Verify that the booking, cancellation, and booking history features work correctly.
2. Validate API Responses: To check and confirm that the Flask API correctly processes user requests, returns accurate responses.
3. Improve UI Reliability: Ensure that the React frontend displays service information properly and also responds smoothly to user actions (clicks on buttons, clicks on navigation menu) with Selenium and TestNG.
4. Ensure Data is being saved: Verify that all bookings and cancellations are accurately stored in the PostgreSQL database as booking history using pytest**.**
5. Create and Automate Test cases: Develop and automate test cases using Selenium and TestNG**.**

Problem Addressed:

The problem addressed in this project is to ensure that the developed Hotel Management System functions correctly.

1. User Interface (UI): Ensuring that the React frontend displays hotel details properly, responds correctly to user interactions, and works smoothly on different devices.

2. API Request and Response Handling: Verifying that the Flask API correctly processes booking and cancellation requests, returns accurate responses.
3. Booking History Accuracy: Making sure that all bookings and cancellations are **p**roperly saved in the PostgreSQL database.

Key Features & Functionalities

1. Responsive UI(Buttons, Navigation bars)
2. Book and cancel rooms with real-time updates.
3. Maintain a transaction history of all bookings.
4. Flask API integration for hotel and booking management.
5. End-to-end testing to test all the functionalities
6. Automated Selenium, testNG tests for system validation.

Scope (Included):

1. UI and API testing for the booking and cancellation process.
2. Verifying real-time updates in PostgreSQL databases.
3. Ensuring UI responsiveness across devices.

Boundaries **(**Not Included):

1. Payment gateway integration.
2. Multi-user authentication scenarios.

4. Motivation and Problem Statement

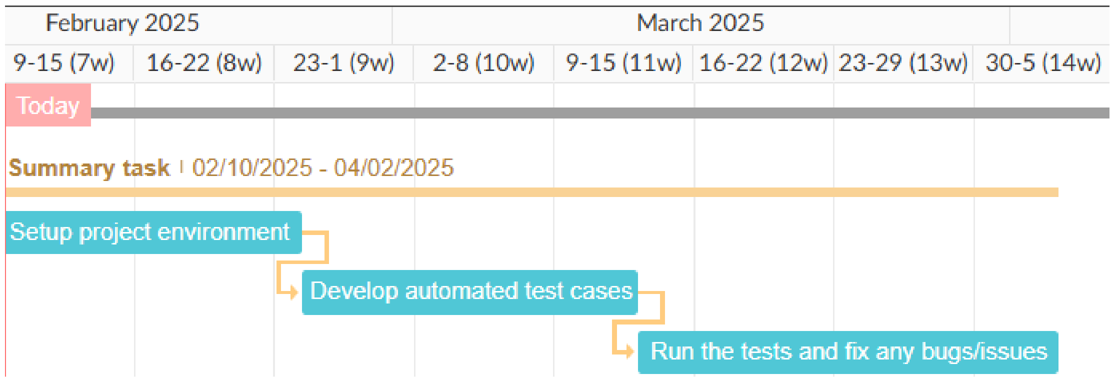
Problem Being Solved:

The project aims to solve the challenges related to the manual process of hotel booking. It is to ensure if all the features are working fine.

Importance of Solving This Problem

- An automated testing framework.
- Faster bug detection using Selenium and TestNG.

5. Tasks management



6. Approach:

The approach used in this project to test the full system is end-to-end testing and testing will be divided into 2 parts. UI is being tested using Selenium and TestNG whereas the server and API calls are being tested using Pytest as the language used for system development and testing is python.

1. **Design Specifications:**

The design of the Hotel management system includes UI, Server code and database.
Technologies used in this Architecture: React frontend, Flask backend, PostgreSQL database.
Components:

- Frontend/UI: Handles user interactions like navigation bar, buttons. Provides information on user clicks.
- Backend: Manages hotel booking and cancellation logic along with managing history if these transactions.
- Database: Stores hotel details and transaction history.

Testing design of this includes UI testing using selenium, Server testing using Pytest.

2. **Technical Design**
   1. Frontend: React.js with responsive UI.
   2. Backend: Flask-based API handling hotel data.
   3. Database: PostgreSQL for booking history.
   4. Selenium: For testing of UI.
   5. TestNG: To manage testing related activities.
   6. Webdriver and Maven: To run the selenium and TestNG effectively.

The UI testing design is as below:

- Firstly, Test cases are written with the help of LLM's and by myself for navigation bar and booking buttons.
- TestNG component is created to manage these test cases.
- Webdriver (chromedriver) and maven is set as an environment to run test.

- Tests are run and set automatically using terminal/command prompt and reports are collected.

