Test Plan Documentation

ONLINE GUEST HOUSE BOOKING SYSTEM

G.V.S.NEEHARIKA(22CS10028)
KARANAM KEERTHI CHANDRIKA(22CS30035)
VURITI KEERTHANA(22CS10081)

CONTENTS

1.TEST PLAN IDENTIFIER

TEST PLAN IDENTIFIER: "GH-TP-001: Functional Testing of Guest House Booking System."

2.REFERENCES

1. Database Management-https://youtu.be/cYWiDiIUxQc
2. Flask Tutorial-https://youtu.be/dam0GPOAvVI
3. Datetime Module-Python datetime module
4. Flask Documentation-https://flask.palletsprojects.com/en/2.2.x/
5. smtplib-How to Send Automated Email Messages in Python - GeeksforGeeks
6. google credentials-bad credentials using gmail smtp

3.INTRODUCTION

This is a master plan designed to guide the development and implementation of our new online booking system for our guesthouses**.**The purpose of this plan is to provide a comprehensive overview of the project,resource identification, budget constraints,and highlights the testing effort and evaluation activities that will be undertaken to ensure the system's success.

The scope of the project includes the design, development, testing, and deployment of the online booking system. This plan is related to our reference documents, including our software requirements specification document,.. The scope of our project is limited to development of the online guest house booking system but does not include any modifications.

In summary, this plan is a vital roadmap for successful completion of the Online

Guesthouse Booking System. The plan’s objectives,scope and its constraints are clearly defined to ensure that its functioning is perfect.

4.TEST ITEMS(FUNCTIONS)

The test items for our Online Guest House Booking System are :

1. User registration and login functionalities .
2. Search and filtering of available guest houses by date, location, and amenities.
3. Booking a Guesthouse with required details like check in/out details, number of guests, and room preferences.
4. Verification of booking confirmation and reservation details.
5. Cancellation of booking and refund processing details.
6. Secured payment processing and storing the details.
7. Integration with third party apps like gmail, google maps for OTP, username, password, location details.
8. Handling of exceptional cases like invalid user input, system errors, system crashes.
9. Availability of reports such as booking reports, occupancy reports, and revenue reports.

These test items can be organised by the functional area of the test plan. By testing these items we can ensure that our Online Guesthouse Booking System is reliable, user-friendly, and efficient for use.

5. SOFTWARE RISK ISSUES

1. **Inconvenience with third parties:** apps like delay of payments, OTP's,..

2. **Incomplete or unclear requirements :** Requirements that are unclear or incomplete may lead to misunderstandings and incorrect implementation of the system.
3. **Complex functions :** Online Guesthouse Booking systems have complex functions such as searching, filtering, booking, payment, OTP process.These functions may have inherent software risks such as incorrect calculations, data integrity issues, and performance issues.
4. **Modifications to components with a past history :** The online booking system may have components that have had past failures or defects. Modifying these components may throw us into new risks or amplify existing ones.
5. **Multiple interfaces :** The Online Guesthouse Booking System may have multiple interfaces with other systems or third party apps such as payment, email, notifications. These interfaces can introduce the risk of communication failures.
6. **Poorly documented modules :** Lack of proper

documentation of modules or change requests can lead to misunderstandings, incorrect implementation, and future maintenance problems.

6.FEATURES TO BE TESTED

1. User Registration and Login
2. Room Availability and Booking
3. OTP and Mail functionalities

4. Search and Filter
5. Modifications and Updates after booking a room
6. Feedback and Reviews
7. Security and Performance

7.FEATURES NOT TO BE TESTED

1. Third party apps integration
2. Browser Compatibility
3. Hardware Compatibility
4. User Interface design
5. Business Logics and Calculations

8.APPROACH(STRATEGY)

1. OVERALL TESTING STRATEGY

The test strategy for this plan should be to test the application from the user’s perspective, including all features and functionalities,ensuring the system is user friendly, securable and scalable through unit testing, integration testing, system testing, acceptance testing.

2.CONFIGURATION MANAGEMENT (CHANGE CONTROL)

The full components of the online guesthouse booking system will be tested and any elements in the requirements that do not make sense or

untestable will be processed by engaging with the engaged team.The system will be version controlled using git, all changes will be tracked through github.

The team will have their own branch for making changes and changes will be only merged after cross checking their entire functioning. Before any changes are made a backup of the current system will be created to ensure that the system can be rolled back in case of any issues or errors.

3.TESTING TOOLS

1. **Pytest** : Pytest is a popular testing framework for python.it supports testing of different types of software projects including web applications like flask. It can be used to write and execute test cases for the flask application.
2. **Selenium** : Selenium is an automated testing tool used for web applications.

It can be used to test functionality and user interface of the web application. You can use selenium with python to automate testing of your flask application.

3. **Flask-Testing :**It is an extension for flask that provides tools for flask applications. It allows you to write unit and integration tests for your flask application using python.
4. **Coverage.py :** Coverage.py is a tool that helps to measure code coverage of your test suite. It can be used to identify areas of your code that are not covered by your tests. You can use Coverage.py with Flask and SQLAlchemy to measure code coverage of your application.
5. **Behave :** Behave is a testing framework for behaviour-driven development(BDD) that can be used with python. It allows you to write tests in natural language format, making it easier to understand and collaborate with stakeholders

4.METRICS AND MEASURES

The following metrics are collected :

1. The number of users registered.
2. The number of rooms booked.
3. The list of amenities associated with each room booked.
4. The food menu ordered by each room.
5. The number of OTPs generated successfully.
6. The number of mails successfully sent regarding username and password details.

