American International University-Bangladesh (AIUB)
Department of Computer Science
Faculty of Science &Technology (FST)
Fall 23 24

Section: B
Software Quality Assurance and Testing

Paying Guest Management System

A Report submitted.
By

SN	Student Name	Student ID
1	SHAKIBUL ISLAM AKASH	20-43861-2
2	MD. OMAR FARUK FAISAL	20-43669-2
3	MIHIR KANTI ROY	20-43785-2

Checked By Industry Personnel

Name:

Designation:

Company:

Sign:

Date:

Software Test Plan

for

<Project>

Version 1.0 approved.

Prepared by <SHAKIBUL ISLAM AKASH, MD. OMAR FARUK FAISAL, MIHIR KANTI ROY

>

<AIUB>

<5/12/23 >

Table of Contents

Revision History

Revision	Date	Updated by	Update Comments
0.1	2023.11.22	SHAKIBUL ISLAM AKASH	First Draft
0.2	2023.11.23	MD. OMAR FARUK FAISAL	Second Draft
0.3	2023.11.25	MIHIR KANTI ROY	Third Draft
0.4	2023.11.28	MD. OMAR FARUK FAISAL	Fourth Draft
0.5	2023.11.30	MIHIR KANTI ROY	Fifth Draft
0.6	2023.12.02	SHAKIBUL ISLAM AKASH	Sixth Draft
0.7	2023.12.05	MIHIR KANTI ROY	Seventh Draft
0.8	2023.12.08	MD. OMAR FARUK FAISAL	Final Draft
0.9	2023.12.10	SHAKIBUL ISLAM AKASH	Revised

1. TEST PLAN IDENTIFIER:RS-MTP01.3

2. REFERENCES

- Software Quality Assurance and Testing cousre slides
- **https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/lifecycle-management/6.0.1?topic=artifacts-reviewing-test**
- **https://www.coleyconsulting.co.uk/references.htm**
- **https://www.onestoptesting.com/test-plan/references.asp**

3. INTRODUCTION

Background to the Problem

The Paying Guest Management System was developed to address issues related to the intricacy and inefficiency of traditional rental procedures for both hosts and visitors. Before this system was put into place, hosts and visitors had trouble using disjointed, manual methods to manage accommodation details, bookings, payments, and reviews.

1. **Fragmented Rental Processes:** Prior to the Paying Guest Management System, hosts most likely used a variety of online platforms or paper records to handle their rental properties, which resulted in a lack of centralized management and coordination**.**
2. **Communication Gaps:** Between hosts, visitors, and administrators, there may not have been clear communication due to the lack of a centralized platform. It is possible that booking requests, payment information, and other important details were communicated over a variety of channels, which could have caused miscommunications.
3. **Booking Uncertainties**: It's possible that the traditional booking approach was unclear since guests lacked a standard booking procedure and hosts had trouble updating room availability. Manual errors and trouble tracking room statuses could arise from this.
4. **Payment Verification Difficulties**: In the absence of a specialized payment verification system, hosts would have had trouble guaranteeing the authenticity of transactions. The absence of an efficient payment procedure may give rise to possible disagreements and doubts.
5. **Restricted Feedback Mechanism:** The lack of an organized review system may have made it more difficult for hosts and guests to give and receive feedback before the Paying Guest Management System. This might have an effect on how good the renting experience is overall.
6. **Manual Payment Handling:** Rent payments may have been manually handled by hosts and administrators, which might have resulted in mistakes, delays, and a lack of transparency in financial activities.

Solution to the Problem

The issues with the conventional rental procedures are fully addressed by the Paying Guest Management System. An efficient resolution of these problems is made possible by the features and functionalities outlined in the background information and introduction:

1. **Centralized Platform:** By offering a centralized platform, the solution solves the issue of disjointed rental processes. With a single, integrated system, hosts can oversee all facets of their vacation rentals, such as room listings, reservations, payments, and reviews.
2. **Efficient Communication:** The system reduces communication gaps by integrating user roles (admin, guest, and host) with a structured communication loop. Booking requests are routinely forwarded to the admin, who either approves or disapproves of them, guaranteeing open and honest communication between all parties.
3. **Real-time Room Status Updates:** Guests can make reservations based on up-to-date information, and hosts can effortlessly alter the status of available rooms. This function removes doubts from the reservation procedure and improves the general effectiveness of handling lodging specifics.
4. **Payment Verification Process:** The admin is involved in a structured payment verification process that is introduced by the system. By ensuring the validity of transactions, this lowers the possibility of problems and conflicts pertaining to payments. For transparency's sake, the payment history is kept and made available to both hosts and visitors.
5. **Feedback Mechanism:** A review system lets visitors and hosts exchange experiences and offer insightful criticism. This improves the whole rental experience and facilitates the process of making well-informed decisions for subsequent reservations.
6. **Automated Payment Handling:** The system allows visitors to pay for their accommodation using a designated method (bkash), which expedites the payment procedure. All pertinent data is saved and reflected in the host and guest dashboards after the admin confirms the payment details. This automation guarantees a clear financial transaction procedure and reduces human mistakes.
7. **Payment Withdrawal for Hosts:** Hosts can use the system to request a withdrawal of their revenue in order to address the manual handling of payments. The administrator streamlines the payment procedure, giving hosts a quick and easy way to get paid.

4. REQUEIREMNT SPECIFICATION

4.1 System Features

- System’s functionalities

1. User Category:

There are 3 types of Users here. They are:

• Host

• Guest

• Admin

2. Feature List:

In this project the “User Type Guest” has the following features:

- Dashboard
- Profile
- Review
- Billing
- History
- Update password
- Logout

In this project the “User Type Host” has the following features:

- Dashboard
- Profile
- Make a post
- View all post
- View booked post
- Review
- History
- Update password
- Logout

In this project the “User Type Admin” has the following features:

- Dashboard
- Billing conformation

1. User Roles:
    - **Admin:** Takes care of booking and payment verification in addition to managing the entire system.
    - **Visitor:** Able to choose and reserve a room, settle bills, and post evaluations.
    - **Host:** Oversees lodging information, lists rooms, and makes withdrawal requests for payments.

2. Room Posting:
    - Rooms with editable data and the ability to activate or deactivate them can be posted by hosts.

3. Booking Process:
    - When a guest books an empty room, the administrator receives a booking request.
    - The booking request may be accepted or denied by the admin.

4. Check-in and Checkout:
    - The guest's check-in page displays the room details if the administrator approves the reservation.
    - The booking status is set to empty after checking out.

5. Review System:
    - Reviews can be shared between guests and hosts, encouraging comments for next enhancements.

6. Payment Process:
    - Rent for guest rooms can be paid with a transaction ID and bkash number.
    - The administrator receives payment requests and verifies them.
    - The information is kept in the host and guest dashboards' history if the administrator approves the payment.
    - The visitor may resubmit payment information if the administrator rejects it.

