Smart Office

Group 12

Integration Testing

Integration Testing

The integration testing basically involved linking all the separate modules: the server, the database, the unity client(robots), the user application, together.

It was a huge task but was easily accomplished since it was done 1 step at a time, as follows:

Server and Database

The first thing that was connected was the server and the database. Since the server was the only module that would have access to the database this would be the easiest to connect as there was only 2 things that needed to be tested.

We used a MySQL connecter driver to establish connection to the server and tested it. Since the server was a console application, testing it with simple SQL queries was pretty straightforward. All we had to do was execute the query, display in console and check for correctness in the results.

This step took the least amount of work.

Server and Client

Now since the server and database were connected to each other, our next step was to connect our client application to the server and request for data to see if it was successfully delivered.

The data was sent as JSON packets and the server and clients decoded/deserialized them based on specific server codes as pre-determined by us. The packets were received, processed based on the request type received, a response was generated and sent back to the sender.

This step involved extensive networking knowledge and organization skills to make sure that the data packets were sent and received successfully between the client and the server and there was no data leak or loss anywhere.

Since sending data over network is cumbersome, this step took the most amount of time and was built upon slowly, i.e as more features in the software, the communication channel between the server and client grew and was tested as needed.

Testing in this case involved sending requests to the server via the client application, making sure a response was received, and then finally making sure that the received response was also correct.

Some issues that we faced while doing this part was that sometimes the server or client would throw a serialization exception where they would be unable to decode the received data based on the request type. This was mainly a JSON issue where it does not parse the string back to an object properly if the class of the object is not set to public. This was fairly simple and was resolved quickly

Server and Unity Client

Although it seems like an arduous task, it was fairly simple as the Unity Client was also a client like the desktop application.

The communication channel remained the exact same since the same kind of data was being sent.

The only thing that we had to change in case of the Unity Client was how the data was processed. For example, in case of a coffee request, the unity client would direct an idle robot to deliver coffee to the employee who requested for it. However, in case of the desktop application, it would merely display the response sent by the server.

This task was fairly easy and took a few hours to accomplish at most.

Some issues we faced while accomplishing this part of the integration test were that the unity client would sometimes disconnect automatically after sending a response back to the server. It was later found that that the unity client was trying to send data in an incorrect format and therefore the server would immediately disconnect the client to prevent any fatal errors. It was fixed by making sure all communication was being done in proper JSON format

Closing Remarks

Since all our modules were fairly self-sufficient, it was easy to connect them one by one which made the overall job easier. We did not have to know the fine details of the working of everything to know how to connect them. Since the connections were mainly **network based**, it was just a matter of making sure the network communication was successful. After that the individual modules took over and processed the received data as per their desires.

Although simple it still took us a lot of time to accomplish it since sending data over a network can be tricky as we need to make sure there is no data loss or leak while it is being sent. Also it was our first time having an application that communicated over a server so there was also a learning curve involved.