The Server/API code testing design is as below:

- Test cases are written with the help of LLM's and some of them are developed by myself and the file is saved in same directory.
- Test cases are run using pyest using terminal.

3. User Interface Design:

The user interface design includes the homepage which has a navigation bar, which includes 3 clicks:

- Home: Refers to the home page which includes information about the hotel management system.
- Search Hotels: By clicking this, we can actually see the list of hotel details along with its availability and buttons to book or cancel the room.
- Booking History: This page is actually showing a table of all the transactions of hotel bookings, like hotel booked or hotel cancelled with its name and time of transaction.

2. The UI testing design is as below:

- Firstly, Test cases are written for navigation bar and booking buttons.
- TestNG component is created to manage these test cases.
- Webdriver (chromedriver) and maven is set as an environment to run test.
- Tests are run and reports are collected.

The testing approach used for web application is divided into 2 parts:

- UI testing:
  For UI testing, selenium and testNG are used with Maven environment. Test cases are written in Java and run through testNG using XML Language.
- Backend/Server/API testing:
- As the backed or the server code is written using Python, I have used pytest to test server calls.

7. Experimental setup:

Tool or framework and the programming language used:

For development purpose,

1. I have used react for UI development, CSS for styling of UI.
2. API calls are handled using flask and server code is written in python.
3. It is run using command prompt on chrome.

For testing purpose,

1. UI testing is performed using java based test cases tested using selenium webdriver, testNG and run using maven.
2. Server code is tested using pytest and run using command prompt.

Database:

The database is developed using PostgreSQL and connected to the system using flask.
The database has 2 tables:

1. Stores hotel details, including its room availability, location, name of hotel.
2. Another table stores the transactional history eg. Booking or cancellation of hotel.

Experimental setup steps are as below:

1. Test cases are written using LLM's and myself for UI in java.
2. Test cases are written using python for server code.
3. Selenium, webdriver for my version of google chrome, maven and testNG are installed and set and test are run.
4. PyTest is installed using pip and test are run.

8. Evaluation

The **test cases** are actually checking below things:
1. For UI testing, the main purpose is to check if given links or buttons navigate or route to the desired page.
To check this, test cases are written for navigation bar by checking if user click routes to the desired location or not.
Next, Book now button is checked if it is routing to hotel details page or not.

2. For server code, the evaluation is done by checking if the system is showing message if room is booked or cancelled.
On the other side, once booked/cancelled, if the entry is saved in history or not.

Evaluation criteria here is:
1. The number of test cases passed/failed:
The number of tests passed or failed. In this, the total number number of test cases are 8.
All the 8 test cases are passed so that the system is working fine and coverage is 100%.

2. Time taken to complete these test cases:
The total time taken here is

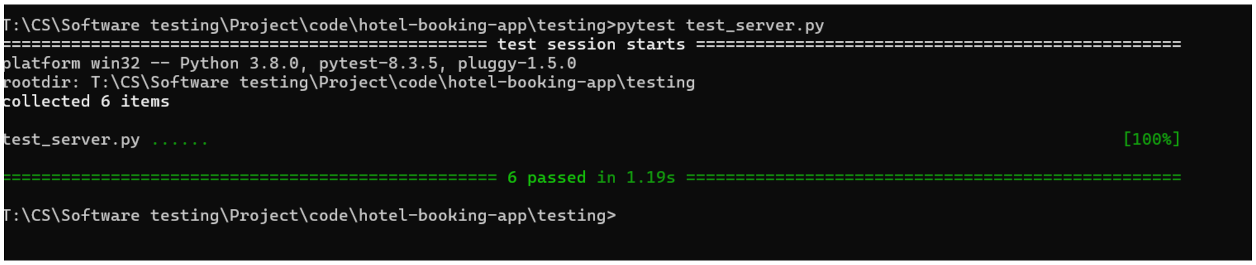
9. Case Study:

The system has below test cases for the whole project:

1. **Home Page Accessibility:**Verifies that the application is accessible and returns a welcome message.
2. **Hotel List Retrieval**: Checks if the endpoint returns a list of hotels, and confirms that each hotel entry contains both a name and location.
3. **Room Booking Functionality**: ensure that a room can be booked and the response confirms the booking success.
4. **Booking Cancellation**: cancels the booking and returns a success message.

5. **Booking History Retrieval**: returns a list of past bookings, and ensures each entry includes the hotel name and the action performed.

