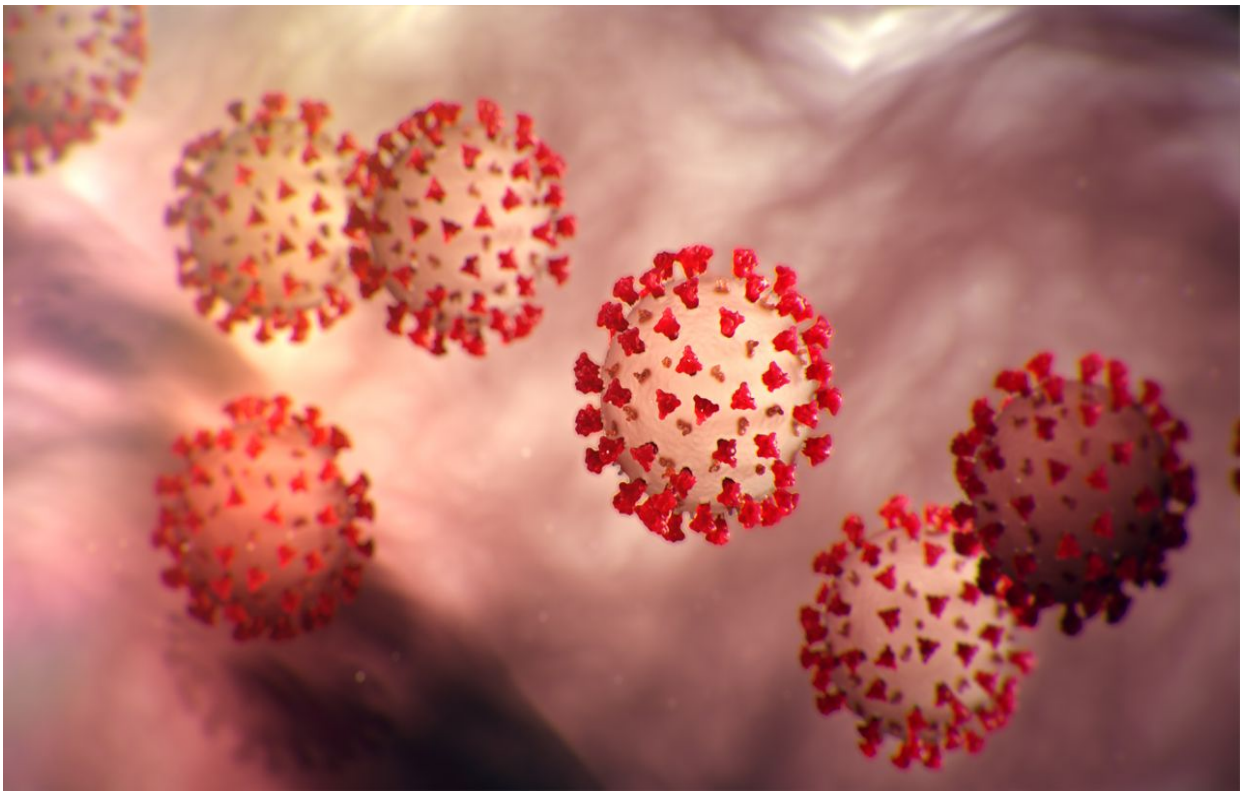
Software Engineering

Report #1

Restaurant Occupancy Prediction (Minority game): Group 3



PROJECT CONTRIBUTION CHART

	Tatsat	Austin	Chris	Eric
Project management	100%			
Problem Statement	50%			50%
Decomp into Subproblems	50%	50%		
Glossary of Terms		40%	60%	

SUBTEAM 1:

	Austin Bae	Christopher Rosenberger	Jin Xu
System Requirements (5 point)	100%		
Functional Requirement Specification (30 points)	100%		
UI Requirements (10 points)	50%	50%	
System Architecture (25 points)		100%	
Project Size Estimation (15 points)		100%	
Plan of Work (5 points)	100%		

SUBTEAM 2:

	Tatsat Vyas	Tanvir Singh	Erik Robles	Nitin Ragavan
Stakeholders		100%		5%
System Requirement	80%			20%
Actors and Goals	100%			
Use Cases *(Casual Descriptions)*	100%			
Use Cases *(Use Case Diagram)*	100%			
Use Cases *(Traceability Matrix)*	100%			
Use Cases *(Fully-Dressed Description)*	100%			
System Sequence Diagrams	100%			
Preliminary Design			100%	
User Effort Estimation			100%	
System Architecture		25%	60%	15%
Project Size Estimation				100%

Plan of Work	80%			20%

Table Of Contents

Problem statement

During these unfortunate times, and the coronavirus pandemic, we are forced to adapt to this new way of living to keep ourselves and everyone around us safe. The technological advances have helped us in many ways by enabling us to work and study from home. Technological business solutions have made running a business easier than ever before, with many groundbreaking software systems such as automated accounting, online ordering and delivery, widespread social media marketing, advanced security systems, cashless checkout, and so much more. Indeed, technology has made the lives of business owners everywhere much easier, more convenient, and made their businesses much more profitable. However, The pandemic has had a major effect on local businesses and restaurants that rely on social interactions. Restaurants are told to follow certain restrictions depending on the COVID cases in a particular area. In some states, restaurants and bars were allowed to open just to shut down again due to the surge in COVID cases. In this time of uncertainty, we restaurant owners need something that we can use to get an idea of what the business will look like.

