SOFTWARE REQUIREMENT SPECIFICATION

System Title: *Web Based Reservation and Scheduling System in Isdaan Floating Restaurant in Calauan, Laguna*

Members:
Hallig, Siera Q.
Macasaet, Kate Ashley A.
Plata, Giannah Faith F.
Torres, Tracy SJ.

Introduction

In today’s rapidly evolving digital era, the restaurant industry faces unprecedented challenges and opportunities. Customers now expect fast, convenient, and contactless services, driven by advances in technology and changes in consumer behavior. To remain competitive and meet these growing expectations, restaurants must modernize their operations by adopting innovative digital solutions that enhance efficiency and improve customer experience.

The Isdaan Floating Restaurant in Calauan, Laguna, is a well-known dining destination that has traditionally managed its reservation system manually, relying heavily on walk-in customers and phone-based bookings. While this approach worked in the past, it has become increasingly inadequate in recent years, especially during the COVID-19 pandemic. The pandemic introduced new challenges such as the need to regulate customer traffic, enforce social distancing, and comply with strict health and safety protocols. These factors exposed the limitations of the manual reservation system, resulting in inefficiencies, longer wait times, and difficulties in managing customer data and preferences.

To overcome these challenges and adapt to the new normal, this project proposes the development of a comprehensive Web-Based Reservation and Scheduling System tailored specifically for the Isdaan Floating Restaurant. This system aims to digitize and automate the entire reservation process, allowing customers to easily book tables online at their convenience, while providing restaurant staff with a robust tool to manage bookings, schedules, and customer interactions more effectively.

The proposed platform will feature both mobile and web-based components, ensuring accessibility across devices and enhancing user engagement. Key functionalities include real-time availability checking, automated confirmation and reminders, and a centralized dashboard for staff to monitor and adjust reservations dynamically. By streamlining the reservation workflow, the system is expected to reduce manual errors, optimize seating arrangements, and improve overall operational efficiency.

Moreover, the system will enhance customer satisfaction by offering a seamless, intuitive interface that simplifies the booking experience and fosters better communication between the restaurant and its patrons. This digital transformation not only aligns with current technological trends but also positions the Isdaan Floating Restaurant as a forward-thinking establishment ready to meet future demands.

In summary, this Web-Based Reservation and Scheduling System represents a strategic response to the evolving needs of both the restaurant and its customers, aiming to deliver improved service quality, operational excellence, and a safer dining environment in the post-pandemic world.

Purpose

The primary purpose of this system is to digitalize and automate the reservation and scheduling processes of the Isdaan Floating Restaurant, eliminating the inefficiencies and limitations of the current manual methods. This software will enable customers to conveniently schedule, modify, or cancel reservations remotely through an intuitive web-based interface, providing greater flexibility and transparency. On the administrative side, the system offers a centralized dashboard for staff to efficiently monitor and manage bookings in real-time, minimizing issues such as double bookings and no-shows. Automated notifications and reminders will enhance communication between the restaurant and its patrons, improving overall customer satisfaction and operational reliability.

Beyond addressing the immediate needs of the Calauan branch, the system is designed as a scalable platform to support future expansion across other branches or similar establishments. By streamlining reservation workflows and optimizing seating arrangements, the system aims to guarantee seating availability, reduce operational workload, and foster better customer engagement. This digital transformation not only modernizes the restaurant's booking process but also positions Isdaan Floating Restaurant for sustainable growth and improved service quality in the evolving hospitality landscape.

Overall Description

The system is a web-based application designed to be fully compatible with mobile devices, ensuring accessibility and convenience for users across different platforms. It supports two primary user roles: Customers and Administrators. Customers can easily create personal accounts, make new reservations, modify or cancel existing bookings, check the status of their reservations, and receive timely notifications such as confirmations and reminders. This functionality aims to provide a seamless and user-friendly experience that encourages engagement and reduces the need for direct communication with restaurant staff.