5.MEETINGS

Our team will meet once everyday to discuss,evaluate progress, learn things that are required for our project, and to identify errors and problems as early as possible. We have planned so well to complete the submissions and to get the day to day progress in our online guesthouse booking system project.

9. STAFFING AND TRAINING NEEDS

- **Staffing Needs:**
  - ○ **1. Customer Support Representatives:** The online software guest house booking software will need customer support representatives to assist customers with their booking and address any issues that may arise.
  - ○ **2. Technical Support Staff:** The software may require technical support staff to ensure that the platform is running smoothly,and to troubleshoot any issues that may arise.
  - ○ **3. Marketing and Sales Staff:** The software will need marketing and sales staff to promote the software and to attack more customers. ●

Training Needs:

- ○ **1.Technical Training:** Staff will require technical training to understand the software and to troubleshoot any issues that customers may face.

- ○ **2. Customer Service Training:** Customer service representatives will require training on how to effectively communicate with customers and handle complaints or issues.
- ○ **3. Sales Training:** Marketing and sales staff will require training on how to promote the software and how to close sales.

10.ENVIRONMENTAL NEEDS

1. Hardware : The system requires hardware resources such as servers, storage devices, network infrastructure, and user devices such as desktops, laptops, tablets, smartphones.

2. **Software :** The system requires various software components such as operating systems, web servers, databases, programming languages, and development tools.

3. **Internet Connectivity :** The system requires a stable and high-speed internet connection to enable users to access the online platform.

4. **Security :** The system requires security measures such as firewalls, encryption, access controls, and security testing to ensure the sensitive user data such as personal information and payment details.

5. **Scalability :** The system needs to be scalable to accommodate increasing traffic and demand as the number of users and bookings grow over time. 6. **Availability :** The system needs to be available 24/7 to enable users to book guest houses at any time.

7. **Performance :** The system needs to perform well under various load conditions, including peak usage periods.
8. **Testing Environment :** The system requires a dedicated testing environment that is separate from the production environment to entire that the testing activities do not interfere with normal system operations.
9. **Maintenance and Support :** The system needs to comply with relevant laws and regulations governing online transactions, data privacy, and security.

11.TEST DELIVERABLES

1. **Test plan document :** It is what we are submitting now which has a detailed description of our project’s scope , objectives, test cases, approach and strategy.

2. **Test cases :** They are the various real time situations we create while we book a room using our online guesthouse booking system, which are given to you in our test plan document.

3. **Test Design Specifications :**

```
Flask == 3.0.2
Flask-SQLAlchemy == 3.1.1
Jinja2 == 3.1.3
MarkupSafe == 2.1.5
SQLAlchemy== 2.0.28
Werkzeug== 3.0.1
click== 8.1.7
Greenlet == 3.0.3
import lib-metadata == 6.1.7
Itsdangerous == 2.1.2
pip == 24.0
setuptools == 58.0.4
typing-extensions == 4.10.0
zipp == 3.18.1
```

12.Responsibilities:

	Dev Team Test Team Client
Acceptance test Documentation & Execution	x x
System/Integration test Documentation & Execution	x x
Unit test Documentation & Execution	x x
System Design Reviews	x x x
Test Procedures & Rules	x x

13.SCHEDULE:

The testing phase of the Attendance Application will be divided into three main

parts: Unit Testing, System/Integration Testing, and Acceptance Testing.

• **Unit Testing:** Unit testing will commence concurrently with the development phase, with the final unit testing scheduled to be completed by 6nd April 2024.During this phase, individual components or modules of the system will be tested independently to ensure their proper functionality.

- **System/Integration Testing:** Scheduled for 9th April, this phase involves testing the entire system once all individual components have been integrated. The Objective is to verify that different parts of the system work seamlessly together.

- **Acceptance Testing:** Also scheduled for 9th April, this phase focuses on testing the system from the perspective of end-users to ensure it meets their requirements and expectations. Testing will be conducted by a group of individuals under the supervision of the development team, and results will be evaluated against predefined acceptance criteria.Upon completion of each testing phase, the results will be analysed, and necessary changes will be implemented in the system’s codebase.

Please note that the provided dates are tentative and subject to adjustments based on the progress of development, resource availability, and any other factors influencing the testing schedule.

14.PLANNING RISKS AND CONTINGENCIES

- **Data Security:** Risk of data breaches or unauthorized access. Contingency: Implement robust security measures such as encryption, access controls, and regular security audits.
- **Performance Issues:** Risk of slow loading times or system crashes under heavy load.
  Contingency: Optimise code, use caching mechanisms, and perform load testing to identify and address performance bottlenecks.
- **Compatibility Issues**: Risk of application behaving differently across different browsers or devices. Contingency: Test application across multiple browsers and devices, and implement responsive design principles to ensure compatibility.