One of the major things in a restaurant and bar industry is to optimize the daily expenditure that we make in order to make sufficient profit. A daily

expenditure is what we buy from our vendors to sell to a customer. In a restaurant and bar business this daily expenditure is groceries, alcohol, cleaning supplies, gas and water bills, employee wages and much more. If we buy all of these things by taking account the turnout of that particular day then we can minimize the waste and as a result, optimize daily revenue. Normally, this is very obvious and can be predicted by an experienced management team and many years of trial and error. However, with the introduction of many dining and operational restrictions imposed by local governments on small businesses to contain the spread of the virus, previously relied on methods of business management that were commonplace are now seen as obsolete in current times. Even the most experienced management teams will need some external help to manage and calculate the many factors, restrictions, and trends that currently threaten businesses to help predict future turn out. For this reason we, the restaurant owners, have come to seek external help from the technical community.

What we want is an application that runs on our computer. The application will be some sort of simulation that will give us a prediction of the turnout in our area. The application needs to take some parameters that influence how people think during this time. Some of our areas have a lot more cases, so the application needs to account for that. In addition, there are also these restrictions such as the restaurant capacity that are set by individual states the restaurants are located in. Other things that also influence the turnout is the current event. What we mean by the current event is what is happening in the area, is the area a source of attraction. All

this necessary information will be provided to the system by the user(the restaurant or bar owner).

We also want this application to have an aspect of timing. We should be able to run this simulation each day(round) to predict what the turnout will be. The hard part that we think for this might be that our customers are humans so we want this application to have individual agents in it that evaluate the information that we supply and make a decision. The agents should be able to also remember their decisions such as if the decision was a good one or not. We think that these previous decisions will affect the future decisions made by individuals. We would also like for the application to consider normal factors such as whether it's a weekday or weekend because there usually always are discrepancies between them. Holidays and other special events should be taken into consideration as well, and we want to be able to see predictions for months in advance.

This application can really benefit the way we prepare our restaurant for the turn-out. There are many restaurants whose businesses and the lives of those employed have been affected by the pandemic, and this will be a handy tool to give them some sort of perspective in dealing with the situation at hand. With this tool we will be able to fairly predict the future and have a helping hand for our skilled management team. We will be able to save money by preparing for the right amount of people. During these unfortunate times, having this support will be able to help save many small businesses. It will help eliminate the unnecessary waste and optimize the way we run our business with advanced computing solutions.

Decomposition into Sub-problems

The overall problem will be divided into two subproblems. These subproblems will be a fully functional system that can hold on it's own but combined will be a full fledged system that implements an overall simulation of smart agent minority game.

1) Agent Interaction(sub-team 1):
   The focus of our system is agent-to-agent interactions. Our system takes inputs from the user and from sub-team 2's system. We group the agents in the population and based off the inputs from subteam 2, the agents will interact with each other in groups to formulate group decisions
2) Venue and external factors based simulation(sub-team 2):
   The venue and external factors are the face of the system. We will be focusing on taking input from the user and creating the simulation. Our system will create agent strategies, evaluate strategies and make decisions for each agent. There will be a dummy evaluating process that makes up the number of turns out(win or lose). Ideally, this information of win or lose will be obtained from sub-team 1.

Glossary of Terms

Agent: A simulated "person" that plays the game (decides to either stay home or go to a restaurant).

Minority: A user-defined percentage that dictates the number of agents (out of the whole population) that it takes to make a decision a favorable outcome (if more than the minority make the same decision, that decision is an unfavorable outcome). Every agent not in the minority is in the majority.

Success/Favorable Outcome: An agent (not user) determined outcome where the agent observes that the % of other agents that share the same decision is less than than the user-defined % for minority.

Outcome: One of two options: Go out, or Stay Inside.

Strategy: A set of decisions made by the agent based on various factors in order to get the winning outcome. There is a certain “threshold” beyond which each person will decide to stay inside.

Round: A unit of time in which agents manage and choose a strategy, and the results (who and how many chose the minority/majority decision) are calculated.

Memory: An agent’s record of their past chosen strategies and their outcomes. Each of these will be weighted along with the current strategy.

Attribute: A trait of an agent that affects how they or other agents choose a strategy.

Personality: An agent’s set of attributes, can contain 1 or more attributes.