Administrators have access to a dedicated management interface that allows them to efficiently handle incoming reservation requests. They can approve or deny bookings based on availability and operational considerations, as well as monitor and analyze historical reservation data to support decision-making and improve service delivery. The system's backend will be developed using Firebase, which offers a scalable and real-time database solution, while the front-end development will leverage JavaScript within the Visual Studio Code environment to create a responsive and interactive user interface. The project will follow the Rapid Application Development (RAD) methodology, enabling quick prototyping and iterative feedback cycles. This approach ensures that the system evolves in close alignment with user needs and delivers functional, reliable features within a reduced development timeframe.

Functional Requirements

- **Customer Functions:**

  - ✓ Account creation and login via OTP
  - ✓ Reservation request submission (date, time, cottage)
  - ✓ Reservation status tracking (Pending, Approved, Denied)
  - ✓ Profile management
  - ✓ Communication with admin via contact page

- ❖ **Administrator Functions:**
  - ✓ Admin login and access to the dashboard
  - ✓ Viewing and managing booking requests
  - ✓ Approving or denying reservations
  - ✓ Access to list of registered users
  - ✓ Historical tracking of past reservations
  - ✓ Search functionality for customers and reservations

Non-functional Requirements

- ➢ **Usability:** The system should be intuitive and easy to use for all user levels, with a clean and responsive interface.
- ➢ **Performance:** The system must support real-time updates and load reservation data promptly.
- ➢ **Reliability:** The system must be available 99% of the time and must handle failures gracefully.
- ➢ **Scalability:** Capable of supporting more users as other branches adopt the system.
- ➢ **Security:** Must ensure secure login through OTP verification and protect user data via Firebase authentication.
- ➢ **Portability:** The mobile version should function well on both Android smartphones and tablets.

System Features and Interfaces

The system will feature:

- **User Interfaces:**
  - **Customer:** Registration, login (OTP), reservation form, reservation status viewer, profile editor
  - **Admin:** Login, dashboard, request approval page, registered users list, reservation history viewer
- **System Interfaces:**
  - Integration with Firebase for data storage and authentication
  - SMS API for OTP verification
- **Hardware Interfaces:**
  - Mobile Devices (Android)
  - Desktop or Laptop with web browser for administrators

Assumptions and Constraints

Assumptions

- Users will have reliable internet access when interacting with the system, whether via web or mobile platforms.
- The mobile application is primarily intended for use within the Laguna region, specifically targeting customers of the Calauan branch.
- Administrators and staff are assumed to have received basic training on how to operate and manage the system effectively.
- Users possess a basic level of digital literacy sufficient to navigate the web and mobile interfaces without extensive support.

Constraints

- The initial deployment of the system is limited to a single branch—the Isdaan Floating Restaurant in Calauan—restricting its immediate geographic reach.
- At launch, the mobile application will only support Android devices, with plans for iOS compatibility considered for future updates.
- The system's backend and data storage rely heavily on Firebase services, making it dependent on the availability, performance, and pricing policies of this third-party platform.
- Any changes or downtime in Firebase services may directly impact system functionality and availability.
- Development and deployment timelines are constrained by the Rapid Application Development (RAD) model, which prioritizes quick iterations but may limit extensive feature customization during initial releases.

Use Case Descriptions

Use Case: Customer Reservation

- Actor: Customer
- Precondition: Customer has an account and is logged in
- Main Flow:
    1. Customer logs into the system using OTP verification.
    2. Customer selects date, time, and cottage.
    3. Customer submits reservation request.
    4. System confirms submission and shows "Pending" status.
    5. System updates status to "Approved" or "Denied" after admin action.
- Postcondition: Reservation is either scheduled or declined.

Use Case: Admin Approval

- Actor: Administrator
- Precondition: Admin is logged into the system
- Main Flow:
    1. Admin logs in.
    2. Admin views new reservation requests.
    3. Admin reviews details (date, time, cottage).

4. Admin approves or denies the request.
5. Customer is notified of the decision.

- Postcondition: Reservation status is updated in the system.

