CS 441 Software Engineering

Assignment 2

Due on Sunday, February 24, 2019, 11:59PM.

Individual Portion (20 points)
Create a UML Use Case Diagram for a vending machine that sells beverages and snacks. Assume there are two actors: customer and assistance operator. They can perform the following operations:

Customer:

1. Buy product
   1.1. Choose product by inputting code.
   1.2. Make the payment: insert bill or slide credit card.
   1.3. Retrieve product and change. Note that retrieving change requires paying by bill.
2. Call assistance if machine is not working correctly (means if anything is wrong in the above).

Assistance Operator:

1. Machine maintenance: fix machine or refill.

Make use of inclusion, extension, and generalization relationships as needed.

Group Portion (80 points)

Choose one of the applications listed below. Work in a group (no more than three students) and create a requirements document for this application. Your requirements document must

(1) Be written in a clear and professional manner.
(2) Follow the format, structure, and guidelines presented in IEEE Recommended Practice for Software Requirements Specifications.
(3) Include at least FIVE use cases to define some main functions of your application.
(4) Be signed by the project owner (i.e., your customer).

Applications:
Note – the descriptions of these applications are the project owners' original words. Some may not be clear or may have errors, which occurs in real software development. Therefore, you need to contact your project owner, interview him/her, clarify the requirements, and present them in a clear and systematic way in your document.

Application A:

"*Teams will design and implement a 3D simulation of a physical machine, to be used for testing its control code in a safe virtual environment. Users of the software will configure machine parameters, and supply customized control code to drive the virtual machine's motors and simulate movement. The simulator should give users control of simulated time, with controls to start, pause, and restart the simulation. In addition, the rate of time must be configurable, allowing speeds ranging from slow motion up to a max speed at the computer's limits. To observe the simulation, users should have controls to fly the camera around to view the virtual machine as it operates. Ideally, this simulator would be licensed as open source, but allow for commercial use; Teams would be responsible for identifying an appropriate license.*

The machine and its simulated mechanics will be simple: A hefty, weighted block moves when pushed and pulled by X-, Y-, and Z-axis motors, like a 3D Etch-a-Sketch™. The moving block has a configurable mass, and builds momentum as it accelerates. The simulation draws its movement over time. Each motor has independently configurable friction and maximum torque. The block is limited to travelling within a rectangle from the origin at (0, 0, 0) to a configurable width, depth, and height. The travel limits should be drawn on-screen, and if the block moves outside of them, the machine crashes and the simulation halts.

The user's machine-controlling code should plug in to the simulation in some way to control the block's movement motors. One thousand times per second, the simulator calls the user's loop function, passing it updated motor position readings. This loop function drives the machine's block by updating desired motor forces in response. The API for controlling motors and interfacing with the simulation should be well-documented. If time permits, the simulator could allow plotting a configurable set of quantities like block velocity and acceleration over time, to aid the user in debugging their control code."

Project Owner: James Abernathy (Email: TBD).

Application B:

"*Application for Google home or Alexa application.*

The main goal of the application is to communicate with a voice assistant (Google Home, Amazon Alexa) to get more information about something, (monument, place, university, stadium, …)

We can easily adapt this application to the text chatbot like Messenger, WhatsApp, … These chatbot have similarities to voice chatbot but they communicate via text with the program.

It could be a great project to learn more about how voice assistant work, build a big project around this new technology, and to adapt this system to CSUSM.

Have a voice assistant in campus or messenger chatbot to help us to get information about a lot of things, will be a real breakthrough in the digital transformation of campus.

The chatbot can also be connected to CSUSM account and help student to manage their schedule or to propose them events that correspond to their schedule, … and so more application.

We will be able to link the Software Engineering course to campus life.

Before to introducing how to develop voice assistant and all we need to have a great system, let talk about voice assistant, to better understand what is it? And why it could be interesting to this course but also to campus life or other things.

Chatbot is the term we use to define a program that interacts with human either by voice or by text.

We don't create a machine learning program. We use Dialogflow NLU (Natural Language Understanding) or Amazon Alexa NLU to understand the voice. Dialogflow or Alexa gives us "Intent" (according to the sentence of the user) that we had defined, and our program will react according to the intent received.

For example, we train the NLU to it match "How is the weather?" or "give me the weather" to this Intent "TodayWeatherIntent". And we know now that we must answer about the weather when the NLU give us "TodayWeatherIntent". It is the simplest example.

Goals:

Create 3 parts, described below:

➔ *One, the voice application itself*

➔ *Second, platform web call « Data Layer » allows for administrator to manage data, manage database, ...*

➔ *Third, create database, and create api to access to the database*

The first part consists to create an application to communicate by voice with IT System.

For this will use the nodeJs language to code the program. Why nodeJs? Because it's a great back office language and the community is very active, more for creating voice assistant or chatbot with nodeJs.

How works voice assistant? And architecture of voice assistant program.

To better understand, we can look an example.

