- Principles of software Engineering Summary/Questions
- Intro to Ux/Product Management summary/Questions
- Other

- course topics, class demos and course labs cover 90%-95% industry javascript, html, css software engineering development/product management, fundamentals, tools and workflows.

Summary Workflow and Examples:

- Project development ie: website development workflow
- Software Engineering
    - Tools, vs.code, browser devtools
    - Command terminal
    - User stories, fixes, other requirements
    - Team development
        - Github
    - Code review/quality tools
    - Code best practices
        - Object Oriented programming
        -
    - Code diagrams - wireframe
    - Database / table diagrams - UML

- Project management
    - Project management tools - Jira, Other
    - Quality Assurance teams - verify/test use cases and other requirements
    - Daily, weekly meetings, scrums
    - Product management tools
        - Measure user stories, requirements, fixes
        - Measure milestones
        - Measure software quality
        - Measure software time,cost,quality
            - Measurement tools and diagrams
                - Ghant, Prat, Other
    - System Development Life cycle
        - Requirements
        - Specifications
            - Data flow
            - Database relationship - UML
            - In-memory code diagrams - State diagram
            - Code best practices
                - Re-use code in new projects
                - Update code without modifying code in production s

            - Other patterns and best practices
        - Design
            - ui/ux design
            - Prototyping
        - Implementation
            - Reusable code, libraries
        - Verification
        - Testing and Maintenance
    - Team development
        - user stories and other requirements
        - Agile tools, Jira
        - Team meetings - daily, weekly
            - Discuss user stories and other requirements
                - Storyboards - group of requirements
                - Epic - more detailed user stories
            - Agile - incremental updates, dail, weekly, monthly
            - Scrum - daily weekly meetings
    - Software development user stories and requirement measurement tools
        - (requirements, specifications, user interaction, testing, customer feedback, prototype, client/customer/user, developer, risks/errors, quality)
        - Sawtooth model
        - V-model
        - Spiral model

- Product management
    - Product manager
    - Customer engagement
- Customers accessing website
- Development operations team - DevOps
    - Deploy application to different servers / Websites
    - Development server / Website
    - Quality Assurance server / Websites
    - Production servers / Websites
    - Monitor website

Software development terms summary
==

UML stands for Unified Modeling Language, which is a standardized visual modeling language used to specify, visualize, construct, and document the artifacts of software systems. It helps in representing complex system designs through various types of diagrams.

Object-oriented programming is a programming (OOP) paradigm based on the concept of objects, which can contain data and code: data in the form of fields, and code in the form of procedures. In OOP, computer programs are designed by making them out of objects that interact with one another

Polymorphism

https://www.w3schools.com/java/java_polymorphism.asp#:~:text=Polymorphism means "many forms"%2C,out.

Java Polymorphism
Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.

Like we specified in the previous chapter; Inheritance lets us inherit attributes and methods from another class. Polymorphism uses those methods to perform different tasks. This allows us to perform a single action in different ways.

For example, think of a superclass called Animal that has a method called animalSound(). Subclasses of Animals could be Pigs, Cats, Dogs, Birds - And they also have their own implementation of an animal sound (the pig oinks, and the cat meows, etc.):

Product management is the business process of planning, developing, launching, and managing a product or service. It includes the entire lifecycle of a product, from ideation to development to go to market

Agile methodology is a project management framework that breaks projects into phases, called sprints, and uses an iterative approach. The goal is to deliver working software quickly and respond to change

Scrum is a project management framework that helps teams work together to achieve a common goal. It's an agile methodology that's often used in software development.

Sprints - specify the timeline ie: Daily, weekly, Monthly, user stories, features, fixes, enhancements, requirements updates. Sprint is used to measure software updates and releases. Product managers, and Quality Assurance and Development managers manage sprints. Jira tool is used to manage and display sprint timeline graphically.

