**Module 2 Notes – Testing**

* There are generally 5 broad categories that most software development work falls under:
  + **Requirements** – During this phase, a dev team must determine what the proposed piece of software must do. Interviews are held with *stakeholders* to determine needs/expectations from the software. *User stories* and *use cases* are created here. A list of functional requirements will be conducted by the end of this phase.
  + **Design** – After the requirements have been solidified, the design phase is where devs layout the architecture of the program. The team will take the requirements and determine the components that will be needed, such as a database or RESTful API. Later in this phase, devs will map out the classes they will need and what each must do. The end of this phase should result in lengthy design documents that will be referenced during the coding.
  + **Implementation** – Most of the coding happens here and this is the phase most people associated *software development* with. In here, Devs do much of the programming and attempt to match their design ideas with the requirements mapped out in the beginning.
  + **Verification** – The primary focus is to ensure the actual implementation satisfies the requirements and design. This phase is generally to find the mistakes introduced during the coding/implementation process. For example, it may be discovered that the implementation phase successfully met realized the design, but not the requirements.
  + **Maintenance** – This is mostly after the software has been released. The objective is to fix any bugs/errors and add new features that clients/users are requesting. Non-functional requirements can be addressed here as well, such as UI re-designs or decreasing execution times (i.e. make the software faster).
* 362 focuses on ***testing*** which is mainly found in the ***Verification*** category
* **Software testing** can be boiled down to trying to generate a fail state in the software with the ultimate goal of being unsuccessful in doing so (i.e. prove the software has no errors/bugs)
* There are generally 3 main things we’re looking for in software testing:
  + **Failure –** a deviation from the expected behavior (Failures occur because there exists a “bug” in the code: a ***fault***)
  + **Fault –** An instance of incorrect code that can lead to a failure (Faults are usually introduced to the program when a programmer makes a mistake: an ***error***)
  + **Error –** A mistake that introduces a fault (i.e. a typo or conceptual misunderstanding)
* Testing helps us in more ways than one. They can help us/devs in the following ways:
  + **Saves time in the long run –** Testing can ensure the program works how it should, or close to it, after spending hours and hours designing and coding.
  + **Fewer bug fixes after a new release –** Less work/maintenance after the software goes live
  + **Makes you look like a rock star –** Solid testing can help avoid any time sinks, which increases productivity and elevates your/dev team’s esteem
* Testing benefits for others include:
  + **Saves money –** Saves money for individual devs, therefore saving money for the company
  + **Protects teammates mistakes –** Many testing methods are designed to prevent errors from being introduced into the program’s code/codebase. These errors could bring a released program to fail, but at the very least, take time to track down
  + **Happier customers/clients –** After eager customers have been awaiting your new game or app, everyone will be happy if it works as intended at launch. This can boost customer confidence and repeat business, which a failed release/buggy release may do the opposite
* **User reasons** for testing include:
  + **Security –** Many users put a great trust in sharing personal info with software companies, testing can alleviate their concerns and help identify security risks
  + **Quality of life –** Technology has replaced the more traditional way of doing many tasks and if this tech fails, people could suffer life altering consequences (i.e. mobile banking, doctor’s appointments, etc.).
  + **Life and death –** There are numerous examples where software bugs have resulted in the death of individuals, such as the Boeing 737 Max crashes or bugs in medical machinery, such as those that give off radiation.
* There are essentially two categories of testing:
  + **Functional Testing**
  + **Non-functional Testing**
* Both testing types have their own approaches:
  + **Functional Testing:**
    - Unit Testing
    - Integration Testing
    - Regression Testing
  + **Non-functional Testing:**
    - Performance Testing
    - Scalability Testing
    - Usability Testing
  + **Functional Testing** will be the main focus of this course
* **Testing Approaches –** Up until now, ***manual testing*** was generally used
  + **Pros of Manual Testing:**
    - **It is intuitive –** Even beginner programmers know to verify their programs by running them
    - **No upfront cost –** No time investment required before you can start testing (i.e. no need to set-up a testing environment/variables)
  + **Cons of Manual Testing:** 
    - **Time consuming –** Takes a long time to verify a large piece of software by hand
    - **We are only human –** People can make mistakes, even in testing
    - **Not easily repeatable –** There is no way to quickly re-run all the tests after making small changes
  + We will be using ***automated testing***
  + **Pros of Automated Testing:** 
    - **Easy to repeat –** once the tests are written, they can be performed at the push of a button
    - **Fewer mistakes –** Computers are guaranteed to run the tests the same way each time
    - **Very efficient –** Computers can simulate tasks at a much higher rate than humans
  + **Cons of Automated Testing:** 
    - **High upfront cost –** Takes time to write the tests and set up the needed tools
    - **Not suited for everything –** UI testing, for example, can benefit more from the human eye/interaction than an automated test
    - **Test maintenance –** Testing suites need to be regularly updated and added to as the software is expanded
* **Unit Testing –** when individual units of the software are tested (or when the smallest component of a software system is verified to produce the expected behavior)
  + A ***unit*** will typically be a single method that takes a few inputs and has a single output
  + Python has a ***unittest*** library to write automated tests