7. Payment Withdrawal:
    - Once guests have paid the rent for their rooms, hosts are able to request payment withdrawals.
    - The host's requested payment withdrawal is made possible by the admin.
    -

    In general, the system appears to provide a well-thought-out solution, attending to important facets of the rental procedure and offering both hosts and visitors an easy-to-use interface.

Priority Level: High
Precondition: User have valid user id and password.

4.2 System Quality Attributes

Performance:
- Objective: Make sure that tasks are processed quickly and effectively.
- Key Attributes: Quick payment and booking procedures, as well as database query optimization.

Availability:
- Objective: Maintain high system uptime and accessibility.
- Key Attributes: Redundancy, failover mechanisms, regular maintenance schedules.

Scalability:
- Objective: Accommodate a growing user base and feature expansion.
- Key Attributes: Scalable architecture, load balancing, expandable infrastructure.

Security:
- Objective: Safeguard user data, transactions, and system integrity.
- Key Attributes: Secure authentication, encryption, regular security audits.

Reliability:

- Objective: Ensure consistent and error-free system performance.
- Key Attributes: Robust error handling, regular backups, proactive monitoring.

Usability:

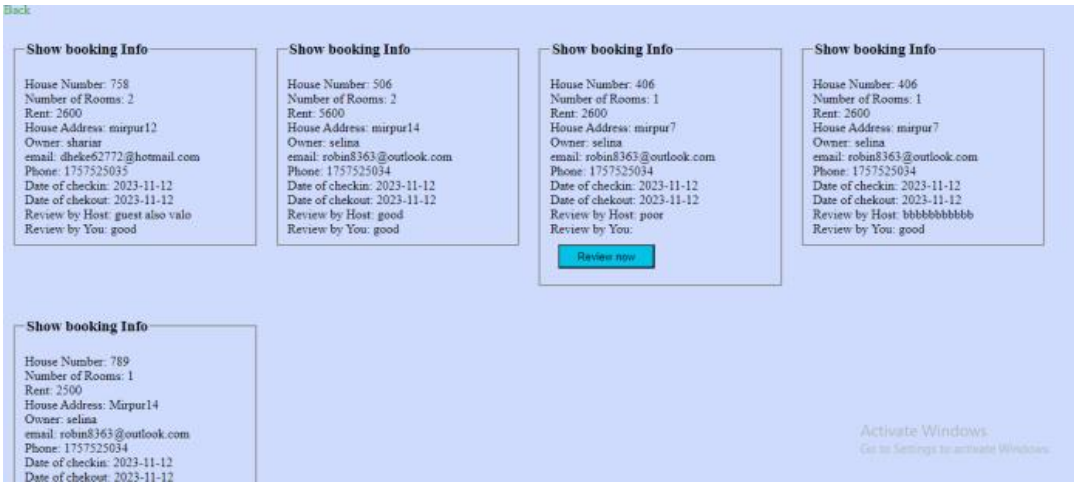
- Objective: Provide an intuitive and user-friendly interface.
- Key Attributes: Intuitive UI design, clear instructions, accessibility features.

Maintainability:

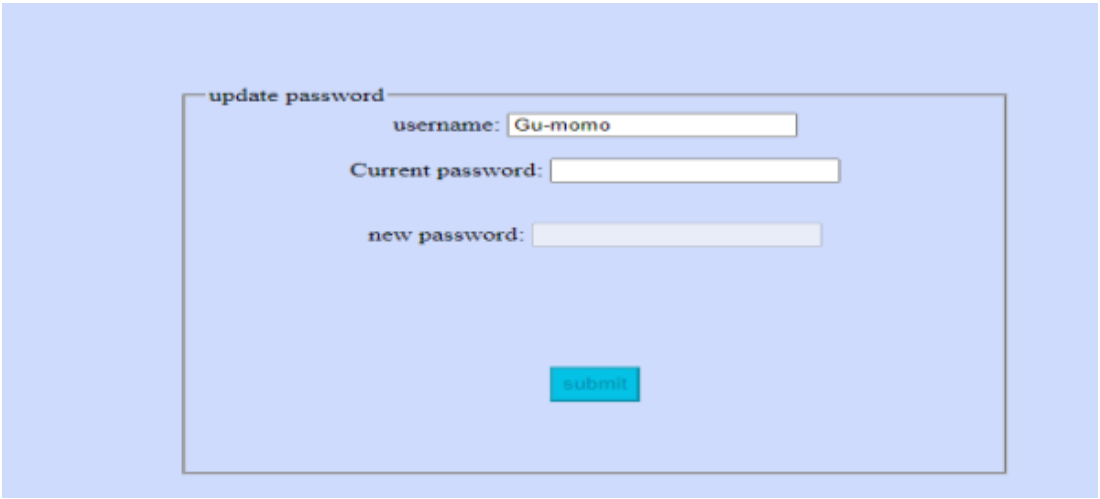
- Objective: Facilitate easy updates, modifications, and maintenance.
- Key Attributes: Well-documented codebase, version control, modular architecture.

4.3 System Interface

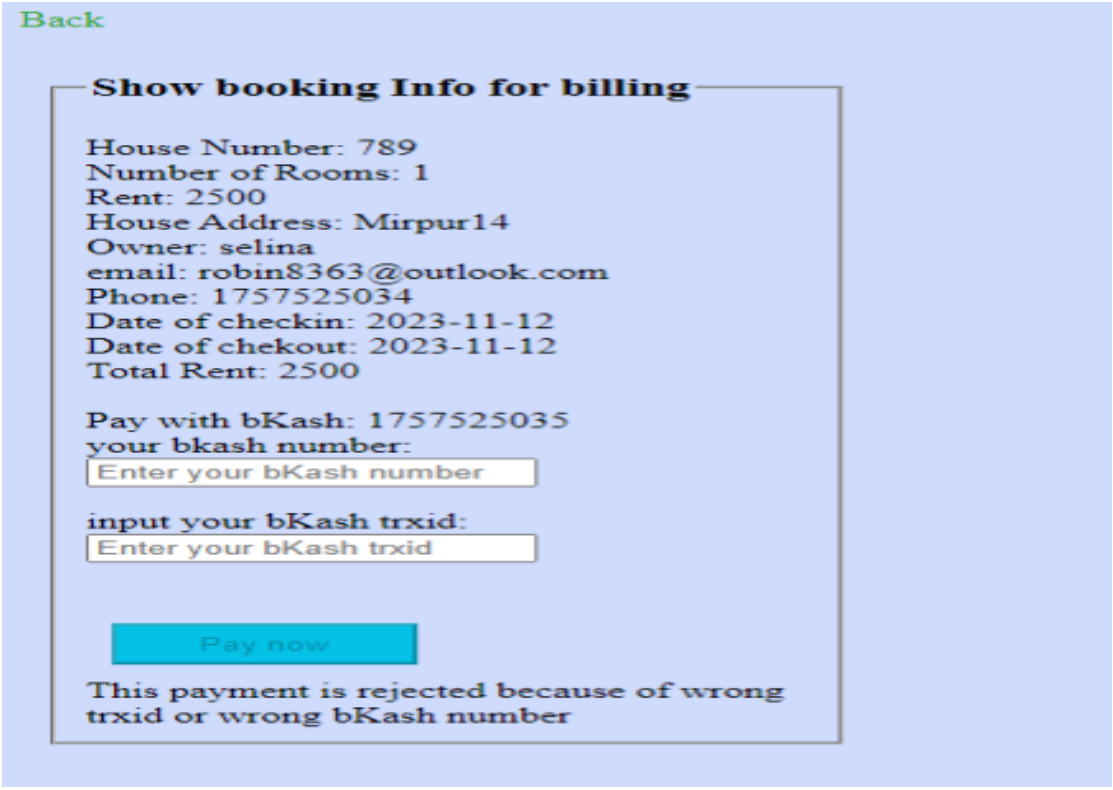
Review:



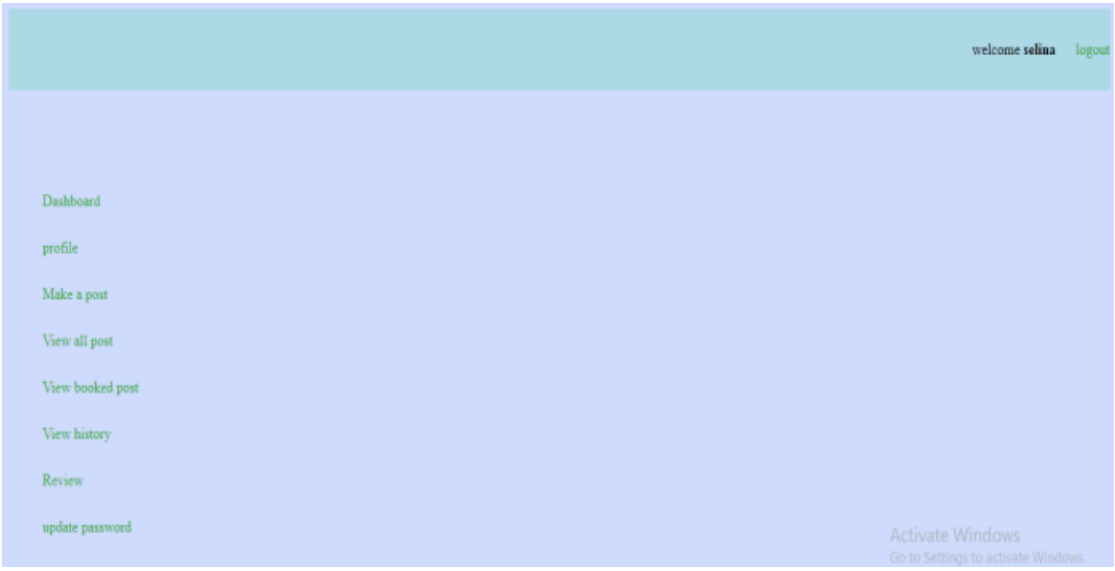
Update password:



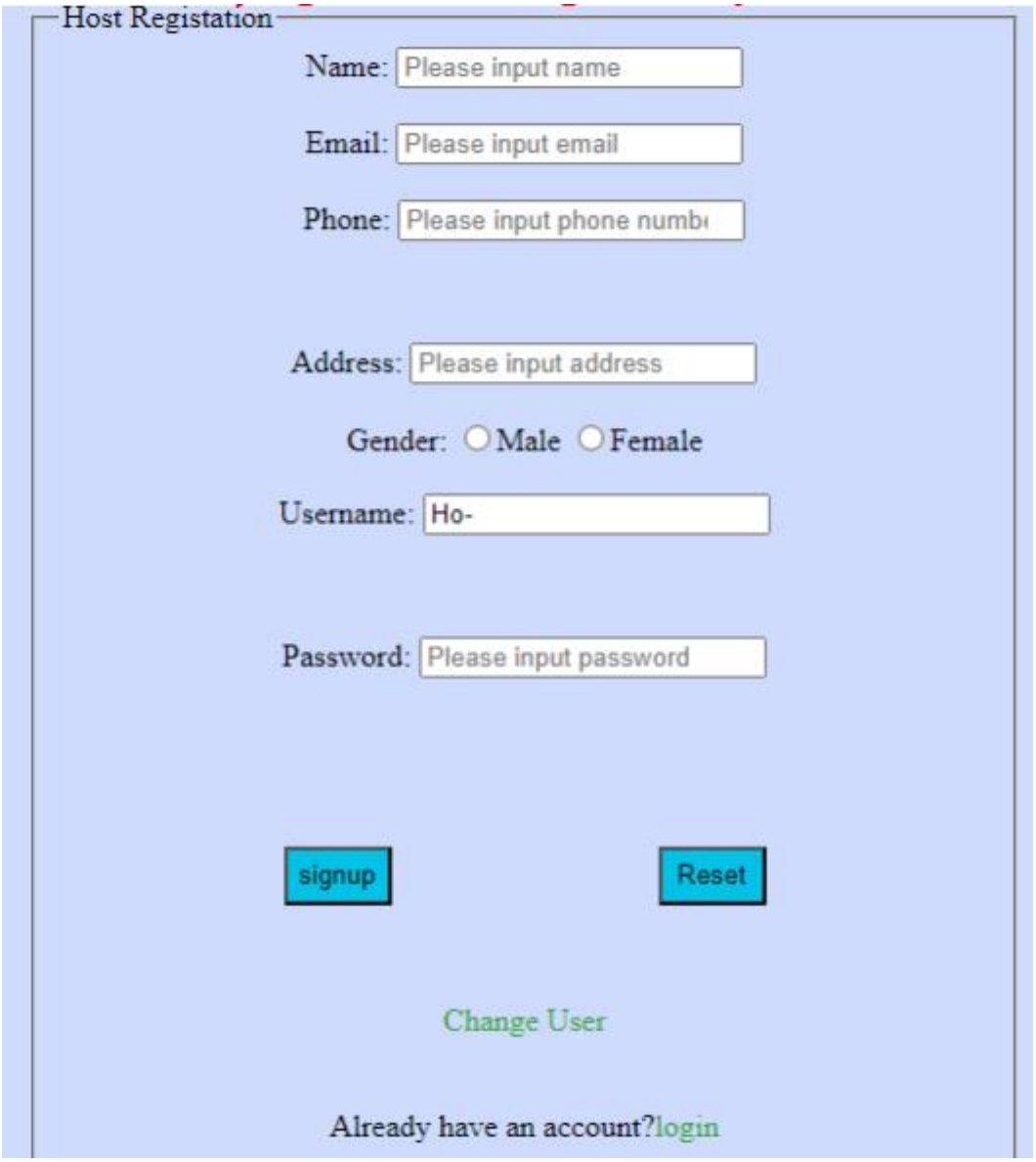
Billing:



Host's dashboard:



Host Registration:



Auth Feature:

4.4 Project Requirements

The overall limitations of a project, such as its time, cost, and risk profile, are known as project constraints. Since project restrictions have an impact on project performance, it is crucial to understand them.

