CS6310 – Software Architecture & Design
Assignment #4 [125 points]: Grocery Express Project – Architecture & Design (v1)
Spring Term 2023 – Instructor: Mark Moss

Submission

- This assignment must be completed as part of an approved group.
- One member of each team must submit the justifications for the proposed project modifications [35 (functional) + 35 (non-functional) + 30 (your choice) = 100 points] along with your technical deployment plans [25 points], in a file labeled ***group_proposals.pdf***.
- Submit your answers via Canvas.
- You must notify us via a private post on Piazza BEFORE the Due Date if you are encountering difficulty submitting your project. You will not be penalized for situations where Canvas is encountering significant technical problems. However, you must alert us before the Due Date – not well after the fact.

Problem Scenario

The clients are excited about your work so far as you enter the final phase of the project. Given the time you've spent studying the problem space, the clients are now asking you to make three different modifications to the system, where each modification is functional or non-functional:

- Functional modifications affect the way that the domain objects (e.g., stores, drones, etc.) operate, behave, and interact; and,
- Non-functional modifications improve the system overall structure based on one or more architectural perspectives.

The clients will designate one functional and one non-functional modification that you must implement, and you are allowed to select a third distinct modification of either type. You will work with your teammates during this phase to integrate your earlier individual efforts into a single system and plan how you will implement the designated modifications for your team. You should also give serious thought to the new commands and instructions that you'll need to provide in your interface to support these new capabilities.

The possible functional and non-functional (i.e., architectural areas of consideration) modifications are listed below, and we will select both types of modifications. You can view the modifications that were assigned to your team by viewing the A4_A5 Modifications spreadsheet from PostEm in Canvas.

Disclaimer

This scenario has been developed solely for this course. Any similarities or differences between this scenario and any of the programs at Georgia Tech programs are purely coincidental.

Deliverables

This project phase requires you to submit the following items:

Project Modifications [100 points total: 35 points each for the designated functional and non-functional modifications, and 30 points for the modification that you select.]

- Submit your proposed project modifications in a file labeled ***group_proposals.pdf***. The file should contain one approx. six to twelve (~6-12) sentence description for each modification.

- For the non-functional modification, you must use an Architectural Decision Record (ADR) format. ADRs are discussed in more depth in Chapter 19 of Fundamentals of Software Architecture (Ford & Richards, O'Reilly 2020). Also, here are some links that contain more examples of (and details about) ADRs and the process:
  - https://adr.github.io/
  - https://www.ozimmer.ch/practices/2020/05/22/ADDefinitionOfDone.html
  - https://medium.com/olzzio/y-statements-10eb07b5a177
- For the functional modification, provide as similarly clear and concise description of how the modification changes and impacts. This isn't intended to be an extensive writing exercise, but rather an opportunity to ensure that you've considered the potential architectural, design and implementation challenges of the assigned modifications.

Planned Technical Deployment [25 points] Provide a brief description (~4-12 sentences) of the languages, systems, and related technologies that you're planning to use – or at least seriously considering using – to implement and deploy your final system. We don't expect a comprehensive list of every single library or package that might be included, but we do want you to be proactive on letting us know your most likely plan for implementation. We give you a relatively wide latitude with the technologies that you can select for the final project, but you must coordinate your choices with your designated grading TA to ensure that they will be able to interact with and evaluate your final submission thoroughly. The TAs have "final approving authority" for technology selections. Please include your "Planned Technical Deployment" content as the last portion of the ***group_proposals.pdf*** document.

- "Significant" in this case means that there must be some clear and substantial level of effort that is needed to modify the current design and codebase in order to implement your proposed modifications. The primary things that we will look for when evaluating your proposals are:
  - Clear identification of the modifications that you will be making. The best way to do this is to use a basic reference architecture for your comparison, such as a relatively simple solution to the previously implemented designs.
  - Presentation of a relatively strong case of how your modifications will improve the system. For performance-based changes, you can show how the improved system will outperform the basic system - either in theory, or with real data and testing, or both. For other situations, you can present a sequence of actions that highlight how the basic system fails to provide the needed properties, while the improved system overcomes these issues.
  - Finally, presenting a strong case for your proposals doesn't mean that you have to avoid any discussion of trade-offs. It's perfectly acceptable to highlight the strengths of your proposals while also discussing some of the difficulties, to include how improving some of the system properties can also have an adverse impact on others.