Sub-team 1:
Team members:
Austin Bae, Christopher Rosenberger, Jin Xu

Goals, Requirements, and Analysis

Business Goals

The business goals of this project will be elaborated upon in Subteam-2's section

Enumerated Functional Requirements

REQ-#	PRIORITY Weight 1 = low, 10 = high	Requirement Description
REQ-1	10	Users should be able to set initial condition
REQ-2	10	Agents should form groups
REQ-3	7	Agents exist in different 'states' based on whether or not they have or had the virus and for how long
REQ-4	7	Agents that are symptomatic do not go out
REQ-5	5	Groups should be able to remember their decision and outcomes for the past 10 rounds (short term memory)
REQ-6	9	Groups have varying 'personalities'
REQ-7	3	The program should keep track of how many people attended each restaurant on each day
REQ-8	5	Groups can create their own strategies
REQ-9	1	Program graphically outputs relevant data associated with the simulation

Enumerated Non-Functional Requirements

REQ-#	PRIORITY Weight 1 = low, 10 = high	Requirement Description
REQ-10	6	The application should be able to handle at least 1000 rounds
REQ-11	7	The application should be able to handle at least 1000 agents
REQ-12	5	Users should be able to download the initial conditions and results for the last 3 games and compare all of them
REQ-13	5	The application should work on various operating systems
REQ-14	1	The application should save user inputs

User Interface Requirements

REQ-#	PRIORITY Weight 1 = low, 10 = high	Requirement Description
REQ-15	10	Must have labeled text boxes for each input parameter
REQ-16	5	Each parameter box must have a descriptor button next to it that explains what the parameter is/how to determine the number
REQ-17	10	Must display the final result of the simulation
REQ-18	10	Must have a clear, distinctive run simulation button
REQ-19	1	Relevant input parameters should be appropriately and neatly grouped

Use Cases

Stakeholders

There are many types of venues and establishments that would be interested in this software. Not only is it great for restaurants and bar owners who are trying to predict outcomes and interpret trends, but it can also be used by anyone that deals with many people and limited space. This could be for airports, gyms with limited machines, hospitals with limited beds, any several other types of establishments. In times like these, it is extremely important to be able to know how many people will arrive and how to prepare for them, This software, with a bit of tweaking, can very accurately help out managers, owners, and data analysts to predict, organize, and prepare for many different events and situations.

Actors and Goals

Initiator	Initiator's Goal	Use Case Name
User	Set the initial conditions for the simulation	ConfigureGame (UC-1)
User	Compute agent turnout given the initial conditions	RunSimulation (UC-2), ComputeGroupDecison (UC-3)
User	Obtain information about expected turnout	DisplayResults (UC-5)
User	Obtain and display previous simulations' initial conditions and results	DisplayThePast (UC-6)
Subteam-2 System	Get decision of each individual agent in the population	InjectData (UC-4)

Use Cases

a) Casual Description

i) UC-1 ConfigureGame: This is where the user establishes initial conditions for the simulation, namely: Simulation Length, Agent Population, and Restaurant Capacity. The data entry is done by the user and will be the first page the user sees after entering the application.

ii) UC-2 RunSimulation: This starts the simulation of group decision making, where agents are grouped and given personalities, and the agent groups communicate amongst themselves and with each other in order to make a decision. The simulation will also store the initial conditions along with the agent turnout for the latest 3 simulations.

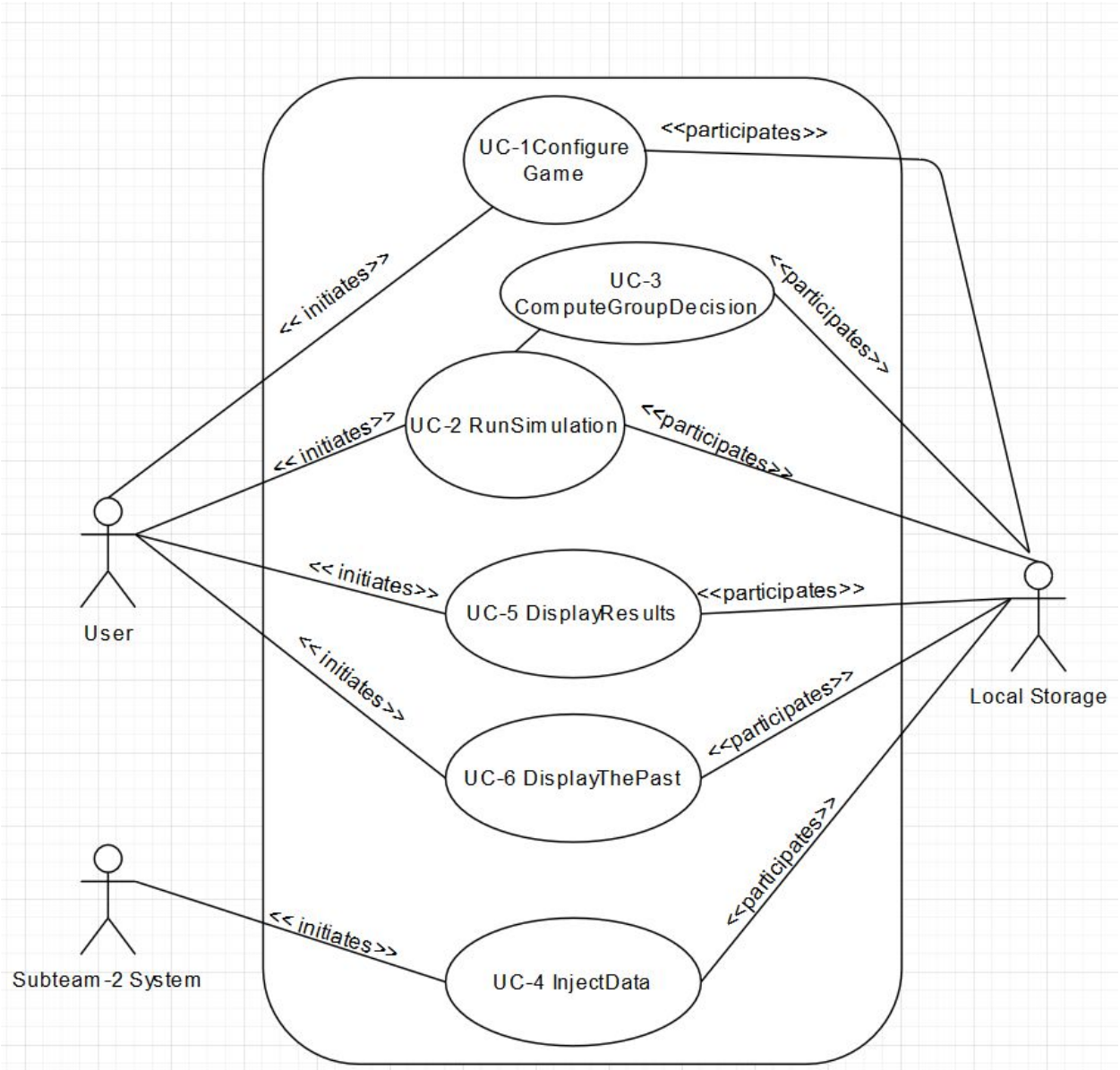
iii) UC-3 ComputeGroupDecision(sub-use case): After the simulation starts the agents are grouped and given personalities and agent-to-agent interactions will be used to re-compute the decision for each agent (agent decisions were computed without agent interactions by Subteam-2 and are given to our system)