For example, a person "John" say at the voice assistant a sentence. The program sends this sentence to the NLU of google or amazon (they exist a lot of other NLU, more or less powerful).

In short, the NLU give at our application one or many intents. An intent is for example (WhichCityIntent) that mean the NLU understand, that the user looking for a city.

After that we treat the request, we call our api, databases, … and we can send a response at our device (amazon Alexa, Google Home, ...)

All the intelligence artificial, for understand the sentences are done by the Natural Language Processing that made by Google or Amazon. But the rest treatment is done by our application.

How can we modify the program, to become a text chatbot?

For example, when we connect our voice assistant with the google technology (DialogFlow) this technology proposes us to adapt our technology to text chatbot.

In our program system, for our application to be compatible with voice and text, we have to determine which device the program is used (google home, amazon Alexa, smartphone, …) From there we can just adapt some response, to have a better user experience.

We have, of course, some modification because it is non-visual design, but vocal design between text chatbot and voice chatbot (voice assistant).

The second part is to create a web platform, called "Data Layer". Why we need "Data Layer" platform.

I call this web platform "Data Layer" because it is a platform between voice assistant program and database. A layer that the administrator can use to make a curation of data.

Let's explain. For example, we want to know where is the University Student Union (USU). If a student asks at our application, where is the USU, and in our database we just have the word "University Student Union" the program will don't understand the request. This problem is commons at all software in touch with machine learning. This example is simple, but we can have a lot of more complex situations.

So, with the Data Layer, we can add synonyms to the data or other thing to handle the data. We can also add data, remove or change data from this platform.

But with data Layer we can make our system even more generic. For example, we can add window to choose which Api, which database we will take the data? And so on, …

Like this, we can just change easily the database and our system can work with other university or other organism that want the same system.

So, this platform is very important to have a better user experience with our application or be more modulable.

The third part is more to create a database and therefore several Api to communicate with it.

We will create services in the voice application to add, change, remove, get, some information in the database. But more, we will also need Api to communicate from the web platform for database.

There are no more things to tell about this part. It is an important part to communicate between our voice assistant application and the Data Layer and database.

With this application, we can create 2 systems that can be useful in the real life. These systems ask a great architecture to be viable and to be used by other universities, if it is an application to help student in university. With a great architecture, we can easily adapt our software for other situation other solution.”

Project Owner: Antoine Boudet (Email: TBD).

Application C:

“*The software application that I would like a group of students in our class to build is a daily task-organization application. This application would be a personalized daily calendar that could be shared and updated among a group of people.*

The application would display the layout of one day on the device's screen, with the day split into 48 blocks (each block represents a 30-minute time frame). The basic layout is that there will be four rows with twelve blocks per row, and an additional row on top to display the date, extra comments, and extra To-do items (see Example A below).

A user can add a colored bar to a specific block when that user completes a task, and the application will be updated among the shared users so that everyone can know when a task has been completed. When a bar is added to a block, an exact time (hour and minute) will be displayed inside of the bar. When adding a bar, the user can choose to have the current time displayed on the bar or to manually enter a time. Manual entry of the time can be useful for when a user for any reason decides to add the bar at a later time than the task was completed. Users can also add comments to a bar, or even add a comment for the day in general. When a block is edited, a colored bar will appear on that block, and a user can tap it to view the details of that bar. For additional personalization and organization, a group of users can set different categories of tasks and/or comments to show up as differently-colored bars.

Furthermore, users can choose receive various notifications, such as: when a user completes a task, when a certain category of a bar is added or edited, when any user makes any edit to a block, and when a set task has not been completed within a set amount of time (for example, a reminder that it has been over 12 hours since the dog ate breakfast).

To be more effective, the application would allow the user to add a widget to their device's home screen to be able to view the day's layout and updates without manually entering the application.”

Project Owner: Mhealyssah Bustria (Email: TBD).

Application D:

“*I'll start with a quick explanation of the field that we would be applying computer science to in this particular project: stock trading. As you may or may not know, numerous companies are financed primarily by publicly trading shares of their ownership amongst the public in various stock exchanges. Purchasers buy shares of a company if they believe that the company will* *flourish over time and that those shares will be worth more money later on. Therefore, investors profit by choosing the right stock options, waiting until the price increases, and then selling those shares at a better price.*

For my group project idea, I propose that a group in this class engineers a tool for themselves that is quickly growing in popularity in the stock trading business: a trading bot. This bot would be an active participant in daily market trading on behalf of the users, and it would attempt to make stock trades on a daily basis in order to make a profit for the user, if possible. Investing strategy would be left up to the particular development team.

