SOFE3650 – Software Design and Architecture
Term Project Description

This project is about harnessing the efforts of volunteers and other good will members of our society for the greater benefit of our community. At times volunteers who unselfishly want to benefit others do not find the right means or venue to volunteer their efforts. Other times there are restaurants who wish to giveaway older food to the needy but are unable to deliver them. The end result is a vast amount of good will not harnessed and the ones in need are the ones who pay the price. The aim of this project is to be able to manage and direct volunteers as well as wealth-givers to communicate better and to become more effective. The system attempts to bridge the gap between volunteers and wealth givers so that the ones in need would benefit much better.

Volunteers are individuals who unselfishly choose to help others for free. Volunteers register into the system and state how they can be reached, what they can do and what is their availability to volunteer. Depending on the information volunteers provide, the system will contact them email, cell phone messages or other digital means. Volunteers will also be able to state the type of help they can provide. For example, some volunteers are physically strong so they can help build and move heavy objects. Other volunteers have pickup trucks or cars so they can transport various items. Other volunteers may have other special skills, such as medical training.

Wealth-givers may include any institution or individual who have an item to give, including but not limited to money. Such institutions may include restaurants, supermarkets or manufactories that have extra food, medicine or building materials that they want to give away. Other institutions may be hospitals that have room to treat ill people for free. Other institutions can be clothing stores who have faulty clothing that they want to give away. Wealth-givers register into the system to notify the system of the type of goods they can offer, when they become available and they type of volunteers required…etc.

Beneficiaries are individuals who benefit from the work and wealth of the volunteers and **wealth-givers**, respectively. Beneficiaries register into the system to notify the system if they need anything, the quantity they need, how urgent do they need it.

Note that in general the difference between the **volunteer** and a **wealth-giver** is that a wealth-giver has something of value to offer but cannot connect with a **beneficiary**. Meanwhile, a **volunteer** is the person that helps make that missing connection. The core of this project is to be able to connect the volunteers with the wealth-givers for the benefit of beneficiary. For example, if a restaurant has food to give away, the system will locate a volunteer nearby that restaurant with a car or a pickup truck to take that food and deliver it to people in need. The system will also notify the volunteers with the locations to deliver such food. Another example may be a notification received by a sick person who needs a doctor whereby there is a hospital nearby that can treat the patient for free.

The system is made of two major components, a managerial component and an Artificial Intelligence (A.I.) component that works in the context of a GPS system. Your company has assigned your team to handle the design of the **managerial component ONLY.** Your company has assigned the A.I. component to a **different** department to implement. The managerial component is responsible for handling volunteers, wealth-givers, beneficiaries as well as other entities in the system. The managerial component uses the A.I. component to perform the required logic behind connecting the volunteers with the wealth-givers with the beneficiaries. For example, the A.I. component can be used to determine routes and the locations of near-by wealth-givers, volunteers and beneficiaries. The managerial component also receives feedback from the A.I. component with regard to actual events occurring in the field.

Phase 1: ***Due: October 5, 2016 @ 11:00pm***

(1) Develop a use case model of the system. Your use case model should have at least 5 main use cases (not inclusion or extension use cases). One obvious use case would be “Deliver food” while another can be “Deliver medical help”. It is up to you and your team to come up with the other three. Admin type of use cases cannot be considered as main use cases (in fact they are not use cases period).

(2) Document each use case in terms of summary, dependency, actors, precondition, description, alternatives, and postcondition. A template will be provided.

(3) Develop a physical entity class diagram for the system.

Phase 2: ***Due: November 2, 2016 @ 11:00pm***

(1) Define alternative package structures for the use case model. Discuss the rationale for each alternative package structure and the structure you will finally select.

(2) Develop interaction diagrams (one for each use case), using either collaboration diagrams or sequence diagrams, depicting the sequence of interactions among the objects participating in each use case.

(3) Develop statecharts, depicting the different states of the state dependent objects.

(4) You should consider your design carefully as you need to provide an implementation of interaction diagrams/statecharts representing one main use case.

(5) Consider one nontrivial method/functionality and develop an activity diagram for its underlying algorithm. The activity diagram should include forks and/or join nodes, as well as decision diamonds.

Phase 3: ***Due: December 5, 2016 @ 11:00pm***

(1) For each package, develop a detailed class diagram that depicts the classes grouped into the package and the relationships between them as well as the attributes and operations of each class. (You should be reusing the classes you created in Phase 2).

(2) Improve the overall quality of your design by applying some design patterns, principles of class design, and principles of package coupling and cohesion. Statechart diagrams developed in Phase 3 should have a corresponding statechart design pattern implementation. Also, the observer design pattern is expected to used heavily.

(3) Provide a class interface specification for the major classes in the system. In other words, specify each operation (method) provided by the class in terms of:
- Function preformed (a brief description, one or two sentences)
- Precondition (a condition that must be true when the operation is invoked)
- Postcondition (a condition that must be true at the completion of the operation)
- Invariant (a condition that must be true at all times)
- Input parameters
- Output parameters
- Operations used from other classes

(4) Provide the implementation for the remaining four use cases.

Important Notes:

- State any assumption you make.
- Whenever possible, document alternative design solutions.