Time: To finish our project, we need **5.5 months or 440 working hours**.

Cost: To build this project we need approximately **437360 bdt**

Resources:

Project Type: Organic
Coefficient < Effect Factor>: 2.4 [P=1.05; T=0.38]
Source Line of Code: SLOC= 4000 Lines

Persons Months, PM = Coefficient < Effect Factor>*(SLOC/1000) P
=2.4*(3000/1000) 1.05
= 7.6

Development Time, DM = 2.50*(PM)T
= 2.50*(7.6) 0.38
= 5.4
= 5.5 months
= 5.5*10*8 working hours
= 440 working hours

Required People, ST = PM/DM
=7.6/ 5.5
=1.4
=2

Developer Salary in 2 Months:

Developer Salary per Working Hour = 400 bdt
Total Developer Salary = 400*440 BDT
=176000 BDT

Requirements Analysis:

Time needed = 10 days (8 working days)
= 8*8 Working hours
= 64 Working hours

Requirement Analysis Person's Hourly wage= 350 bdt

Total Requirement Analysis Expense = 350*64 bdt
= 22400 bdt

Transportation Cost:

Estimated Cost for transportation=8000 bdt

Training and Hardware Expense:

Estimated Cost for Training and Hardware =40000 bdt

Rent Expenses:

Rent per month= 8000 bdt
Total rent in 2 months = 2*8000 bdt
=16000 bdt

Utilities Cost:

Total utilities bill in 2 months =6000 bdt

Maintenance (Till 6 months after delivery):

Expense per hour = 500 bdt
Total Estimated Time needed for maintenance = 72 hours Total
Estimated maintenance cost =72*500 bdt
=36000 bdt

Miscellaneous:

Total Miscellaneous cost =8000 bdt Total

Estimated Expense:

Total Estimated cost = 176000+ 22400+ 8000+ 40000+ 16000+ 6000+ 36000+ 8000
= **312400 bdt**

Profit:

40% of total estimated expense = 312400*40% bdt
=312400+124960

Project Budget: In total (312400+124960) bdt = **437360 bdt**

5. FEATURES NOT TO BE TESTED

The following is a list of the areas that will not be specifically addressed. All testing in these areas will be indirect as a result of other testing efforts. For example:

- Review
- History
- Make a post
- Link confirmation

6. TESTING APPROACH

6.1 Testing Levels

Testing is a crucial phase in software development to ensure that the Paying Guest Management System functions as intended and meets the specified requirements. Here are the testing levels for the system:

1. **Unit Testing:**
    - Objective: Verify the correctness of individual components or units of the system.
    - Key Aspects:
        - Test each function, method, or module in isolation.
        - Identify and fix defects at the code level.
        - Utilize automated testing tools where applicable.
2. **Integration Testing:**
    - Objective: Validate the interactions between integrated components or modules.
    - Key Aspects:
        - Test the interfaces and interactions between different modules.
        - Ensure data flow and communication between components.
        - Verify that integrated components work together seamlessly.
3. **System Testing:**
    - Objective: Assess the entire system's functionality and behavior.
    - Key Aspects:
        - Test end-to-end scenarios, including room posting, booking, payment, and reviews.
        - Validate system functionalities against the specified requirements.
        - Address issues related to system integration.
4. **Acceptance Testing:**
    - **User Acceptance Testing (UAT):**
        - Objective: Validate that the system meets user expectations and business requirements.
        - Key Aspects:
            - Test the system in a production-like environment.
            - Engage real users (admins, hosts, guests) to perform test scenarios.
            - Ensure the system aligns with business goals and user needs.
    - **Alpha and Beta Testing (Optional):**
        - Objective: Obtain feedback from a select group of users before full deployment.
        - Key Aspects:
            - Conduct alpha testing with an internal team.
            - Conduct beta testing with a limited group of external users.
            - Gather user feedback for further refinement.
5. **Regression Testing:**
    - Objective: Ensure that new changes do not negatively impact existing functionalities.

- Key Aspects:
  - Execute test cases that cover critical functionalities.
  - Identify and fix any issues introduced by recent changes.
  - Automated regression testing can be beneficial for repetitive testing scenarios.

6. **Performance Testing:**
   - Objective: Evaluate the system's performance under various conditions.
   - Key Aspects:
     - Test system response times during peak loads.
     - Assess the system's scalability.
     - Identify and address performance bottlenecks.
7. **Security Testing:**
   - Objective: Identify and address potential security vulnerabilities.
   - Key Aspects:
     - Assess the system for potential security risks.
     - Verify secure user authentication and data protection.
     - Implement penetration testing to identify and fix security issues.
8. **Usability Testing:**
   - Objective: Evaluate the user-friendliness and intuitiveness of the system.
   - Key Aspects:
     - Assess the user interface for clarity and ease of use.
     - Ensure consistency in design elements.
     - Obtain user feedback on the overall user experience.

6.2 Test Tools

Working in the AS/400 environment requires us to use the basic commands and utilities that come with the system. The following essential AS/400 commands and utilities can be used as test tools for the Paying Guest Management System.:

1. **Data Queue (DTAQ):**
   - **Commands:** CRTDTAQ, CHGDTAQ, WRKDTAQ are the commands.
   - **Application:** During testing, create data queues to transfer messages between jobs or programs.
2. **Job Logs:**
   - **Commands:** WRKJOB, WRKSPLF
   - **Usage:** Review job logs and spool files for messages and errors generated during test executions.
3. **Debugging Tools (STRDBG):**
   - **Commands:** STRDBG, ENDDBG, ADDWTR, RMVWTR
   - **Use:** During testing, initiate and terminate interactive program debugging sessions.
4. **Batch Job Submission (SBMJOB):**
   - **Commands:** SBMJOB, ADDJOBSCDE
   - **Usage:** Submit batch jobs to mimic scheduled tasks and background processing.
5. **Query Management (STRQRY):**
   - **Commands:** STRQRY and RUNQRY commands.
   - **Usage:** Create and run queries to analyze and verify data during testing.
6. **Line-Of-Command Interface:**
   - **Usage:** Run different AS/400 commands to start particular processes and check system responses.

7. **Command for Message Queues (WRKMSGQ):**
    - **Commands:** WRKMSGQ, SNDMSG
    - **Use:** During testing, keep an eye on message queues for system and application messages.
8. **Job Scheduler (WRKJOBSCDE):**
    - **Commands:** WRKJOBSCDE, ADDJOBSCDE
    - **Usage:** To replicate repeating tasks during testing, schedule jobs to execute at particular periods.
9. **Command for the Data Areas (CRTDTAARA, CHGDTAARA, RTVDTAARA):**
    - **Usage:** Establish and oversee data regions for testing-related data value storage and retrieval.
10. **Command for the File Editor (EDTF):**
    - **Usage:** Modify data and simulate various scenarios during testing by editing source or database files.
11. **WRKSPLF Spool Files:**
    - **Order:** WRKSPLF
    - **Usage:** During testing, check the spool files for output and generated reports.

