# Essay on Object-Oriented Information Systems

## Introduction

From my point of view that I can attest as I have been going through the units in this course is that objects oriented programming is a process. A process that does not necessarily begins at just the programming part of any system but from a piece of paper or a drawing to model the object, relationships between the objects, and also the interaction between the objects. Development of any information system always but not necessarily begins with modeling the database using a modeling language. For this case, I have learned a lot of unified modeling language(UML) which then I used to model the structures of the information system database objects. What I came to realize is that UML diagrams can ease the implementation of the actual product intended since it acts as a blueprint and you can use these diagrams to model the relationships, interactions of various objects, states, and variables on the real product. The software architecture process is always complex and a means to effectively communicate these complexities make the software understandable and hence minimizing the headaches involved with this process.

## Database Design

When modeling a database, I learned there are various types of modeling a database and all of these types are used to model certain aspects or aspects of the database under development. It was also clear to me that there are just two domains in which we could characterize a model. These include the diagrams that model the structure of the database and the other ones are concerned with the behavioral modeling of the database. For the case of the database, I used a combination of both the structural and behavioral models to structure the objects of the database. Just mention a few of the modeling diagram I used to describe the database includes;

1) The famous class diagrams- which I used to model the different objects that went into the final design of the e-commerce database, and also showed the various attributes of the objects, their operations, and their relationships.

2) Activity diagrams- then I proceeded to draw an activity diagram which I sued to showcase the flow of activities with the system. The diagram proved how important it the implementation of the database since I could tell which processes follow or attached to a particular process and it was easy to follow and implement as a result.

The above-mentioned diagrams are the ones I interacted with during the initial database design process. In the other parts of the development process, I also interacted with more which I will mention in the later parts of this essay when relevant. I was able to use these blueprints to implement the database successfully which was with much pleasure since during the implementation process, I was never thinking about what elements or objects were supposed to go to the database instead I was thinking about how to implement the modeled classes and make them work with accordance to the various relationship modeled in the diagrams.

After learning the various aspects of modeling language through implementing a class diagram and activity diagram, the database designing was beginning to take a shape. I learned designing a database involved a little bit of critical thinking on the requirements going into the database writing these logical requirements down on paper without even thinking about the physical database the data will reside. It involves laying down the various components without getting the hands dirty on the process of implementation and this is important since it gave me a chance to think over the requirements at this stage and how the requirements will interact and interface with each other on the physical database once implemented. The deigning process was able to give me a bird's view of the database but again a very realistic one since nothing changes during implementation if the design is thought through thoroughly.