iv) UC-4 InjectData: This is how we receive data about each agent's decision from Subteam-2's system. With this data we execute the group decision making process.

v) UC-5 DisplayResults: At the end of the simulation the user will be able to see their set of input conditions and the corresponding agent turnout after however many rounds the user initially set. The user will be given a graph of agent turnout per round for the user-set number of rounds.

vi) UC-6 DisplayThePast: The user will have the option to refer to the past simulations that have been created in the current session. Optionally, the user should be able to download the simulation results and simulation parameters in order to access the information in a different session

b) Use Case Diagram



c) Traceability Matrix

REQ	PW	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6
REQ-1	10	X	X				
REQ-2	10			X	X		
REQ-3	7			X			
REQ-4	7			X			
REQ-5	5			X			
REQ-6	9			X			
REQ-7	3			X		X	X
REQ-8	5			X			
REQ-9	1					X	X
REQ-10	6		X				
REQ-11	7		X				
REQ-12	5						X
REQ-13	5		X				
REQ-14	1	X					
REQ-15	10	X					
REQ-16	5	X					
REQ-17	10					X	
REQ-18	10		X				
REQ-19	1	X					

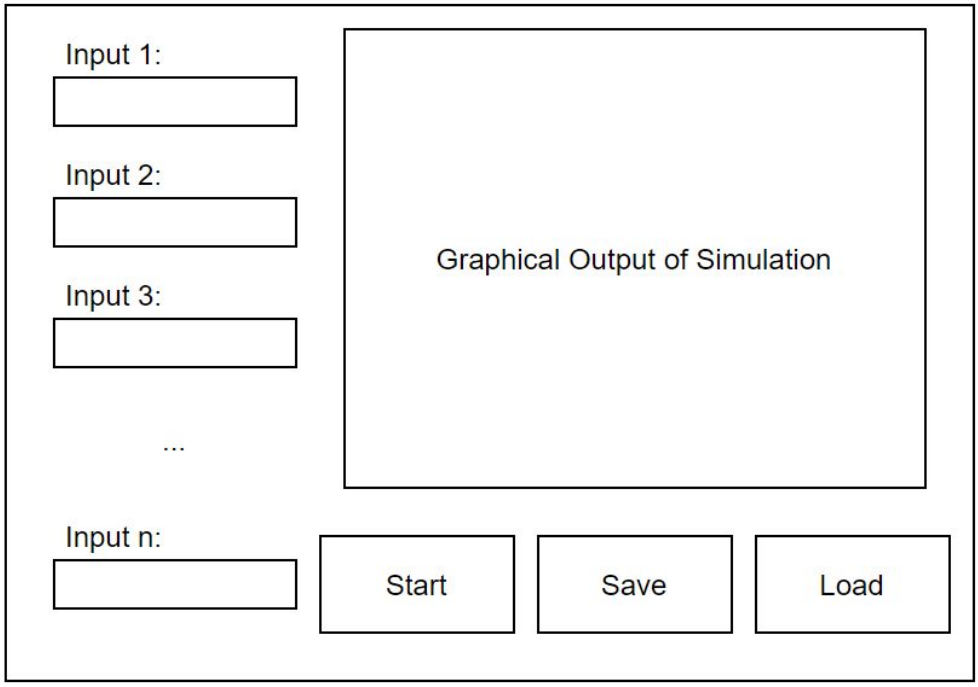
d) Fully Dressed Description

Use Case UC-1: ConfigureGame
Related Requirements: REQ-1, REQ-19 Initiating Actor: User Initiator's Goal: Set the initial conditions of the simulation Participating Actor: Local Storage Preconditions: The entered parameters must be valid Postconditions: The system will have all the necessary data except for agent decisions Flow of Events for main Success Scenario: 1) ⟶ **User** enters the required parameters and presses enter 2) ⟵ **System** validates the parameters 3) ⟵ **System** prints the successful user input message 4) ⟵ **System** is ready for **RunSimulation** once the user presses the start button Flow of Events for Alternate Scenarios: Invalid parameters: 1) ⟶ **User** enters no or invalid parameters and presses enter 2) ⟵ **System** detects the invalid parameter(s) and sends a message to the user 3) ⟵ **User** enters valid parameters and presses enter