Dynamic website development - Dynamically updating parts of the web page without refreshing the entire webpage. The entire webpage is not reloaded. Only some of the webpage is updated.
Technologies used to create dynamic websites. Javascript DOM, Single Page Application (SPA) development ie: Ract, Angular, Vue, Other.
Some websites can be dynamic ie: 80%-90% dynamic - 10-20% static content, 30% dynamic, 70% static etc..
Examples of dynamic web pages include:
User interaction on webpage, mouse click, on page load update screen, display date/time, e-commerce websites, Responsive websites, navigation menu, Large number of webpages, large number of custom website, ie: shopify store, shopping cart, animation, dynamically displaying ads on a webpage from an api call, other

Javascript DOM - Document object model -

programming interface that allows you to interact with and manipulate the elements of an HTML or XML document.
Think of it as a tree-like representation of the document, where each element, attribute, and text content is a node in the tree.
This structure allows you to access and modify these nodes using JavaScript

SPA - React -

It's a collection of pre-written JavaScript code that you can use to build your own applications.

User interfaces:
It's specifically designed for building the visual part of your application, the part that the user interacts with.
Component-based:

React encourages you to break down your UI into small, reusable components. This makes your code easier to manage and maintain.
Declarative:

You tell React what you want your UI to look like, and it takes care of updating the DOM (Document Object Model) efficiently.

JavaScript (often abbreviated as JS) is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. It is used to make web pages dynamic and interactive.

https://www.w3schools.com/js/default.asp

Here's a basic overview of JavaScript:
Purpose:
JavaScript allows you to implement complex features on web pages, such as:
Displaying timely content updates
Creating interactive maps
Animating 2D/3D graphics
Building scrolling video jukeboxes
Handling user interactions (e.g., button clicks, form submissions)
Making asynchronous requests to servers
Syntax:

JavaScript's syntax is similar to C and Java, making it relatively easy to learn for those familiar with these languages.
Features:

Object-oriented: JavaScript supports object-oriented programming (OOP) with object prototypes and classes.

Functional: JavaScript also supports functional programming, treating functions as first-class objects.
Dynamic: JavaScript is a dynamically typed language, meaning the type of a variable is determined at runtime.

How it works:
JavaScript code is executed by the browser's JavaScript engine, which is built into every modern web browser.
Example:
Here's a simple JavaScript code snippet that displays an alert message:
JavaScript

```
alert("Hello, World!");
```
Where to use it: JavaScript is primarily used for client-side web development, but it can also be used in other environments, such as:
Node.js: A JavaScript runtime environment for building server-side applications
React Native: A framework for building native mobile apps using JavaScript
Electron: A framework for building cross-platform desktop apps using JavaScript

REST API, or Representational State Transfer Application Programming Interface, is a type of API that uses HTTP requests to allow applications to communicate with each other. REST APIs are often used in web and mobile app development.
How does REST API work?
A client sends a request to a server

The server responds to the request
The client and server can only interact in this way
What are the principles of REST API?

Uniform interface: REST APIs have a consistent interface
Stateless: REST APIs are stateless, meaning they don't store information about previous requests

Cacheable: REST APIs can be cached
Client-server: REST APIs are client-server based, meaning the client initiates all interactions
Layered system: REST APIs are layered systems
What are some examples of REST APIs?

OpenWeatherMap is a REST API that provides weather data based on a user's location
What are some types of APIs?

Open API: A public API that anyone can access
Partner API: A restricted API for business partners and clients
Private API: An internal API used within a company
Composite API: An API that combines data and services to streamline tasks

Node.js is an open-source JavaScript runtime environment that allows developers to run JavaScript code outside of a web browser. It's used for server-side programming, and is often used to build back-end services.

https://www.w3schools.com/nodejs/

Features of Node.js
Cross-platform
Node.js can run on many operating systems, including macOS, Linux, and Windows

Lightweight
Node.js is efficient and scalable, and can handle thousands of concurrent connections
Non-blocking
Node.js uses asynchronous I/O primitives to prevent JavaScript code from blocking

Event-driven
Node.js is built on event-driven programming, which allows it to handle requests without blocking the thread

