Introduction: The Web Application that I decided to create was one that I have mulled over for the duration of the semester. In one of my other classes (CS 340) we have used UML diagrams to model the Classes that we were building for that class. As I did this, I realized that I would rarely use UML in a real life situation, not because I don't see the value it it, but because it takes a significant amount of time. Although I have since changed that point of view, gaining a much stronger testimony of pre planning projects, the idea on which this project was based stuck in my head, and I have been excited to take a stab at it.

Here is the idea: I want to build a UML diagram builder (somewhat similar to Lucidchart) that will be able to not only map out programming ideas graphically, but would also be able to export the idea to real code. This web application would be able to take objects stored in a database that represent classes, interfaces, methods, and how they relate to one another and export them to any object oriented language.

Unfortunately: I, unfortunately, did not have enough time to build out a full scale drag and drop UML diagram builder in the process of working on this project. I, instead build a proof of concept that builds the classes and interfaces with forms.

Use Cases: I work as an Android developer, and have used some tools that do something similar to this product, however, they are strongly tied to Eclipse and Android and are unable to export Object Oriented objects (only relational Objects). They have, however been amazingly useful in saving time when building the Model layer and saving objects off to a database. This web app will one day build both iOS and Android models from its database, synchronization code, as well as a node.js server side web application (After this project, I may also look into having a Django option for the server side code). By using it to model your data, it will save developers hundreds of man hours creating the basics of a fully synchronized application environment.

Database Schema: I will include a schema of the database as a pdf. It shows a basic representation of a the relationship between classes, interfaces, methods, members, and more.

Conclusion: This project was a wonderful opportunity to do some research into some jQuery libraries that I have wanted to look into for some time, including the jQuery Mobile library, and the splitview library. It was also very informative using Django and learning how it works.

Deviations: I bit off more than I could chew. This project was much bigger than what was required. Mainly because I did something that I really wanted to do, not just a project that would fit the requirements. I was forced to cut things out from the UI that I would have liked to keep, including packages, more modifiers for functions, more restrictions on modifiers (abstract methods can only exist in abstract classes, etc...). I would also have loved to actually implement the UML drag and drop interface.

Additions: In the future, I will add the ability to export the data structure to java, objective-c, python, node.js, and more.