Testing Tool Documentation

I. Tool Used

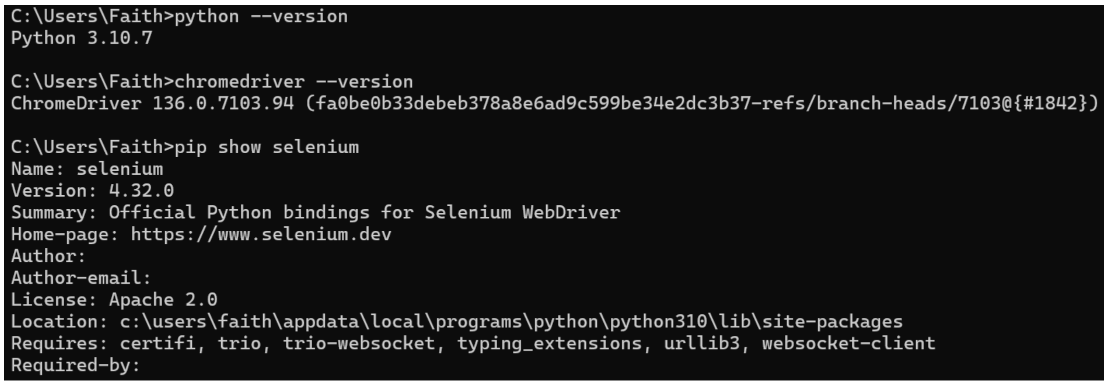
- **Selenium**

Selenium was used because it is easy tool and an open-source framework for testing web applications especially for the language used in this system which is JavaScript. There were 3 main tests that were made for the main functions to show all functionalities if working properly. The three main webpages that was tested are the admin-dashboard, admin-login, and index.

Type of testing per HTML:
admin-dashboard.html – page load & title verification, sidebar tabs visibility test, dashboard widget visibility test, dropdown menu interaction test
admin-login.html – functional testing, URL redirection testing, content validation after login
index.html – UI testing, element visibility testing, modal interaction testing, navigation testing

II. Test Environment

The browser used for testing is **Google Chrome** and the installed WebDriver used is **ChromeDriver** along with installing selenium by opening the command prompt and enter pip install selenium. After installing all the required application, we made sure that they are installed properly by checking the versions of it in the command prompt.

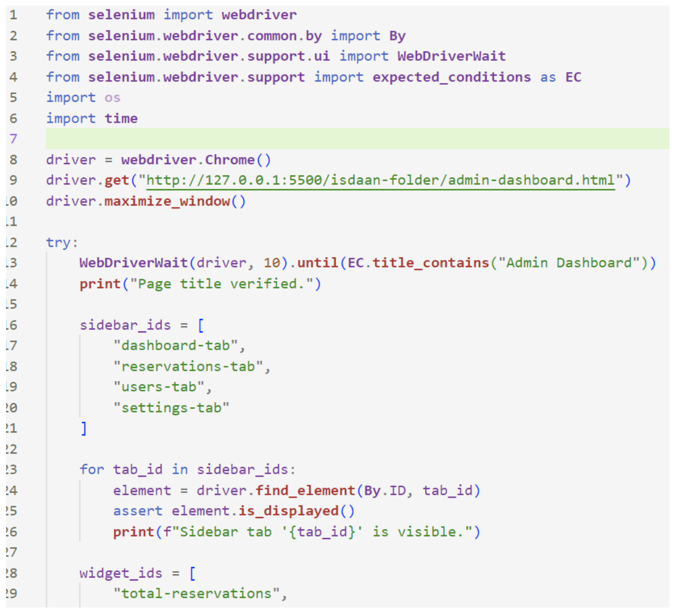


```
C:\Users\Faith>python --version
Python 3.10.7

C:\Users\Faith>chromedriver --version
ChromeDriver 136.0.7103.94 (fa0be0b33debeb378a8e6ad9c599be34e2dc3b37-refs/branch-heads/7103@{#1842})

C:\Users\Faith>pip show selenium
Name: selenium
Version: 4.32.0
Summary: Official Python bindings for Selenium WebDriver
Home-page: https://www.selenium.dev
Author:
Author-email:
License: Apache 2.0
Location: c:\users\faith\appdata\local\programs\python\python310\lib\site-packages
Requires: certifi, trio, trio-websocket, typing_extensions, urllib3, websocket-client
Required-by:
```