Use Case UC-2: RunSimulation
Related Requirements: REQ-10, REQ-11, REQ-13 Initiating Actor: User Initiator's Goal: Start the simulation so that the system can compute the decisions for the groups of agents based off agent-to-agent interactions Participating Actor: Local Storage Preconditions: The initial conditions have been set, user's machine fits the requirements Postconditions: The system is ready to receive data from Subteam-2's System Flow of Events for main Success Scenario: 1) ⟶ **User** presses start button 2) ⟵ **System** verifies user's machine fits functional requirements 3) ⟵ **Subteam-2's System** injects agent data into our local storage Flow of Events for Alternate Scenarios: User's machine doesn't meet functional requirements: 1) ⟶ **User** presses start button 2) ⟵ **System** detects which functional requirements aren't being met 3) ⟶ **User** attempts to meet the requirements

User Interface Specifications

Preliminary Design



On the left are a series of text-boxes where the user will be required to input the starting conditions of the simulations. Once they have successfully initialized all of them, they will be allowed to start the simulation using the start button; said button will be disabled until all inputs have been set. After a simulation, the user will be able to save the results using the save button; this button will be disabled until this requirement is met, similar to the start button. The load button will allow the user to load a previous simulation; it is only disabled while a simulation is currently running. These series of disabling and enabling will help the user work with the UI. Furthermore, the user will be able to hover over either the text-box or the plain-text where each input is specified to receive more information about said input along with what are qualified as valid-values for said input. If they input an invalid value, the UI will prompt them to change the value. Some inputs implemented in the final version may be: restaurant capacity, initial population, number of rounds. In the graphical output section, the following are being planned to be implemented: turnout over time.

User Effort Estimation

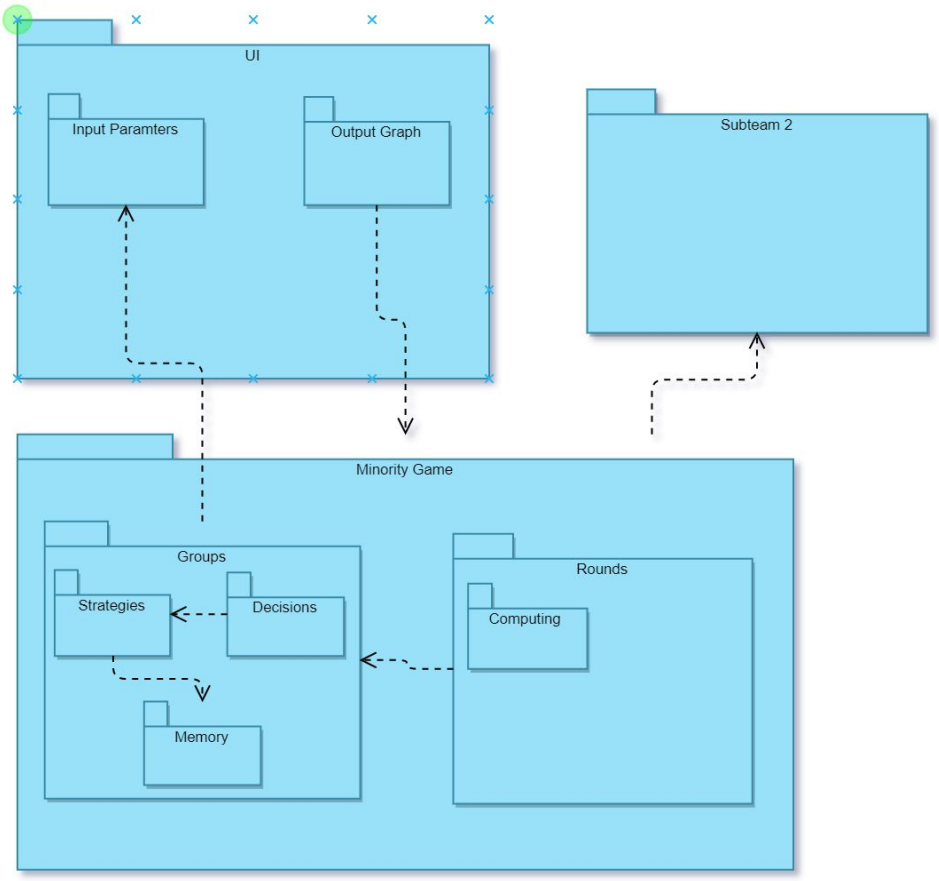
The enabling and disabling of buttons will make it easy for a user to know when they are able to perform certain operations on the UI. Furthermore, they will be able to use the hover-over feature to learn about the different inputs, but will have to be savvy enough

to realize this is a feature. The UI will also let them know when an invalid value is inputted. All together, this will make the UI easy for new users and a breeze for experienced ones. The minimum number of clicks to use the UI is 0 (the user is able to tab through the UI), but it is effectively n+1. The user must first navigate to the Input 1 text-box, then fill it out, then to Input 2, then 3, and so forth to n. The user will then have to navigate to the start button to start the simulation. All together, the user will need to perform n+1 navigation and n clerical data entries to run a unique simulation.