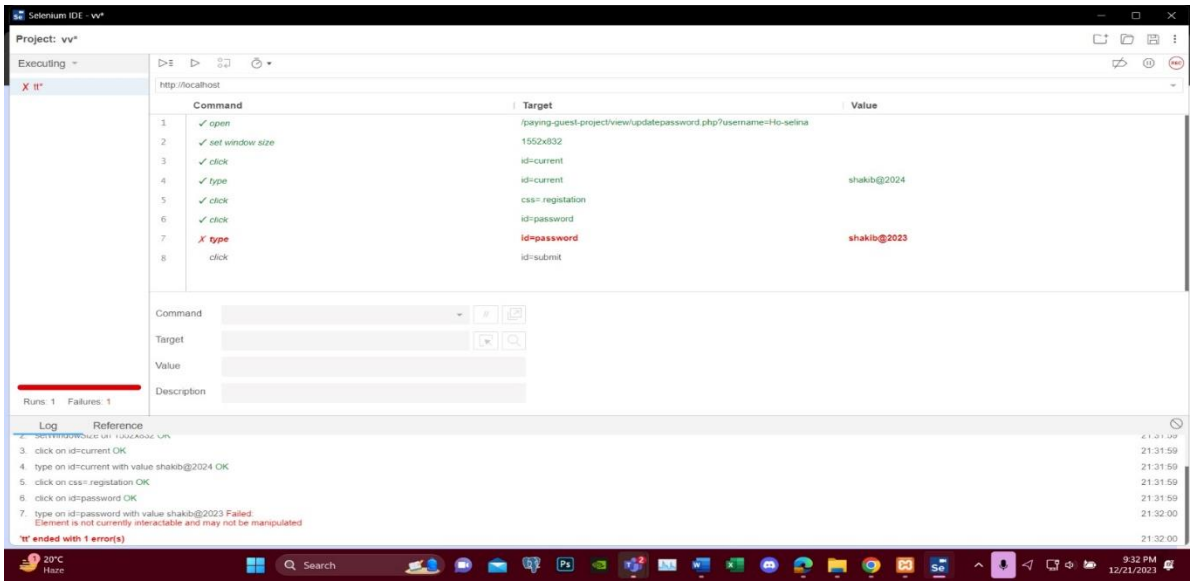
6.3 Meetings

Our team meets twice a week, on Mondays and Wednesdays, for two weeks after the initial draft. A Microsoft Teams meeting took place.

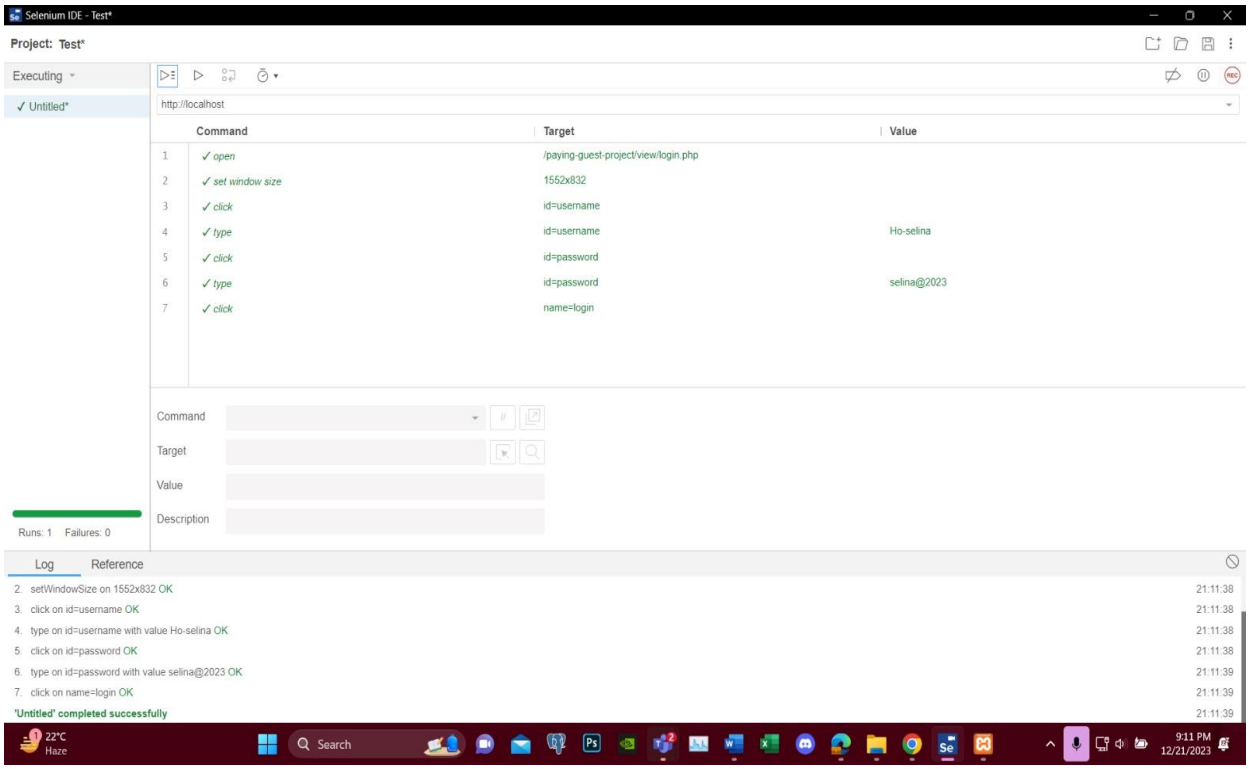
Meeting Date	Meeting Criteria	Objective.
20/11/23	Analysis	○ Functions analysis. ○ Work process analysis.
22/11/23	Progress Evaluation	○ How far the project is prepared. ○ Progress evaluation.
27/11/23	Error Trends	○ Checking the error trends and bugs.
29/11/23	Revise The system	○ Check the whole system. ○ Run the system. ○ Revise all