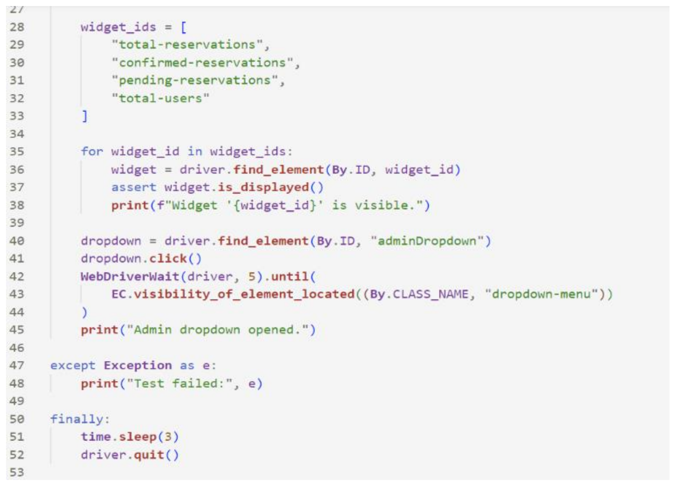
The testing was done on a server because the system's data base needs to be connected to an open live server. The operating system used was windows 11.
The specific configurations needed was to install Python and the version of the Chromedriver should be the same as with the Google Chrome for the connection to work properly. When the chromedriver is installed, it should be copied to the PATH in order to work properly.

III. Results

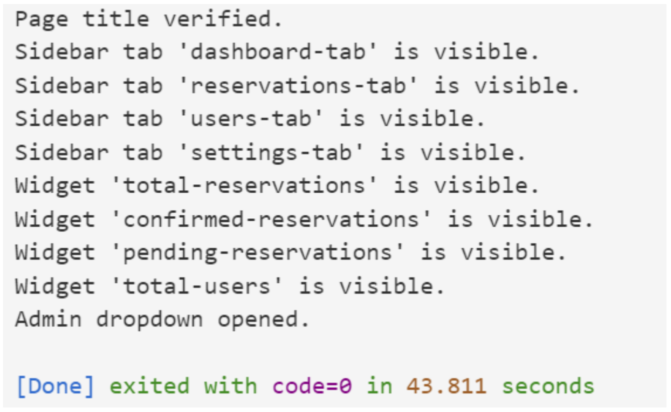
1. **admin-dashboard.html**



```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import os
import time

driver = webdriver.Chrome()
driver.get("http://127.0.0.1:5500/isdaan-folder/admin-dashboard.html")
driver.maximize_window()

try:
    WebDriverWait(driver, 10).until(EC.title_contains("Admin Dashboard"))
    print("Page title verified.")

    sidebar_ids = [
        "dashboard-tab",
        "reservations-tab",
        "users-tab",
        "settings-tab"
    ]

    for tab_id in sidebar_ids:
        element = driver.find_element(By.ID, tab_id)
        assert element.is_displayed()
        print(f"Sidebar tab '{tab_id}' is visible.")

    widget_ids = [
        "total-reservations",
```