System Architecture

Identifying Subsystems

Subteam-1 will work on 2 different subsystems which will be the UI and the Minority Game subsystems. The UI subsystem will include the input parameters and the output graph. The inputs will be sent to the Minority Game subsystem in which groups of agents will interact with each other in order to make strategies and decisions, of which the final decisions will be output to Subteam 2.



Architecture Styles

Our architectural design style is a layered structural format. We chose this format for easy integrability with subteam-2's system. Our idea was to have two different systems that rely on each other once integrated, but can fully function independently. This is also the main reason why our both subteams have similar system structures. When we integrate the systems, we will only need to slightly modify the UI layer and we can easily add the functional layer from one system to the other. This layered architectural design allows both subteams to work independently on different aspects of the simulation.

Mapping Systems to Hardware

N/A

Connectors and Network Protocols

N/A

Global Control Flow

The application is a procedure-driven system. The user gives the application a series of inputs and then waits for the application to finish processing with the given series of inputs. During that time the program loops through a series of set events which collectively qualify as a round. First the program gains agent decision for the round then uses that data to compute the group decisions for the said round. The groups then go out to where they have decided previously and agents are infected randomly based on the attendance. The outcomes of the round are then sent back so they can be used to computations in the next round. This loop is repeated until a specified amount of rounds is completed. The results of these rounds are then displayed for the user to interpret. There are no timers in the system.

Hardware Requirements

The hardware requirements are the same as Subteam-2's system. They will be further elaborated there to reduce redundancy

Project Size Estimation

Use Case	Complexity Description	Weight
UC-1 ConfigureGame	**Simple** The user will have labeled text-boxes to fill in corresponding to the input conditions for the	5

	simulation. It will be very clear to the user what an acceptable value for each input. If an incorrect value is inputted, the UI will display the value is invalid. From a development standpoint, this should only require a series of if-statements to catch incorrect values.	
UC-2 RunSimulation	**Simple** The user will be provided a click-button on the UI to start the simulation. If the game is configured, the simulation will start; if the game is not, the simulation will not start and the user will be told to configure the game first. This will simply take a check to see if the game is configured.	5
UC-3 ComputeGroupDecision	**Complex** Groups make decisions based on the individual decisions of each agent as derived from Sub-team 2 and complex algorithms. The complexity of the algorithms leads to this being a challenging use case to tackle.	15
UC-4 InjectData	**Average** After each round is over the outcome of that round is processed and handed over to Subteam 1. This calculation will have to be done for every agent in the simulation.	10
UC-5 DisplayResults	**Average** After all rounds have been completed the results of the simulation are displayed on the UI in a manner that is easy for the user to interpret. This will involve	10

	keeping track of data from every round of the game.	
UC-6 DisplayThePast	**Simple** A previously saved simulation is loaded. The results of that simulation are then displayed to the user. This will simply require reading from a save file.	5

Plan of Work

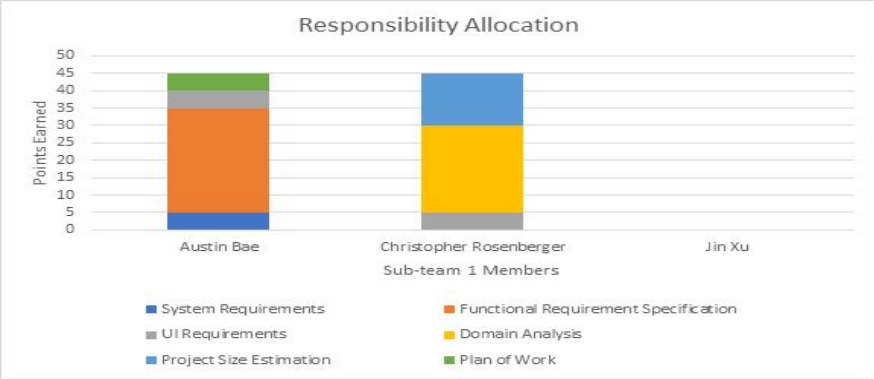
Since we are comparatively short on manpower due to the absence and complete lack of communication from Jin Xu, the most efficient course of action would be to assign a person to each layer of the system, that is have one person working on UI while the other works on the group decision making process. Because the UI portion is expected to be significantly smaller in scope, the person assigned to UI would assist the other in the group decision making process. With the deadline for the first demo in mind, we will prioritize the requirements with the highest weight, with the final goal being the integrated system that contains aspects from both Subteam-1's and Subteam-2's systems

