**Software Lifecycle**

Software Development Lifecycle Intro

* The typical software development lifecycle details how a piece of software is built. It covers inception, creation, and maintenance. This model is very high-level, meaning it covers only the large areas of the cycle. Each area can be further expanded to more specifically define the process.

Why use Software Models?

* Bad code could produce **technical debt** which is *poor and hard to understand, hard to maintain implementation that will have to be repaid with interest later on*. Example - An hour saved now, could cost 20+ hours later
* No documents and comments on why this implementation is there
* Annually $85 Billion Cost of Bad code
* On 41hrs of avg developer work, 13.5hrs is technical debt and 3.8hrs for bad code
* Design is very important

Typical Software Development Cycle

* **Requirements**
  + **What** are we building?
  + No one will be able to contend what we are going to build
  + Define the problem at hand and take the idea of the problem and represent it into more concrete and real world explanations
  + For instance an idea of a website and what are the list of things that this website should be able to do
* **Design**
  + **How** are we going to build?
    - What technologies to use?
    - How are the servers going to be setup
    - How is the front end going to be setup
  + Documentations
  + Front end and Back end design Diagrams and Wireframes
  + After detailing the problem we work on how to solve the problem. Plans on how we create a product which satisfies the requirements
* **Implementation**
  + **Build** the thing. **Build** the product
  + Begin creating the solution according from the design
* **Verification**
  + **Testing** is asking the question, Is what we built is what we wanted to build?
  + In testing, you need to know the requirements. You need to know what it supposed to do
  + We test the product, making sure the solution actually solves the problem. Look for bugs and adjust any areas which are incorrect
* **Maintenance**
  + End cycle where you are **fixing bugs, adding new features, updating some features and adding enhancements, cycling to other parts of the Software Dev phase.**
  + Just **keeping it running**
  + Launching the product and from this point continue on maintaining the features and solving bugs
  + This part of the cycle can go on indefinitely

Software Example - Building a Form

* **Requirements**
  + Collect email address and message
  + Send to and store in a database
  + Prevent user from bad input
* **Design**
  + Use HTML and CSS for building the framework of the form
  + Use Javascript for verification of the input
  + Use JQuery and MySQL for contacting backend
* **Implementation**
  + Code and document the work
  + Setting up the server and the frontend-backend
* **Verification**
  + Does the form collect information
  + Does the form send that information to a database?
  + Does the form prevent bad user input?
* **Maintenance**
  + Create lifecycle plan, fix any bugs
  + Lifecycle plan can be
    - check on it every six months to make sure that all the technology used is most up to date.
    - And same with the database updated every six months.
    - Check any bugs, maybe update that every six months, or maybe you have a weekly schedule of on bugs