7. TEST CASES/TEST ITEMS

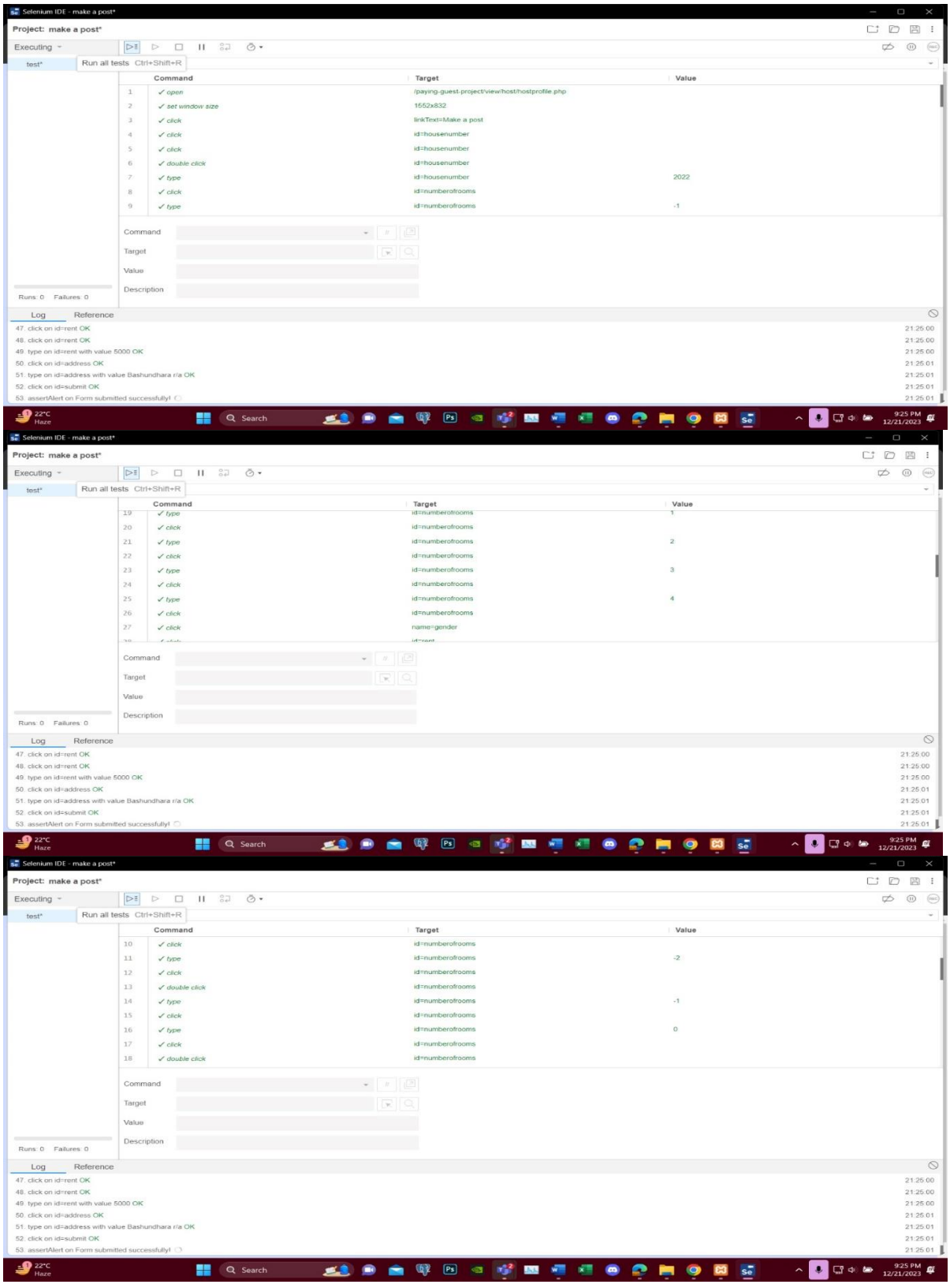
Project Name: Paying Guest Management System			Test Designed by: MD. OMAR FARUK FAISAL	
Test Case ID: PGMS_01			Test Designed date: 16.10.2023	
Test Priority (Low, Medium, High): High			Test Executed by: SHAKIBUL ISLAM AKASH	
Module Name: Update password Process			Test Execution date: 01.11.2023	
Test Title: Verify current password and enter ne password.				
Description: Test update password page				
Precondition (If any): User must know current password.				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Go to the website. 2. Click update password. 3. Enter current password. 4. Enter a new password. 5. Click submit	Current password: 12345@mi New password: 1234567@mi	Users should enter current password.	As expected,	Fail
Post Condition: User is validated with database and successfully login to account. The account session details are logged into the database.				



Project Name: Paying Guest Management System			Test Designed by: MD. OMAR FARUK FAISAL	
Test Case ID: PGMS_01			Test Designed date: 16.10.2023	
Test Priority (Low, Medium, High): High			Test Executed by: SHAKIBUL ISLAM AKASH	
Module Name: Login Session			Test Execution date: 01.11.2023	
Test Title: Verify login with valid username and password.				
Description: Test website login page				
Precondition (If any): User must have valid username and password				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Go to the website. 2. Enter username. 3. Enter password. 4. Click submit	Username: Gu-Mihir Password: 12345@mi	User should login into the application	As expected,	Pass
Post Condition: User is validated with database and successfully login to account. The account session details are logged into the database.				