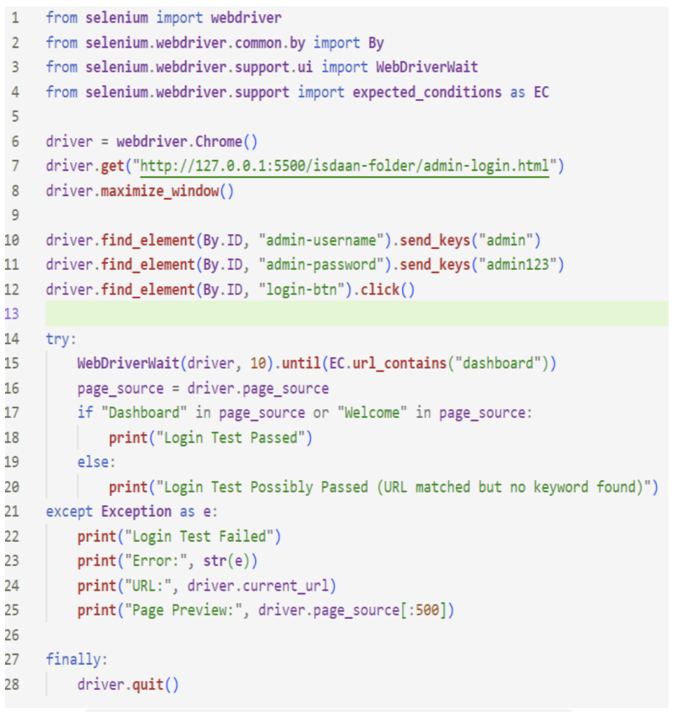


```python
    widget_ids = [
        "total-reservations",
        "confirmed-reservations",
        "pending-reservations",
        "total-users"
    ]

    for widget_id in widget_ids:
        widget = driver.find_element(By.ID, widget_id)
        assert widget.is_displayed()
        print(f"Widget '{widget_id}' is visible.")

    dropdown = driver.find_element(By.ID, "adminDropdown")
    dropdown.click()
    WebDriverWait(driver, 5).until(
        EC.visibility_of_element_located((By.CLASS_NAME, "dropdown-menu"))
    )
    print("Admin dropdown opened.")

except Exception as e:
    print("Test failed:", e)

finally:
    time.sleep(3)
    driver.quit()
```

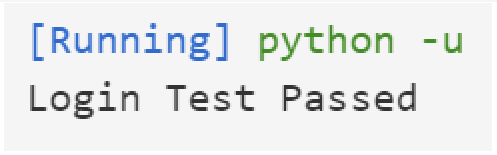


```
Page title verified.
Sidebar tab 'dashboard-tab' is visible.
Sidebar tab 'reservations-tab' is visible.
Sidebar tab 'users-tab' is visible.
Sidebar tab 'settings-tab' is visible.
Widget 'total-reservations' is visible.
Widget 'confirmed-reservations' is visible.
Widget 'pending-reservations' is visible.
Widget 'total-users' is visible.
Admin dropdown opened.

[Done] exited with code=0 in 43.811 seconds
```

2. admin-login.html

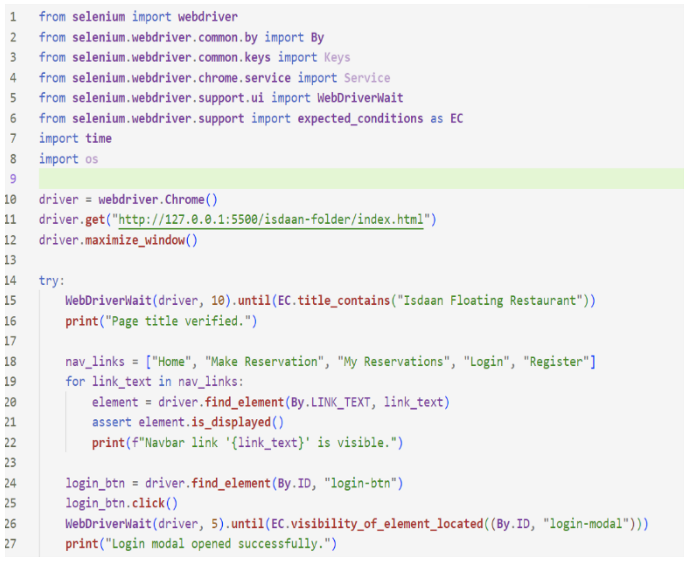


```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

driver = webdriver.Chrome()
driver.get("http://127.0.0.1:5500/isdaan-folder/admin-login.html")
driver.maximize_window()

driver.find_element(By.ID, "admin-username").send_keys("admin")
driver.find_element(By.ID, "admin-password").send_keys("admin123")
driver.find_element(By.ID, "login-btn").click()

try:
    WebDriverWait(driver, 10).until(EC.url_contains("dashboard"))
    page_source = driver.page_source
    if "Dashboard" in page_source or "Welcome" in page_source:
        print("Login Test Passed")
    else:
        print("Login Test Possibly Passed (URL matched but no keyword found)")
except Exception as e:
    print("Login Test Failed")
    print("Error:", str(e))
    print("URL:", driver.current_url)
    print("Page Preview:", driver.page_source[:500])

finally:
    driver.quit()
```