- You should also review Heilmeier's Catechism below for some issues and considerations that you can address to strengthen your proposal descriptions. These questions are credited to George Heilmeier and are relevant when evaluating research projects or product development efforts. You can find more details about Heilmeier's Catechism at the following link:

https://en.wikipedia.org/wiki/George_H._Heilmeier

- What are you trying to do? Articulate your objectives using absolutely no jargon.
  Describe the proposed improvement in a clear and concise manner.
- How is it done today, and what are the limits of current practice?
- What's new in your approach and why do you think it will be successful?
  In the context of this project, does the current system provide similar capabilities already? Is your proposal a totally new capability, or an improvement over an existing but limited capability?
- Who cares?
- If you're successful, what difference will it make?
  In the context of this project, which groups of simulation users will actually find this improvement useful, and why/how?
- What are the risks and the payoffs?
- How much will it cost?
- How long will it take?
  You must provide some estimate of the difficulty that will be involved in implementing this proposed improvement in terms of development resources.
- What are the midterm and final "exams" to check for success?
  You must provide the "acceptance criteria" for this improvement in terms of clear and quantifiable metrics that can be used to verify that it has been implemented correctly.

You don't have to specifically address every question above for each proposal, but when used appropriately they can help you make proposals as clear, concise and persuasive as possible.

Architectural Areas of Consideration (for Non-Functional Modifications)

We have identified various architectural terms below. These terms and their descriptions were taken from the textbook "Fundamentals of Software Architecture" by Neal Ford & Mark Richards (O'Reilly Media, 2020). Your task is to determine how you would modify the architecture and design of your system to better address these topics.

Scalability: Ability for the system to perform and operate as the number of users or requests increases.

As the popularity of the system you've developed grows, it might be deployed across a wider – and possibly worldwide – audience. Consider also that not only might the number of users increase, but the users might be distributed across a larger geographical area. How would you modify they system to better support a very large number of users being able to access the system concurrently without corrupting the operation of the system?

Performance: Includes stress testing, peak analysis, analysis of the frequency of functions used, capacity required, and response times. Performance acceptance sometimes requires an exercise of its own, taking months to complete.

If there are hundreds of thousands, or possibly millions, of streaming service viewers using the system on a "round the clock/24 hour" basis, then the data collected in the system will generally increase.

The increase in the size of the data set and the number of transactions will potentially increase amount of time needed to display reports, or to perform checks to ensure that transactions can be executed correctly. How would you modify the system to maintain and/or improve the performance of the system?

Robustness: Ability to handle error and boundary conditions while running if the internet connection goes down or if there's a power outage or hardware failure.

There have not been many error handling requirements so far. This is an opportunity to improve the error handling of the system and much more. Consider the possibility of the program crashing – for example, power being lost – while executing a program run. How would you modify the system to allow your system to resume execution after such an event without major delays, data corruption, etc.?

Configurability: Ability for the end users to easily change aspects of the software's configuration (through usable interfaces).

We implemented a relatively simple Command Line Interface in earlier assignments, so this is an opportunity not only to generate a better, richer interface, but also to address things like the ability to set the starting month and date of the program run instead of hardcoding it into the program. Similarly, the implementation of some of your other architectural and design changes might create other configuration parameters that need to be changed. How would you modify the system to allow users to manage these configuration parameters as easily as possible?

Archivability: Will the data need to be archived or deleted after a period of time? (For example, customer accounts are to be deleted after three months or marked as obsolete and archived to a secondary database for future access.)

Suppose that the data that is being collected and analyzed must be stored for a certain period of time. This might be driven by a legal requirement, a policy requirement, or a technical limitation, but either way the system must be implemented to manage the storage of this information. And, in some cases, there must also be ways to ensure that the information is purged after a certain amount of time.

These next two architectural concepts are "intertwined" in that it's difficult to consider one of them in isolation, because implementing one of them effectively all but requires that you consider the impacts (and dependencies) on the other. I'm listing these two terms here as a "bundle".