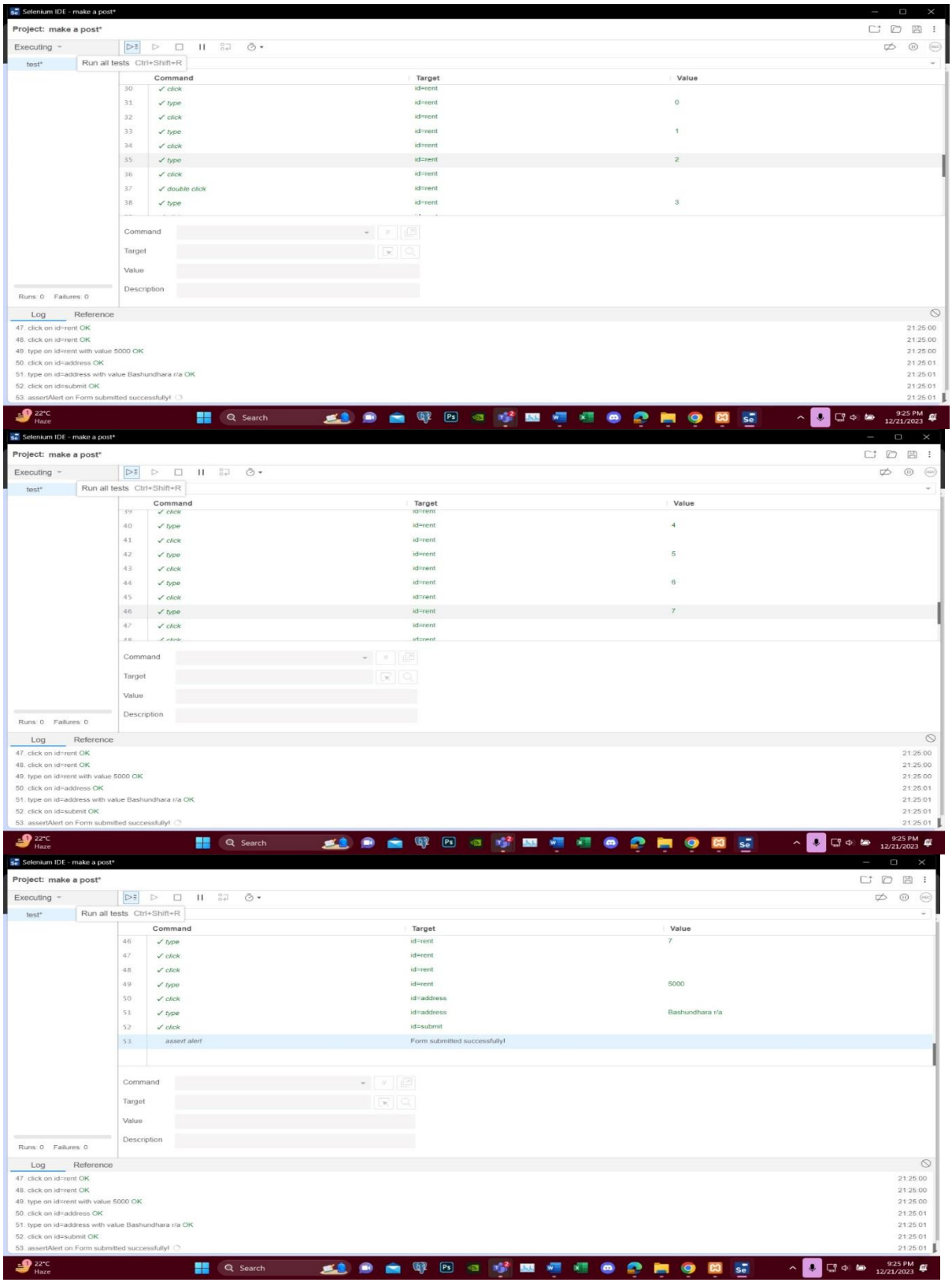


Project Name: Paying Guest Management System			Test Designed by: MD. OMAR FARUK FAISAL	
Test Case ID: PGMS_01			Test Designed date: 16.10.2023	
Test Priority (Low, Medium, High): High			Test Executed by: SHAKIBUL ISLAM AKASH	
Module Name: Registration Process			Test Execution date: 01.11.2023	
Test Title: Verify login with valid username, email, phone number, address password				
Description: Test website Registration page				
Precondition (If any): User must have valid username and password,				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Go to the website. 2. Enter username. 3. Enter Email 4. Enter phone number. 5. Enter Address 6. Enter password. 7. Click submit	Username: Gu-Mihir Email: shaakib.siam33@gmail.com Phone number: 01771800276 Address: Bashundhara R/A Password: 12345@mi	User should register into the application	As expected,	Pass
Post Condition: User is validated with database and successfully login to account. The account session details are logged into the database.				


Selenium IDE - make a post*
Project: make a post*
Executing
test*
Run all tests Ctrl+Shift+R
Command
Target
Value
1 ✓ open /paying-guest-project/view/host/hostprofile.php
2 ✓ set window size 1552x832
3 ✓ click linkText=Make a post
4 ✓ click id=housenumber
5 ✓ click id=housenumber
6 ✓ double click id=housenumber
7 ✓ type id=housenumber 2022
8 ✓ click id=numberofrooms
9 ✓ type id=numberofrooms -1
Description
Runs: 0 Failures: 0
Log
Reference
47. click on id=rent OK 21:25:00
48. click on id=rent OK 21:25:00
49. type on id=rent with value 5000 OK 21:25:00
50. click on id=address OK 21:25:01
51. type on id=address with value Bashundhara r/a OK 21:25:01
52. click on id=submit OK 21:25:01
53. assertAlert on Form submitted successfully! 21:25:01
22°C Haze
Search
9:25 PM 12/21/2023

Selenium IDE - make a post*
Project: make a post*
Executing
test*
Run all tests Ctrl+Shift+R
Command
Target
Value
19 ✓ type id=numberofrooms 1
20 ✓ click id=numberofrooms
21 ✓ type id=numberofrooms 2
22 ✓ click id=numberofrooms
23 ✓ type id=numberofrooms 3
24 ✓ click id=numberofrooms
25 ✓ type id=numberofrooms 4
26 ✓ click id=numberofrooms
27 ✓ click name=gender
Description
Runs: 0 Failures: 0
Log
Reference
47. click on id=rent OK 21:25:00
48. click on id=rent OK 21:25:00
49. type on id=rent with value 5000 OK 21:25:00
50. click on id=address OK 21:25:01
51. type on id=address with value Bashundhara r/a OK 21:25:01
52. click on id=submit OK 21:25:01
53. assertAlert on Form submitted successfully! 21:25:01
22°C Haze
Search
9:25 PM 12/21/2023

Selenium IDE - make a post*
Project: make a post*
Executing
test*
Run all tests Ctrl+Shift+R
Command
Target
Value
10 ✓ click id=numberofrooms
11 ✓ type id=numberofrooms -2
12 ✓ click id=numberofrooms
13 ✓ double click id=numberofrooms
14 ✓ type id=numberofrooms -1
15 ✓ click id=numberofrooms
16 ✓ type id=numberofrooms 0
17 ✓ click id=numberofrooms
18 ✓ double click id=numberofrooms
Description
Runs: 0 Failures: 0
Log
Reference
47. click on id=rent OK 21:25:00
48. click on id=rent OK 21:25:00
49. type on id=rent with value 5000 OK 21:25:00
50. click on id=address OK 21:25:01
51. type on id=address with value Bashundhara r/a OK 21:25:01
52. click on id=submit OK 21:25:01
53. assertAlert on Form submitted successfully! 21:25:01
22°C Haze
Search
9:25 PM 12/21/2023


Selenium IDE - make a post*
Project: make a post*
Executing
test*
Run all tests Ctrl+Shift+R
Command
Target
Value
30 ✓ click id=rent
31 ✓ type id=rent 0
32 ✓ click id=rent
33 ✓ type id=rent 1
34 ✓ click id=rent
35 ✓ type id=rent 2
36 ✓ click id=rent
37 ✓ double click id=rent
38 ✓ type id=rent 3
Command
Target
Value
Description
Runs: 0 Failures: 0
Log
Reference
47. click on id=rent OK 21:25:00
48. click on id=rent OK 21:25:00
49. type on id=rent with value 5000 OK 21:25:00
50. click on id=address OK 21:25:01
51. type on id=address with value Bashundhara r/a OK 21:25:01
52. click on id=submit OK 21:25:01
53. assertAlert on Form submitted successfully! 21:25:01
22°C Haze
Search
9:25 PM 12/21/2023

