The software testing techniques that were employed during milestone one are basic class implementation (Appointment.java). Introducing JUnit testing, this verified that appointments are correctly created following the requirements (ID length, valid date, description size). In part of the code, **assertThrows** is the assertion used to confirm exceptions are thrown for invalid inputs. Next, in milestone two, this outline is Appointment Service (AppointmentService.java). Utilizing Junit, this milestone verified that appointments could be added and deleted successfully. I incorporated **asserEquals** and **asserNull** to check deletion and retrieval correctness, also adding **assertThrows** to confirm the correct exception is raised. In this specific milestone. Included in the code, you can see the manual testing can play a role in making sure that execution for the program without unit tests to verify proper functionality. For the last milestone (three), this outlined debugging & refinements (AppointmentTest.java, AppointmentServiceTest.java). Starting with regression testing, re-running all previous tests to ensure new changes didn’t ruin existing functionality. During re-testing, I was running in common barriers with simple signs such as **; , ()** or **{}**. Behind missing these simple structural roles, running code was, at first, stressful and identifying where to add and when to add can fix syntax errors. Just as equally important, **functional testing** should be incorporated every 10% step of the way through coding. Meeting business requirements means understanding what they expect to come out from the code. For instance, weight loss consumers who use Weight Loss Applications track their workouts and will always rely on data to by sync across their devices such as apple watches or tablets. Syncing the correct information across everyday use platforms. What I can improve is not only simple syntax errors but more understanding about system testing. **System testing** verifies that the entire system meets requirements as a whole and early detection of vulnerabilities while ensuring software is protected against potential threats. For instance, a healthcare software verification program that has patience data encryption while complying with HIPAA. Referring back to understanding customer code requirements, covering end-to-end testing of full code run.

My future steps will always include reading and understanding requirements first, then constructing with a clear structure, coding one section at a time, looking out for syntax errors, & making it a habit of early test.