Authentication/Authorization

- **Authentication: Security requirements to ensure users are who they say they are.**
- **Authorization: Security requirements to ensure users can access only certain functions within the application (by use case, subsystem, webpage, business rule, field level, etc.).**

Authentication and authorization are often essential if you're going to enforce any kind of security and/or privacy protocols.

Privacy: Ability to hide transactions from internal company employees (encrypted transactions so even DBAs and network architects cannot see them).

Security: Does the data need to be encrypted in the database? Encrypted for network communication between internal systems? What type of authentication needs to be in place for remote user access?

Also, the public has become much more aware of the need for data privacy, and there are various things that can be done to ensure that user's data – especially medically and/or financially sensitive data – is accessed only by authorized users for its intended purposes.

***Auditability: Ability to track the "activity within the system" (e.g., which commands were executed by which users) in such a way that a user's activity records can only be reviewed by personnel in appropriate roles, such as a Security Administrator; and, that a user cannot remove or otherwise corrupt their own activity records.**

**I included this one from other sources including my own personal experience.*

The unifying theme of all of these architectural considerations is making the system more "trustworthy" against the case of deliberate, adversarial threats as well as accidental, non-adversarial issues that might corrupt the data and/or operations of the system. How would you modify the system to improve its overall level of trustworthiness?

"Get these Snakes off of my Drones…" and Other Hazards (for Functional Modifications)

We have identified various functional options below. These options require you to modify the problem domain to develop these new capabilities:

[1] Add entities/classes, attributes, associations between classes, etc. as needed to design and implement your changes.

[2] Also implement new aspects for the problem domain that will allow you to demonstrate the impact of your new capabilities on the main use case. The main use case is allowing customers to request, purchase and/or cancel orders, and delivering those orders using drones in a timely manner. More specifically, you must introduce concepts of distance, time, and cost to be able to better measure how your new capabilities are affecting the original problem domain.

Angry Birds: You must design and implement the concept of one or more angry birds for your system. Each angry bird must exist at some location – for example, at a store or a customer's location - and be allowed to move between different locations in a reasonable manner. When a drone travels from a store to deliver a package to a customer, then an angry bird located at either location has a possibility of harming the drone, thus forcing it to return to the store for repairs and preventing successful delivery of the order. The system user must be able to adjust the number of angry birds, along with the probability that an angry bird will successfully attack a drone at its location (or in a reasonable proximity).

Lightning Storms: You must design and implement the concept of lightning storms for your system. These storms may be system wide or may occur in a certain region of the "system map", and even move across the map much like storms do in real life. The storms may also vary over time – for example, they might be more likely during the late evening and early morning timeframes. When a drone travels from a store to deliver a package to a customer, then a storm in the proximity of the drone has the likelihood of striking the drone with lightning, thus causing the destruction of most of the packages on board, and possibly the complete destruction of the drone itself. If the drone is struck by lightning, then the store must replace the drone, restock the necessary ingredients, and fulfill the order as quickly as possible. The system user must be able to adjust the intensity of the storm, including the likelihood that drones will be struck by lightning, and adjust the general level of damage caused by the strikes. Also, if the storms vary over time, then the system user must be able to adjust the frequency and duration of the storms.

Time-Sensitive Coupons: You must design and implement the concept of time-sensitive coupons for your system. These coupons should be randomly distributed to customers over time and may be distributed more frequently to customers who have a higher rating. The coupons offer reductions on the overall price of an order or on specific items in the order. If a customer has applicable coupons for their order, then the purchased order must be delivered before the coupon's expiration date, and the coupon discounts must be applied to the order cost. If the store is unable to deliver the orders in time, then there should be some penalty (financial?) for the store. The system user must be able to adjust the frequency and distribution of the coupons.

