Most of the studying I did was over databases as well as Go since those are the things that I'm weakest in

https://go.dev/doc/tutorial/getting-started

- Go has a clear hierarchy compared to Python
- Static language
- Has embedding, an automated form of composition
- Interfaces are a class of types and provide a limited form of structural typing
- The definition of an interface type lists required methods by name and type
- The Go language has built-in facilities, as well as library support, for writing concurrent programs
- NOT aimed for parallel processing

Dependency
https://go.dev/doc/modules/managing-dependencies

To track and manage the dependencies you add, you begin by putting your code in its own module. This creates a go.mod file at the root of your source tree. Dependencies you add will be listed in that file.

When you've found a package you want to use in your code, locate the package path at the top of the page and click the Copy path button to copy the path to your clipboard. In your own code, paste the path into an import statement, as in the following example:

import "rsc.io/quote"

Elastic Beanstalk:
● AWS managed services that automatically manages application deployment infrastructure
● Handles: Configuration, deployment, etc.
● No extra charge, just pay for resources used
● https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/go-tutorial.html

Database Options:
MySQL:
● Most commonly use SQL DBMS
● Purely relational databases

MongoDB:
- Used to save unstructured data in JSON format
- Non-relational

Go Databases:

https://go.dev/doc/tutorial/database-access

● To use SQL databases in Go apps you need to to both:
○ Import “database/sql” for interface and objects for interacting with the DB in code
○ Import DBMS specific driver, implements JDBC/ODBC
Go WebApps:
● “net/http” package is used to build web applications
● “html/template” library can process static html templates
● https://pkg.go.dev/html/template
● Can create REST APIs with Gin framework

Dependency Management:
● Go manages dependencies using go.mod files.
● Go checks for dependency differences between import statements and the mod file
● Can set custom versions or proxies for packages between multiple people in an organization

Django:
● Most popular Python web framework
● Supports build in auth and account management features
● Can be complicated to use, essentially have to learn an entire framework that kinda happens to use Python
● Supports front end development in the same framework, most apps organized through a model view controller

Python MySQL:
● Install and import mysql.connector driver
● Can use mysql.connector.connect(user, host, db), etc.
● Can also set a db config dictionary
● To use with Django need to update settings.py DATABASES

3rd Party Auth Services:
Amazon Cognito:

Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps. Your users can sign in directly with a user name and password, or through a third party such as Facebook, Amazon, Google or Apple.

https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html

● Manages user sign-up, sign-in, and an access control for websites
● User pools create sign-in and sign-up, social media sign in, a built in UI, MFA, and user directory management
● Can be managed/used through AWS SDK
● After successful user pool authentication, the app receives user token
● Uses OAuth2 authentication flows in applications

Microservices:
● Create rest API for separate services to communicate with each other and with front end
● Typically used with containers to separate services and scale individually
● Use standard templating/formatting to communicate between services, typically done with something like JSON