CSC300 Midterm

Part One UML and Code

(i) I chose to sketch up a couple classes, with only one class having inheritance. The Business Class contains all the logic for maintaining vendors and employees. The employee class has a base class but also two subclasses which include a salary worker and intern as subclasses. With any business that sells products there are outside vendors required for providing supplies to a company. Also since the employee class is not coupled to the business class directly in all circumstances the vendor class could implement the employee class and subclasses with only needing to modify the vendor class.
(ii) You can see that there is composition between the employee class and the business class. This is intended because without the business class the employee class would not exist. And the business class would not exists properly without the employees.
(iii) In my diagram I specified that the aggregation was between the vendor and the business. This is because the business could potentially not have vendors and the vendors may not work with a business, so there relationship is not as solid as the employee and business relationship.
(iv) Class Inheritance is displayed in the diagram with the two subclasses inheriting from employee. This method of inheritance lays a base structure that any employee at the business could have such as name, title, and department. Since the base class is very general the subclasses can provide their own additional functionality that may not be need in every class. Take for instance an Intern and Salary Worker, an intern does not a paid worker but a salary worker is and may have benefits while the intern may only be receiving credit towards their university.
(v) The use case shows a more in depth explanation for how the interaction between the classes would be implemented and what actions would need to be handled. The end result is a customer receiving a product. Each class has its own responsibility such as the employee class which handles day to day operations and the company which handles large scale planning and interaction with the employees and vendors.

Option Section

Question:

In the context of Requirements Elicitation and Functional Modeling:
(i) Describe a technique used to write requirements (8)
(ii) Once the requirements are gathered describe a technique to find use-cases. (5)
(iii) What do you think should be the order of steps when writing a use-case?

Answer:

When it comes to planning for a software product eliciting information for a solution requires several approaches to ensure you are able to achieve the system requirements. Some possible techniques for gathering requirements would be through the clients/stakeholder, customer needs, scenarios under which the system will operate, and how the system will be used, guidelines from industry standards such as being PCI-compliant are a few examples. The main technique I use and has worked well in the past is working directly with the client in determining what they need a product to be able to do. If you are able to talk in non-technical jargon and relay to the client the possible issues within their product or solutions to how they want their product implemented you can start creating requirements. It is best to use past experiences in gauging possible requirements when discussing the product with a client so you can try and cover all aspects of the product in the beginning instead of allowing for scope creep later on in the development process.

Once all the known requirements have been gathered a great technique to determine use cases would be to collaborate with the entire software team, client, and possible end users of the product. In this approach all the possible scenarios to be handles could be displayed with technical approaches and views from non-technical types. Brainstorming sessions and white board collaboration are great ways to draw out the possible use cases that need to be handled. Understanding the products requirements will provide a decent amount of general use cases but it takes more in depth thinking within a group of diverse people to figure out ass possible use cases. Using software visualization products is also a great approach to be able to visualize the process and identify cases.

My opinion for the steps in determining the use case would keep the current model of: identifying actors, identifying goals, defining pre-conditions and post-conditions, describing the main main flow, expectations and alternative flows. But, before step one I would start with collecting resources such as documentation on the technology stack to be used, regulations, requirements, and end goal expectations. From their I would have my team go through the materials to become familiar with the product to be developed. Next, depending on the product and if there was a visual design for it I would use a mock-up of the possible page layout to determine interaction and how to handle the user interaction, if no visual aspects were to the product i’d observe similar systems to gain insight on their process. Then I would continue with the traditional steps for writing the use cases but with more informed and knowledgable developers of the possible cases.