Unfortunately, this project would be complex and long-term. So, for the purposes of this class, I suggest a more simplified bot project based on the bulleted criteria below, to ensure that the project is not too long for the semester:

● *The group must make a simulated stock market exchange, with several different company stocks available, and whose prices change throughout the course of the program based on price changes that have occurred to the real-life shares throughout the course of a single day or several days in the real-world market. There should be a wide variety of stocks that end at a higher or lower price than they started, and a wide variety of price changes over time until the end of the program.*

● *Design and implement a simplified banking system with several planned cash deposits as well as an amount of starting money.*

● *Design and implement a market trading bot, which will:*

○ *Collect pricing information about all available shares on the market from the exchange simulation.*

○ *Rank all shares according to both current price and previous increases to determine the best possible candidates for investment.*

○ *Based on both available funding and available viable investment options, choose a number of shares from one or more companies to purchase as an investment.*

○ *Take home a profit of any kind at the end of the program.*

● *Display trading performance upon termination of the simulation, including but not limited to: the shares bought and which quantities of each, total profit, days elapsed, etc.*

The developers would most likely use at least 5 separate classes to implement this project, and the resulting program would be useful for possible real-life trading endeavours."

Project Owner: Tyler Gerritsen (Email: TBD).

Application E:

"*I am wanting to design a hotel management software on phones for customers. Right now, there are tons of online hotel booking website but I don't remember any hotel has their own App for the customers to manage their rooms or schedules. The App will require the customer to register an account by using their phone number and Email address. Once login, the customer* *will be separate into two different state, those already checked in are called "In" and those are not called "out". The difference between two states are customers will see a different UI once login; customers are in "out" state will have a option to book any hotel cooperated with the App, you only need to input number of people, date check in and address, the App will automatically suggest the best result for the customers; or you can simply search the nearest hotels that have rooms available.*

For the customers in "In" state once login, they will the option for calling room service, call a cab and manage how many days they want to live in if they want to extend or reduce. Every customer will have a unique QR code that can display from the App, the QR code can be scan at the hotel restaurant, store or any place that required to pay, once scan money will simply charge from the linked credit card from the customer. While a customer is away, he can use the App to call room cleaning service, that way cleaners do not need to ask every day and knock on doors. In my opinion, the "In" state of the app does not require to have many functions because customers can call front desk and do almost exactly the same thing, so the developers can focus more on the "Out" state of the app."

Project Owner: GenHong Lin (Email: TBD).

Application F:

"*The software application idea I have is a website. This website is a ride sharing website which differs from ordinary rideshare services like Uber or Lyft. This website's purpose will allow users around the world to rideshare with each other. The defining difference between Uber or Lyft is, Uber/Lyft does not fulfill the ability to plan ridesharing ahead of time. Uber, Lyft, and other rideshare companies offer on the spot ridesharing. Sometimes, however, that is not what is desired by the user. People may wish to plan their trips out ahead of time for a variety of reasons including following an itinerary or cost saving. Their only option currently is to book a taxi which can be a costly investment and defeats the purpose of planning. My rideshare website will accomplish and fulfill the needs of this target market (planned rideshare). The tools that I am planning to use to build this website include: HTML, CSS, JavaScript, jQuery, Bootstrap, PHP, MYSQL, Google Maps API, AJAX, and JSON. I may not utilize all these tools, but I can imagine seeing a need for each of these languages. The website will work by prompting a user to sign up or log in. If user needs to sign up, upon successful signup the user will receive a push email to active his or her account. A user should also be able to change his email address or reset his password if required. Once this is done, the user should be able to put a start and end destination into the Google Maps API. When this action is completed, there will be a called to the MYSQL database to query all drivers willing to drive from the start destination to the end destination. This query will display drivers name, vehicle, cost per seat, departure time, phone number, etc. The user will also be allowed to add a trip, edit a trip, or delete a trip if the user himself/herself wants to offer a ride sharing service from a destination to an end destination. Additionally, a user should have the ability to update his profile information with pictures, details, etc. As more and more users sign up, I hope to cover the entire area of San Diego!*"

Project Owner: Parth Kapur (Email: TBD).

Application G:

"*Create a mobile application that can be used as a social platform by both the students and instructor of a class in a university. Different from traditional course management systems (e.g., Cougar Course), this application is primarily to facilitate the communications among the students and communications between the instructor and students.*

The instructor can create a class in the system, add students into the system (e.g., by sending invitation links or generating a QR code), review student profiles (e.g., pictures, hobbies), send a message to a student or the whole class.

This application may include special programs (e.g., checking attendance, creating a poll or survey) to support in-class activities.

A student can create a profile and share the information (e.g., pictures) about himself or herself. A student can join a class, add another student in the same class as friend, create a group (e.g., to work on a class project), and share notes with other students.

A student can also create an event (e.g., an after-class meeting for midterm exam) and ask other students to participate.

Students can also use the application to discuss assignment and class project.

Overall, this application will allow us to communicate not only in class but also out of class."

Project Owner: Yongjie Zheng (Email: yzheng@csusm.edu).

Note: I got some of these ideas from the applications proposed by Alexis Vasquez, Alyssa Snow, Guadalupe Guerrero, and Seren Yavuz.