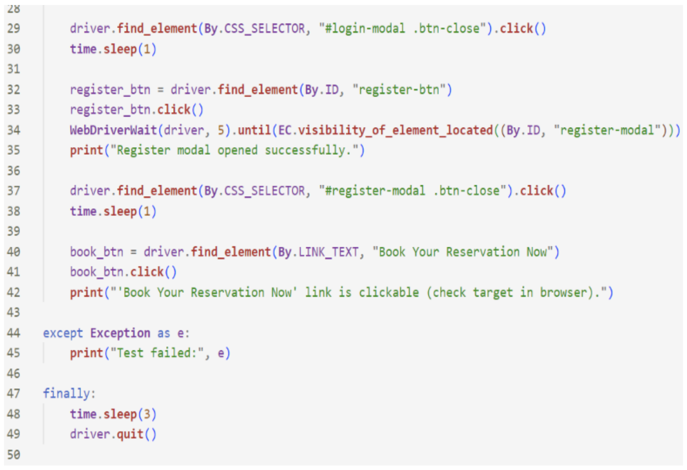


```
[Running] python -u
Login Test Passed
```

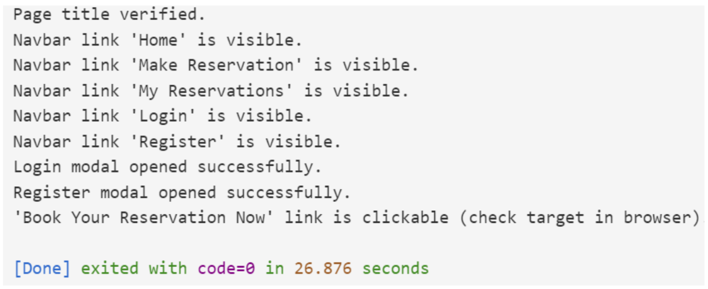
3. index.html



```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time
import os

driver = webdriver.Chrome()
driver.get("http://127.0.0.1:5500/isdaan-folder/index.html")
driver.maximize_window()

try:
    WebDriverWait(driver, 10).until(EC.title_contains("Isdaan Floating Restaurant"))
    print("Page title verified.")

    nav_links = ["Home", "Make Reservation", "My Reservations", "Login", "Register"]
    for link_text in nav_links:
        element = driver.find_element(By.LINK_TEXT, link_text)
        assert element.is_displayed()
        print(f"Navbar link '{link_text}' is visible.")

    login_btn = driver.find_element(By.ID, "login-btn")
    login_btn.click()
    WebDriverWait(driver, 5).until(EC.visibility_of_element_located((By.ID, "login-modal")))
    print("Login modal opened successfully.")
```



```
    driver.find_element(By.CSS_SELECTOR, "#login-modal .btn-close").click()
    time.sleep(1)

    register_btn = driver.find_element(By.ID, "register-btn")
    register_btn.click()
    WebDriverWait(driver, 5).until(EC.visibility_of_element_located((By.ID, "register-modal")))
    print("Register modal opened successfully.")

    driver.find_element(By.CSS_SELECTOR, "#register-modal .btn-close").click()
    time.sleep(1)

    book_btn = driver.find_element(By.LINK_TEXT, "Book Your Reservation Now")
    book_btn.click()
    print("'Book Your Reservation Now' link is clickable (check target in browser).")

except Exception as e:
    print("Test failed:", e)

finally:
    time.sleep(3)
    driver.quit()
```



```
Page title verified.
Navbar link 'Home' is visible.
Navbar link 'Make Reservation' is visible.
Navbar link 'My Reservations' is visible.
Navbar link 'Login' is visible.
Navbar link 'Register' is visible.
Login modal opened successfully.
Register modal opened successfully.
'Book Your Reservation Now' link is clickable (check target in browser)

[Done] exited with code=0 in 26.876 seconds
```

It is important to import all necessary modules and classes per script to run especially the selenium's webdriver module to control the browser. Overall, the results made in each page was successful and that concludes that the system functionalities were operating well because of the efficiency and simplicity of using Selenium as a tool.