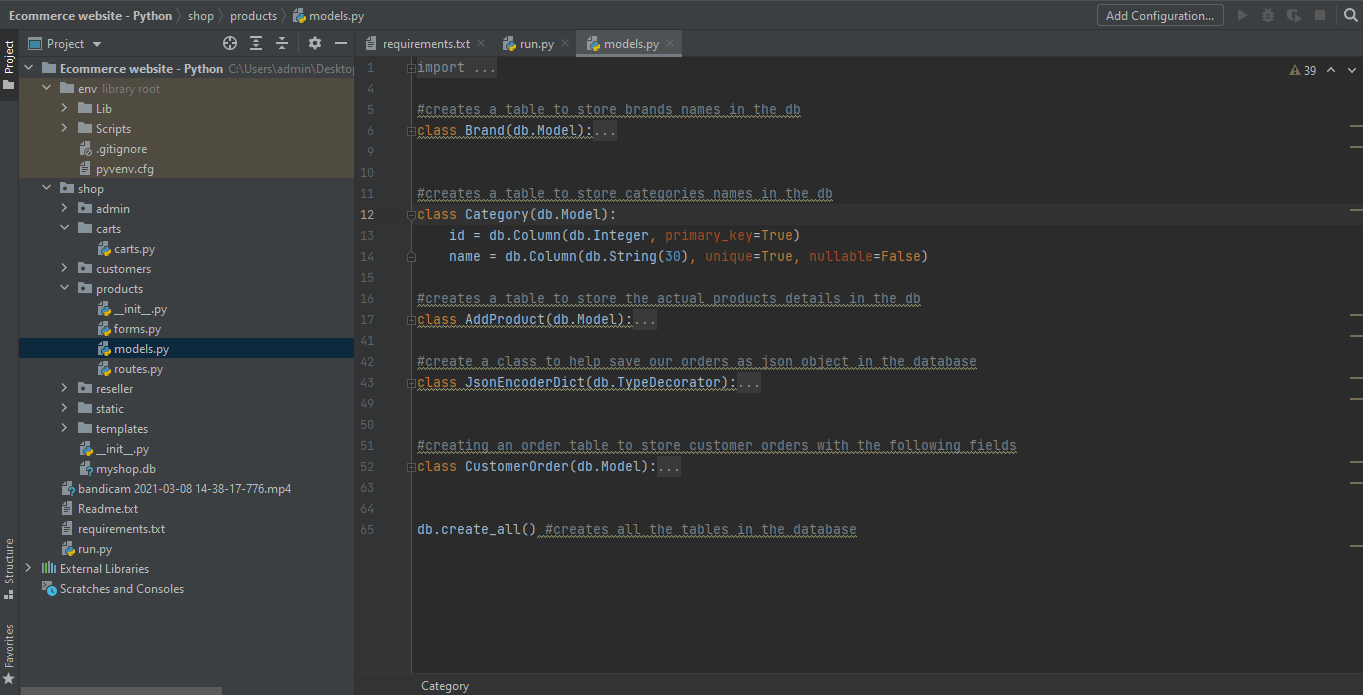
During the database design of the online store database, I was also able to learn and experience database designing techniques which are commonly used to design databases in the industry and I felt like it was a giant leap forward in my career goals to have understood and experienced the two techniques. These techniques are normalization and entity relation modeling. Normalization entailed organizing my database tables in such a way that it reduces redundancy and dependencies. Now, this technique is important to make sure that I did not repeat myself in the database tables declarations. It also gave me the freedom to be able to subdivide bigger tables into smaller ones and then link them using relationships as a result I was able to minimize redundancy on the design.

Entity-relationship modeling was all about modeling the various entities of the database in which every entity represented a real-life object like for example a product and user. Some entities relate to other entities which for example a user account which for the case of an e-commerce database may have customers and resellers. The entity relationships models use to show these relationships between an account a user whether a customer account or a reseller account. Doing this on paper was gave me an idea of how the database will look like and how I will be able to join different tables depending on the relationships modeled on the entity-relationship diagram.

With designing the database out of the way, the next step is going ahead to implement the database. Now from the above sections of this essay shows how far the process of developing a database can take someone through. Just to summarize the steps as I learned through experience, the first step is to identify the requirement, this requires identifying what goes into the database. The second stage is to design the database using the techniques given above and the last one is to implement and test the database. Now it seems like we are on the third stage of the development process.

## Implementing Database

Implementing the database for me mostly meant bringing the models we drawn during designing into “life”. To do this I needed a way to communicate with the host database through a query language, for our case on the online store we learned and used SQL for the implementation. The biggest advantage I found with SQL is that it is minimalistic and quite easier to learn. The way I implemented the database was that first I initialized an empty database, from the entity-relationship diagram, took every entity, and gave it a table on the database. So every entity represents a table where a table is a collection of related attributes. In the designing phase also, I had identified the primary keys and foreign keys, so I was able to map the attributes with their respective keys. It was quicker and intuitive implementing the database with the already existing knowledge and can minimize the development time. This is a process that I can never overlook once a database is needed at any point in my career because the benefits are enormous not also forgetting it minimize the likelihood of me committing an error during implementation.

In my personal opinion, working with SQL directly is good to create a database but it increases the complexity when working with big databases that have hundreds of relationships. It can get messy and confusing sometimes. The way that I worked with the SQL on the e-commerce store database is using a Pythonic object-relational mapper called SQLAlchemy which is a query building infrastructure. It was easier working with SQLAlchemy since I could turn python classes or objects into database tables at the same time working on the same Pythonic environment which was a little advantageous because I didn’t have to explicitly write queries to the database hence reducing time spent and the complexities involved when developing a database using SQL directly.