Selenium IDE - make a post*
Project: make a post*
Executing
test*
Run all tests Ctrl+Shift+R
40 ✓ type id=rent 4
41 ✓ click id=rent
42 ✓ type id=rent 5
43 ✓ click id=rent
44 ✓ type id=rent 6
45 ✓ click id=rent
46 ✓ type id=rent 7
47 ✓ click id=rent
Command
Target
Value
Description
Runs: 0 Failures: 0
Log
Reference
47. click on id=rent OK 21:25:00
48. click on id=rent OK 21:25:00
49. type on id=rent with value 5000 OK 21:25:00
50. click on id=address OK 21:25:01
51. type on id=address with value Bashundhara r/a OK 21:25:01
52. click on id=submit OK 21:25:01
53. assertAlert on Form submitted successfully! 21:25:01
22°C Haze
Search
9:25 PM 12/21/2023

Selenium IDE - make a post*
Project: make a post*
Executing
test*
Run all tests Ctrl+Shift+R
46 ✓ type id=rent 7
47 ✓ click id=rent
48 ✓ click id=rent
49 ✓ type id=rent 5000
50 ✓ click id=address
51 ✓ type id=address Bashundhara r/a
52 ✓ click id=submit
53 assert alert Form submitted successfully!
Command
Target
Value
Description
Runs: 0 Failures: 0
Log
Reference
47. click on id=rent OK 21:25:00
48. click on id=rent OK 21:25:00
49. type on id=rent with value 5000 OK 21:25:00
50. click on id=address OK 21:25:01
51. type on id=address with value Bashundhara r/a OK 21:25:01
52. click on id=submit OK 21:25:01
53. assertAlert on Form submitted successfully! 21:25:01
22°C Haze
Search
9:25 PM 12/21/2023

8. ITEM PASS/FAIL CRITERIA

- The system should correctly capture and store guest information without errors.
- Smooth and timely processing of guest arrivals and departures.
- The system should respond promptly to user input and requests.
- The system should effectively identify and handle errors to prevent disruptions in service.
- An intuitive and easy-to-use interface for staff to navigate and manage guest information.
- Ensuring that the system complies with relevant legal and industry regulations related to guest data and privacy.

9. TEST DELIVERABLES

- Test Plans: Documents outlining the scope, objectives, strategies, and resources for testing the system.
- Test Cases: Detailed descriptions of individual tests designed to evaluate specific system functionalities and requirements.
- Test Reports: Documents summarizing the results of testing, including any identified defects and their severity levels.
- Defect Logs: Lists of identified defects, along with their descriptions, steps to reproduce, and assigned priorities.
- Traceability Matrices: Documents that map test cases to specific requirements, ensuring all requirements are adequately covered by testing.

10. STAFFING AND TRAINING NEEDS

- Staff members should be proficient in using the Guest Management System. Training sessions can cover basic functionalities, data entry,
- If staff members interact directly with guests, training should focus on excellent customer service, effective communication, and problem-solving.
- Given the sensitive nature of guest data, staff should be well-versed in security protocols and best practices to prevent data breaches.
- Training on handling emergency situations, such as evacuations or security incidents, is crucial to ensure the safety of guests and staff.
- Keep staff updated on system changes and provide periodic refresher training to reinforce their knowledge and address any new features or updates.
- Consider cross-training staff members to handle multiple aspects of the Guest Management System. This can enhance flexibility and ensure continued operations during staff absences.

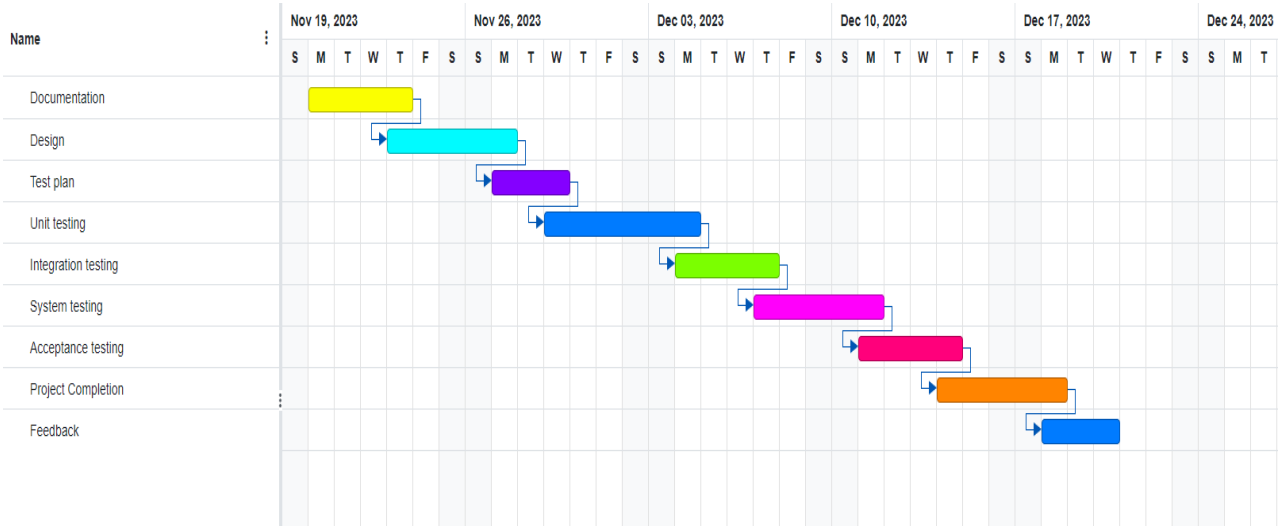
This comprehensive approach ensures that the staffing structure is well-aligned with the operational needs of the Guest Management System, and the training programs address the specific skills and knowledge required for each role.

11. RESPONSIBILITIES

	TM	PM	Dev Team	Test Team	Client
Acceptance test Documentation & Execution	☑	☑		☑	☑
System Integration test Documentation & Exec.	☑		☑	☑	
Unit test documentation & execution	☑		☑	☑	
System Design Reviews	☑	☑	☑	☑	☑
Detail Design Reviews	☑	☑	☑	☑	
Test procedures and ules	☑	☑	☑	☑	
Screen & Report prototype reviews			☑	☑	☑

Change Control and regression testing	☑	☑	☑	☑	☑

12. TESTING SCHEDULE



13. PLANNING RISKS AND CONTINGENCIES

➢ Delays in identifying and resolving defects could lead to missing deadlines.
➢ The project may go over budget if extra resources are required to complete it.
➢ Inadequate testing could result in poor quality outcomes.
➢ Failure to understand the customer's needs may result in their not meeting their expectations.
➢ Insufficient security testing could pose security risks.
➢ In order to manage project risks, a quality assurance plan should be developed for software quality testing. A risk management strategy must be used to detect, assess, and mitigate risks. during software quality testing. Consistent code reviews may assist in detecting potential issues.

14. APROVALS

Project Sponsor - Steve Sponsor	
Development Management - Ron Manager	
EDI Project Manager - Peggy Project	
RS Test Manager - Dale Tester	
RS Development Team Manager - Dale Tester	
Reassigned Sales - Cathy Sales	
Order Entry EDI Team Manager - Julie Order	