How Node.js works

Node.js runs the V8 JavaScript engine, which is also used in Google Chrome
Node.js apps run in a single process, without creating a new thread for each request
Node.js uses an event loop to handle requests, allowing it to move on to the next task while waiting for a response

Uses of Node.js
Building web applications
Building back-end services, such as APIs
Powering client applications, such as web apps and mobile apps
Who uses Node.js?
Many big businesses, including Amazon, Netflix, eBay, Reddit, and PayPal

Assignment Questions
references and article and video of examples

user store

Object Oriented Design Principles: Encapsulation, Abstraction, Inheritance, Polymorphism

object oriented programming

Polymorphism - many
With polymorphism- what is a child class instance vs a parent?
QA Engineers

The root cause of a malfunction of a system

Encapsulation

Gantt

State Diagrams

also course chat or email summary of questions

QA - quality assurance - test the software
gantt chart summary

```
poject1
 feature-a - timeline
 feature-b - timeline
 feature-c - timeline
 feature-4 - timeline

//////////////////////////////////////////
gantt chart
//////////////////////////////////////////
1    2    3    4    5
 feature-a - timeline
     feature-b - timeline
   feature-c - timeline
         feature-4 - timeline
```

### in-Class practice exercises
### in-Class Demo project
 - user stories
 - wireframe
 - project website layout and pages

#user story demo
<pre>
User stories are brief, simple descriptions of a feature or functionality from the perspective of the end user.

Who - What - Why
User, request, why
As a [type of user], I want [an action] so that [a benefit].

Examples:

User Registration:
As a new user, I want to be able to register with my email and password so that I can access my personal account.

Shopping Cart:
As a customer, I want to add items to my shopping cart so that I can purchase multiple items at once.

Personal website user stories example

As a User, I would like to display a list of hobbies information on the home page.
As a User, I would like to display summary information about the website on the about page.
As a User, I would like to enter contact information such as name, email and comments on the contact us page.
As a user, Please provide a navigation menu to allow users to goto different webpages, for example, home page, contact us page, and about pages.

</pre>

#wireframe ie: mockup demo
- Diagram of general features of each website page . ie: mockup
- text base wireframe

- drawing tool wireframe

index page
----

home | about | Contact us

[image]

item 1
item 2
item 3

about
----

description ....

[image]

Example - Website - User Stories, fixes, Issues, Features, revisions,Requirements
===

Website - pages, user stories, fixes, requirements, revisions ie: index, page1, etc
- Frontend, backend, api - middleware
- html, cs, javascript, Bootstrap, Other
- React, Nodejs, SQL Other

# Use stories, fixes, Features, etc, are assigned a unique number
# User stories, fixes, Features, etc, are added to jira project management tools
# User stories, fixes, Features, etc, are added to to pages in comments
# User stories, fixes, Features, etc, are assigned to 1 or more developers
# User stories, fixes, Features, etc, history of modifications are added to github - called commit history
# Different types of User stories, fixes, Features, etc,
- Parallel User stories, fixes, Features, etc,
- Independent User stories, fixes, Features, etc,
- Dependent or nested User stories, fixes, Features, etc, ie:
    - Feature a depends on feature b
        - Nested features in feature a
    - Feature b

# Daily, weekly Scrum meeting discuss Use stories, fixes, Features, etc
# QA team verify test , User stories, fixes, Features, etc,
# UI/UX design some website screens, user stores, other
#Tools and diagrams are used to manage user stories, fixes, feature timelines
#Team managers manage user stories, fixes, feature timelines

Page1 , Css features example:
- Feature#1 - create page layout
- feature#2 - add style - center page
- feature#3 - add style -center page content
- feature#4 - remove unordered list style

# Different types of User stories, fixes, Features, etc,
- Parallel User stories, fixes, Features, etc,
- Independent User stories, fixes, Features, etc,
- Dependent or nested User stories, fixes, Features, etc, ie:
    - Feature a depends on feature b
        - Nested features in feature a
    - Feature b
- Other