The above figure shows the implementation of the database using Pythonic classes collectively known as models. For example, the class Category is a model of a table in the database to be created with attributes id and name respectively. For this case, we declare id to be the primary key on the table. The other classes in this model kind of work similarly, I define the class then within the class I define attributes with their relationship and what type of data the attribute will store. What I learned from the documentation of sqlalchemy is that the models will be converted to SQL in the backroad so it's like working with SQL indirectly and lets me develop the database in the same Pythonic environment as the other codes. This means I don’t have to write SQL on the python codes or separate SQL files, working like this is intuitive and easier to follow as well. The best part about this is that I could map the class UML diagrams into objects classes on the SQL models together with their attributes, this is handy since you know I could only think of how to make this work with my frontend.

## Web Development in Python

Having learned to apply UML modeling on the database, I replicated this to the design and implementation of the frontend e-commerce application. With the front end, I explored some more UML diagrams to help model behavioral and structural aspects of the application. A part class diagram and activity that I used for the frontend modeling, I also was interested to model the states of various entities. Entities or objects behave differently not only on inputs but also depends on the state the entity was previous so I found it useful to kind of think of this and try to anticipate the behavior of objects with different inputs at a certain state. The result of this was a state machine diagram that modeled this behavior and this impacted the development of the final product because it led to more understanding of the implementation features together with their behavioral transitions. The other diagram that I thought was important was an interaction overview design to model the flow of control for various actions as the user performs them. This is important because it helps during the development to allocate resources to nodes that have more control over many different actions. This was like drawing the nervous system of the application, and it clearly showed how the objects defined earlier interact and interface with each other at the system level. This was very handy to have during the implementation of the system because with the interactions clearly shown on paper, my job was only to transfer that to the system implementation and one particular thing I noticed during the process is that I was able to group objects which are critical to the system and the ones that were non-critical and by doing this, it enabled the ability to understand and start with the more critical process while the less one coming last.

This endeavor has main been fruitful because it led me down a path to be not only an applications developer who not only develops applications starting from coding which I used to do before but who plans and design the aspects of the application before it goes down to implementation while enjoying the benefit of reduced implementation time and complexity. For me, the development process of any information system means first identifying the database requirements that the software application needs to accomplish, then second design the implementation of the requirements on paper using a modeling language like UML and the reason why to use UML is that it's widely used by developer universally and once you give a UML to any developer they will know what goes where on the final implementation of any software, lastly when the designing process is done its time to implement the database using the modeling diagrams designed.

For the frontend design, I also used some more tools like wireframes. Designing the UML diagram is a good thing to have but I noticed that since the classes are real objects the users interact with on the front end, they do not address where, or how the objects will be positioned on the front end interface. To do this task, a wireframe is used to design the overview of the frontend implementation details. Doing this just like designing using UML alleviates the complexities involves and I only have to spend my time on things that matter on the application rather than how it will look and feel, this is already done using the wireframe, hence reducing the amount of time required to develop the real application. I find it much convenient now to set up all the requirements of the application before the real development because of the very reasons discussed in this essay and maybe many that I won't be able to cover. With wireframes, I was able to name all the functionalities required in the database and all the screens design that the frontend will make use of. The nest thing was to map the wireframe design with real implementation using the python programming language. The e-commerce website was implemented using a python framework called a flask. The advantage of using flask is that it is fairly easy to learn with some knowledge of python and also a lightweight framework that does not overload the application with unnecessary codes to call for other services that may be the application do not require hence increasing the bundle size of the application. The other reason why python is such a nice language is that it allows for object-oriented programming which for this case was necessary to implement the modeled classes defined previously using UML modeling language.

## Conclusion

Learning the development processes in these units has given my life as a developer a fresh start since then I have been able to apply this knowledge to a couple of websites that I have been developing to sharpen my skill. I successfully applied this during a flight booking system using python using the same process but of course different classes and attributes. It turned out great and I was able to even take less time on the development and even modeling with the UML since I already had the framework used during the development of the e-commerce website. In comparison with how I used to do it in the beginning with how am doing it right now, this process makes all the difference for me. It has changed my perspective from software development being hard to be more positive which in my opinion, without the process of developing an application can be a steep mountain to climb. The other hidden advantage which I quite find fascinating as a developer with using the above-named process led me to be able to subdivide my tasks in every development stage and I knew exactly what I was supposed to accomplish for every task and that’s the key to faster and efficient software development. Having to experience the different processes hands-on has greatly lifted my spirit and I feel more confident in the products that I develop and that’s what a developer needs to settle into this vast field of software development.