Tyler Humphries

Southern New Hampshire University

CS-320 Software Test Automation & QA

April 15th, 2023

Project Two – Reflection

In the software requirements for the contact object, the contact ID, first name string, last name string, and phone string could not be null and had a required length of no more than ten characters. The address string could not be null and had a require length of no more than thirty characters.

In the software requirements for the task object, the task ID could not be null, had a required length of no more than ten characters, and could not be updatable. The name string field could not be null and had a required length of no more than twenty characters. The description string could not be null and had a required length of no more than fifty characters.

In the software requirements for the appointment object, the appointment ID could not be null, had a required length of no more than ten characters, and could not be updatable. The appointment date field could not be in the past and had a specific Java utility that needed to be used. The description string could not be null and had a required length of no more than fifty characters.

To ensure these requirements were met in the software, I conducted Junit tests to verify lengths, null, and date requirements. Upon conducting the tests, I found that they were successful, which showed me the program was correctly developed according to the requirements. Additionally, I conducted tests that ensured fields that were not supposed to be updated could not be updated by the user.

To ensure the project was technically sound, I utilized correct syntax when conducting the Junit tests. Specifically, in lines 8, 14, 22, and 32 in TaskServiceTest, I created a new instance of the TaskService and then assigned the required variables to then test that the objects meet all of the project requirements. The same thing is done on lines 11, 18, 27, 38 in ContactServiceTest. In AppointmentTest, lines 21 and 30 create a new date to compare with given the project requirements.

As I completed the various stages of the project from past weeks in the course, I certainly learned a lot about software testing. I was previously employed as a Junior Game Developer for a small startup mobile application development studio called Raincrow Studios, to which I did conduct some software testing and quality assurance for the game we’d designed. However, all I really did was play the game repeatedly, hoping to find bugs through gameplay instead of through testing the code itself. While this method can work sometimes, it’s not nearly as accurate as testing the code by performing tests such as Junit tests. Looking back, I know what I would have done differently to test the software, and that is something I have learned in this course.

Currently, I am employed as a freelance developer for a video game server. In this video game, FiveM, which is a modded version of Grand Theft Auto 5’s online multiplayer, I can sometimes be bias when it comes to testing the additional modifications (or resources, as it’s known in FiveM) for the server. I have a few testers working for me, and sometimes when they locate potential bugs, instead of find a better way to do it, I find myself letting my own personal pride come in the way and responding with “that’s just how the script works.” Instead, I could have a better, less bias outlook and reformat the script so it’s more user-friendly.

Whether it’s developing a game, software, or server, it’s important not to cut corners and not to be “lazy” to provide the best quality work you possibly can. Developers digitally sign their name to their work. Cutting corners or programming lazily can add a bad image to their name, and it’s overall morally unjust. In my opinion, if someone is paying a developer to make them something to suit their needs, it’s the developer’s sole responsibility to produce their best quality work and ensure every need the customer presents is satisfied, assuming it’s possible to do so.