CS455: Introduction to Software Engineering

Report, Homework-3: Multi-tiered Architecture

Team Information

This assignment is presented by:

1. Aditi Khandelia, Roll No. 220061
2. Kushagra Srivastava, Roll No. 220573

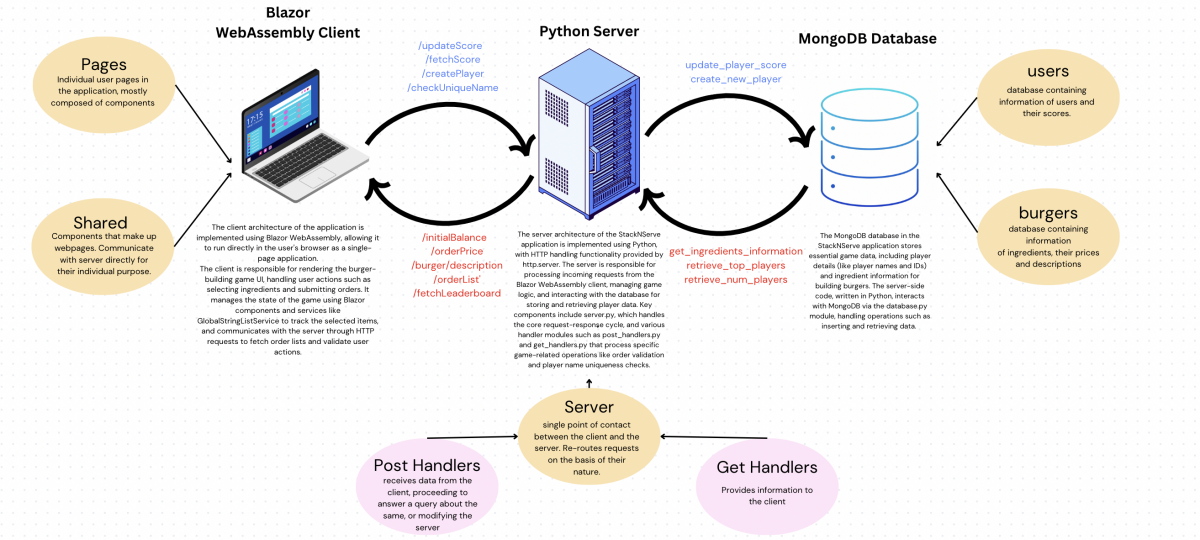
Codebase

The codebase for the application can be found at : `https://github.com/CS455-Assignment-1/StackNServe`

Deployment Information

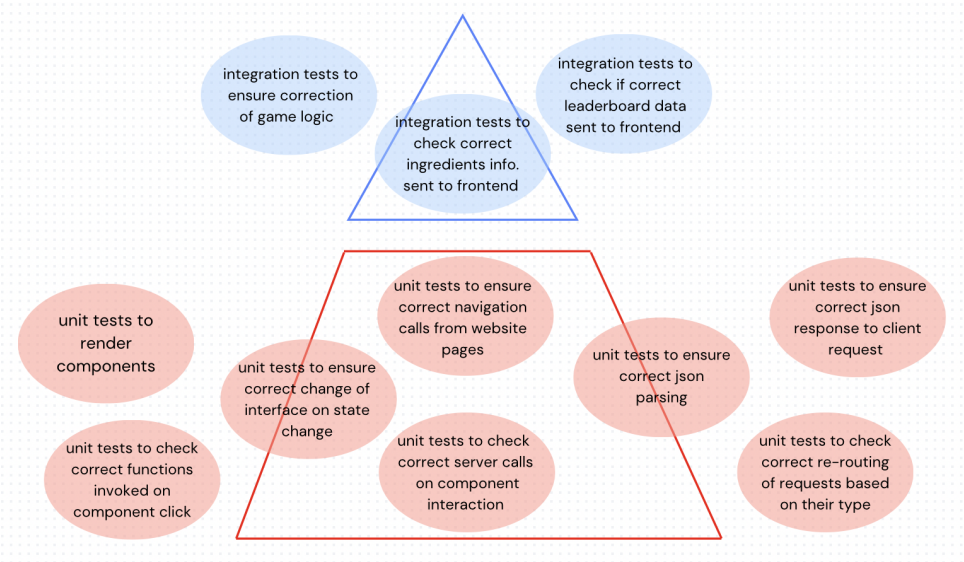
The server for the game is hosted at `https://stacknserve.onrender.com` using Render.
The frontend for the game is hosted at `https://cs455-assignment-1.github.io/StackNServe` using GitHub Pages.

Architecture Diagram



The Diagram may be accessed at : `https://www.canva.com/design/DAGNPAtcjYQ/WI_57OYJ3qqM8Mc39acChw/edit`

Test Pyramid



The Diagram may be accessed at : `https://www.canva.com/design/DAGNPAtcjYQ/WI_57OYJ3qqM8Mc39acChw/edit`

Assignment Information

1. Convert the Static Web Game to a client-server application:
   (a) A python server was made for the project, using http.server package.
   (b) HTTP requests are made by the Blazor WebAssembly client using HttpClient.
   (c) Information such as ingredients controls and user details are stored on MongoDB database, connected to the server using pymongo. Earlier, "models"(ingredients and user classes) were present in the client to handle this information.
   (d) The client now renders the frontend, and contacts the server whenever some information is required on the basis of data in the database, or regarding game controls. The server accordingly fetches the data, performs operations on it, and sends it to the client.
2. Present scores and leaderboard.
   (a) A "user" database is made in MongoDB, that stores the Player Name, Player ID and Player Score.
   (b) After the timer runs out, the players are redirected to a new page where their score is displayed, and the option is given to play a new game
   (c) The leaderboard feteches the top 10 players and their scores, and displays it.
3. Keep a clean architecture.
   (a) There is a single point of interaction between the server and the client, server.py
   (b) There is a single point of interaction between the server and the database, database.py
   (c) server.py redirects requests appropriately to get_handlers.py and post_handlers.py
   (d) database.py creates database and collection objects from the data of the database, and exports them
4. Testing and deployment
   (a) unit tests for client can be found in tests/client.Tests. The three folders Pages.Tests, Shared.Tests and Services.Tests contain tests for the respective folders in client.src. Unit tests are written using Xunit(testing framework), Bunit(for testing of components) and Moq(for mocking).
   (b) unit tests for python can be found in tests/server.Tests. The tests are written using pytest, with pytest_httpserver being used to mock client requests.
   (c) Xunit, Bunit and pytest used to write integration tests between client and server, and between server and database respectively.
   (d) Server is deployed using render, which gets server data from the main project repository itself.
   (e) Frontend is deployed using github-pages.
   (f) The project workflow runs unit tests and integration tests, and passing of which allows the game to be deployed.
5. Turn in an architecture and test pyramid diagram
   (a) Attached above in the report.
   (b) Can be accessed in the project repository as well, inside /diagrams