Customers Returning Items: You must design and implement the concept of customer returns for your system. Customers may return items from their previous orders. This opportunity is limited to a certain number of previous orders, or orders purchased within a certain duration. The customer may return the entire order or only portions of the order. The store should maintain enough information to ensure that a customer's return claim is valid. When a return claim is made, the store must dispatch a drone to pick up the items being returned and return the appropriate amount of credits to the customer as quickly as possible. Only empty drones can be used to process returns. If a drone is sending a new order to the customer requesting the return, it can be used if (and only if) it doesn't have any other pending order items on board. The system user must be able to adjust the frequency and distribution of the customer return claims, including whether it will be a full or partial return, and which items will be returned.

Refueling Stations: You must design and implement the concept of refueling stations for the drones for your system. The system must track the distances between stores and customers using fixed location concepts or through some other means. Drones will use some sort of physical liquid or solid fuel for power and consume fuel when delivering orders based on these distances. If drones don't have enough fuel to return the store or another refueling station, then they should stay at their current location until another "refueling" drone can be sent to that location. The store's location is always a default refueling station, and stores can invest in extra refueling stations in other (strategic) locations to facilitate timely deliveries. The system user must be able to adjust the numbers and

placement of refueling stations, along with the fuel rate needed to travel a certain distance. Also, the drones must have a reasonable (and adjustable) maximum fuel capacity.

Solar-Powered Drones: You must design and implement the concept of solar-powered drones for your system. The system must track the distances between stores and customers using fixed location concepts or through some other means. Drones will use solar energy for power and consume fuel when delivering orders based on these distances. If drones don't have enough fuel to move to another valid location, then the drone must stay in a valid location. The beauty of the system, however, is that all drones are refueled "automatically" by the energy of the sun over time. Your system must implement a "day/night-like" clock cycle such that drones receive more power during the bulk of the day than in the early morning and late evening timeframes. The system user must be able to adjust the refueling rate for drones, along with the fuel rate needed to travel a certain distance. Also, the drones must have a reasonable (and adjustable) maximum fuel capacity

All these functional modifications would have significant impacts on the opportunities to deliver orders to customers in a timely manner. Your system must also implement aspects that can be used to better quantify and demonstrate these impacts. The main three aspects here are **distance**, **time**, and **cost**. In this way, your system can assist the users in evaluating different strategies for employing your resources to deliver orders as efficiently and effectively as possible.

Distance: Distances are helpful in demonstrating the impact of drones moving to different locations to deliver items: the longer the distances, the more time and fuel that will be needed to reach the destination. One idea is that locations could be implemented using a relatively straightforward coordinate system, with distances calculated using a basic Euclidean formula.

Time: It takes time to complete many different actions, but especially including order deliveries. Time could be implemented in a logical fashion, by saying that each command consumes a certain "unit" of time. Certain commands might then consume more units based on their specific actions – for example, delivering orders might take a certain number of time units based on the distance between the locations and the speed of the drones, etc. Alternatively, time could be implemented using timestamps to denote when events have occurred. Using this "timestamp-based" system still requires that your system ensure that the timelines "make sense" from a real-world perspective.

Cost: Somewhat jokingly – but seriously as well – "time is money." More specifically, if orders are not delivered in a timely manner, then there will normally be cost impacts for the stores: both for the immediate events, and for future events in terms of dissatisfied customers not placing more orders. Your system already implements costs for the items and orders, but can also implement costs for purchasing drones, refueling stations, replacement order items, penalties for not delivering orders in a timely manner, etc.

Closing Comments & Suggestions

We (the OMSCS 6310 Team) will conduct Office Hours where you will be permitted to ask us questions in order to further clarify the client's intent, etc. Also, the TA who will evaluate your final

submission will be pre-assigned to your team. You can communicate with them if you have questions related to application design, implementation, deployment, etc.

Quick Reminder on Collaborating with Others

Since this is a group project, you may (and should) communicate freely with all of your group members. However, your group is not allowed to communicate with any other groups while working on this project, including outside personnel or consultants. Please use Piazza for your questions and/or comments, and post publicly whenever it is appropriate. If your questions or comments contain information that specifically provides an answer for some part of the assignment, then please make your post private (your group members and OMSCS6310 TAs/Instructors only) first, and we (the OMSCS 6310 Team) will review it and decide if it is suitable to be shared with the larger class. Best of luck on this assignment, and please contact us if you have questions or concerns.