Results after we run the test:



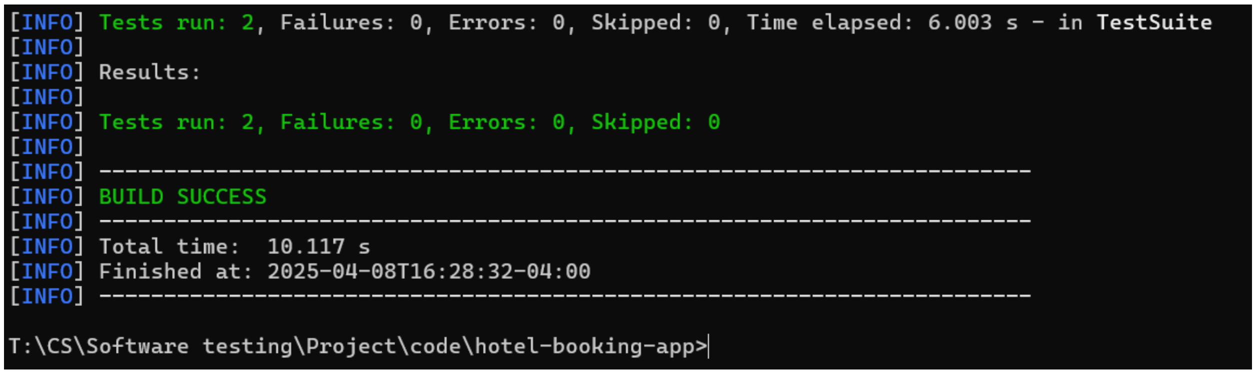
```
T:\CS\Software testing\Project\code\hotel-booking-app\testing>pytest test_server.py
============================================== test session starts ==============================================
platform win32 -- Python 3.8.0, pytest-8.3.5, pluggy-1.5.0
rootdir: T:\CS\Software testing\Project\code\hotel-booking-app\testing
collected 6 items

test_server.py ......                                                                                      [100%]

=============================================== 6 passed in 1.19s ===============================================

T:\CS\Software testing\Project\code\hotel-booking-app\testing>
```

All the test are run and passed successfully in just 1 seconds.Hence, The system is working fine.

UI Test Cases (Java + Selenium + TestNG)

1. **Home Page Navigation (Book Now Button)**: Tests that clicking the "Book Now" button on the homepage redirects the user to the hotel search page.
2. **Navbar Functionality**
   Validates that the main navigation bar works correctly:
   - Clicking the "Search Hotels" link takes the user to the search page.
   - Clicking the "Home" link brings the user back to the homepage.
   - Clicking the "Booking History" link redirects the user to the booking history page

Results after we run UI test using selenium:



```
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 6.003 s - in TestSuite
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  10.117 s
[INFO] Finished at: 2025-04-08T16:28:32-04:00
[INFO] ------------------------------------------------------------------------

T:\CS\Software testing\Project\code\hotel-booking-app>
```

The above shows that it has run and pass 2 test which has multiple test cases inside it and the time taken is 6 seconds.
Hence UI is working well.

10. Limitations:

1. **Focus on Functional Testing:** The project concentrates on functional aspects of the application, specifically user interactions and API endpoints. Non-functional aspects

such as performance, security, and scalability are outside the project's scope and were not evaluated.

2. **Browser and Device Coverage:** While the Selenium-based UI tests cover major browsers, they may not encompass all possible browser versions or mobile devices, potentially limiting the test coverage across diverse user environments.

11. Conclusion:

This project aimed to develop an automated testing framework for a hotel booking application, utilizing Selenium for UI testing and Pytest for server-side code validation. The primary objectives were to enhance testing efficiency, ensure consistent application performance, and improve overall software quality. The outcomes included the successful implementation of automated test scripts that validated both user interactions and backend functionalities, leading to more reliable and maintainable testing processes.

Future Work:

To further advance the testing framework and address current limitations, the following areas are proposed for future development:

1. **Cross-Platform and Cross-Browser Testing:** Expand test coverage to include a wider range of browsers and devices, ensuring consistent user experiences across different platforms. Implementing tools like BrowserStack could facilitate this process.
2. **AI-Driven Test Optimization:** Explore the use of artificial intelligence to analyze test results and optimize test case selection, focusing on areas with the highest risk or recent code changes. This approach could enhance testing efficiency and effectiveness.

12. Data availability

The data used for hotel details is randomly generated database using PostgreSQL.
It has 2 tables one for hotel details and other for transaction history.

13. References:

1. https://www.selenium.dev/documentation/
2. https://flask.palletsprojects.com/
3. https://www.postgresql.org/docs/
4. https://testng.org/doc/
5. https://maven.apache.org/guides/index.html
6. https://developer.chrome.com/docs/chromedriver/downloads
7. https://www.selenium.dev/
8. https://docs.pytest.org/en/stable/