UI Priority: Austin Bae
Group Decision Making Priority: Chris Rosenberger

	Week 1	Week 2	Week 3 (Demo 1)	Week 4	Week 5	Week 6	Week 7	Week 8 (Demo 2)
Create Agent Grouping Algorithm								
Create Agent/Group Personality Matrix								
Create Group Decision Making Algorithm								
Create Group Strategy Making Algorithm								
Create Group Strategy Evaluation Algorithm								
Input Parameter UI implementation								
Simulation Graph UI implementation								
Simulation Final Results UI implementation								
Input Parameter Descriptor UI implementation								
Clean and Finalize UI design								
Combine subteam systems								

References

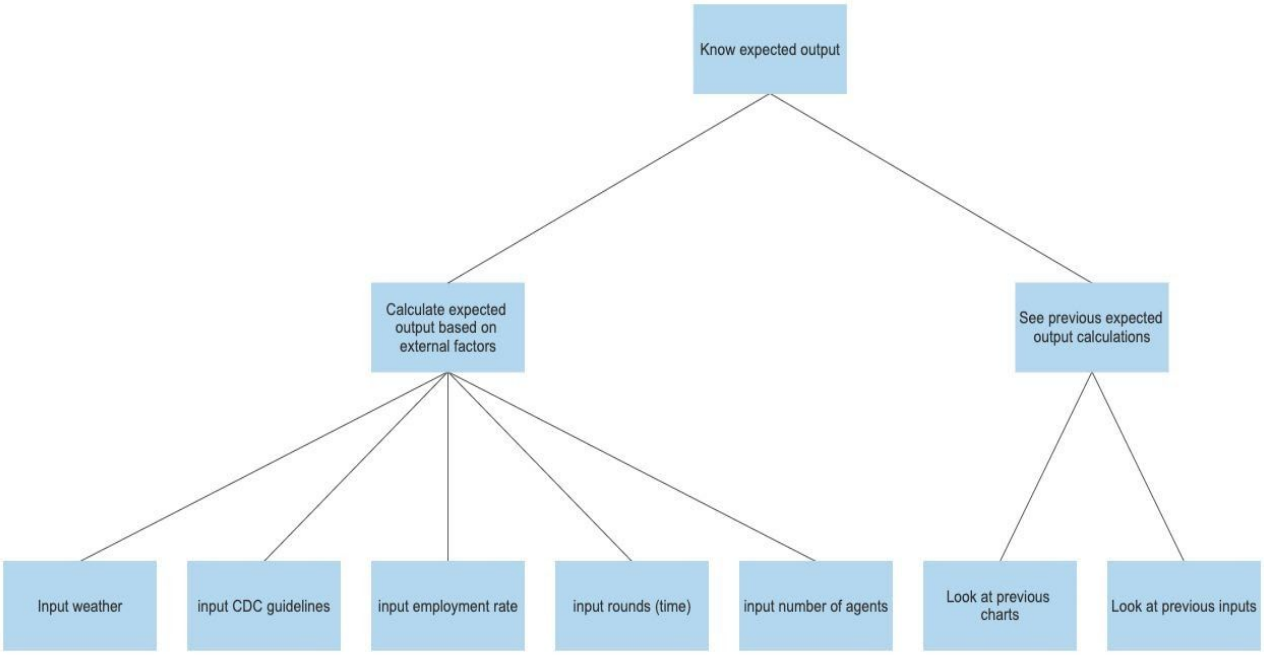
https://www.ece.rutgers.edu/~marsic/Teaching/SE/report1.html
https://iopscience.iop.org/article/10.1088/1367-2630/12/7/075033#:~:text=1.,(in%20parallel%20decision%20mode).
https://en.wikipedia.org/wiki/El_Farol_Bar_problem



Sub-team 2:
Tema members:
Tatsat Vyas , Eric Robles, Taranvir Singh , Nitin Ragavan

System Requirements:

Business Goals:



Enumerated Functional Requirements:

REQ-#	PRIORITY Weight 1 = low, 10 = high	Requirement Description
REQ-1a	10	Users should be able to set all the current CDC guidelines(restaurant capacity)

REQ-1b	8	User should be able to set overall state economy(unemployment rate)
REQ-1c	10	User should be able to set rate of infection in the city
REQ-1d	10	User should be able to set weather condition
REQ-1E	10	Users should be able to set # of rounds in the game
REQ-1F	5	User should be able to input number of agents
REQ-1G	8	Users should be able to define the threshold for minority/majority
REQ-2	10	The application should output the expected agent decisions at the end of the rounds as a graph
REQ-3	7	The user should have option to see the description of each input parameter
REQ-4	5	The user should have option to see the past 3 iteration of inputs
REQ-5	5	The user should have option to see the past 3 iteration graphs
REQ-6	7	The application Should be able to give error to incorrect Inputs
RWQ-7	7	The user should have option of download the resulting graphs

Enumerated non-Functional Requirements:

REQ-#	PRIORITY Weight 1 = low, 10 = high	Requirement Description
REQ-8	7	The application should be able to handle at least 1000 rounds
REQ-9	8	The application should not have any visual glitches
REQ-10	4	The application should work on various operating systems

User interface Requirements:

REQ-#	PRIORITY Weight 1 = low, 10 = high	Requirement Description
REQ-11	10	Must have boxes and labels for each input parameters
REQ-12	10	Must be able to output an easily understandable graph of each round
REQ-13	10	Must be able display final result of the simulation
REQ-14	10	All parameters must be visible, neatly grouped, and editable by the user
REQ-15	10	There must be a clearly visible button that calculates output
REQ-16	6	Each parameter box should have a descriptor button that describes what it is

Functional Requirements Specification

Stakeholders

There are many types of venues and establishments that would be interested in this software. Not only is it great for restaurants and bar owners who are trying to predict outcomes and interpret trends, but it can also be used by anyone who is around many people with limited space. This could be for airports, gyms with limited machines, hospitals with limited beds, any several other types of establishments. In times like these, it is extremely important to be able to know how many people will arrive and how to prepare for them, This software, with a bit of tweaking, can very accurately help out managers, owners, and data analysts to predict, organize, and prepare for many different events and situations.

Actors and Goals

Initiator	Initiator's Goal	Participants	Use Case Name
User/visitor	Enter data into the program	Database	Initial simulation start(UC-1)
User/visitor	Run the simulation in program	Database, Sub-team 1 system	Initial simulation start(UC-1),Data validation(UC-2)
User/visitor	Obtain information about expected turnout	Sub-team 1 System	Compute Agent strategy(UC-3),Compute Agent

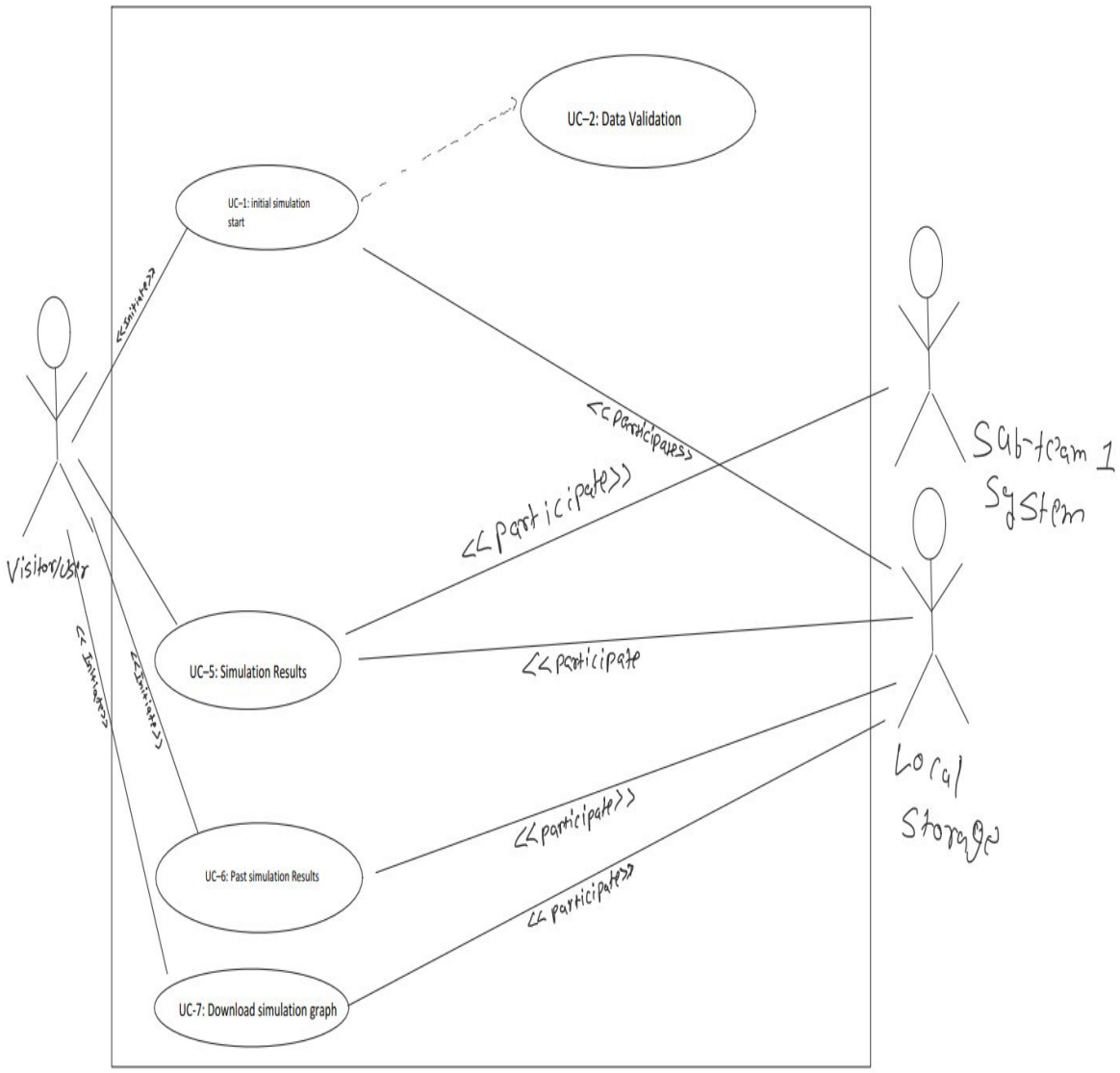
			decisions(UC-4), Display simulation results(UC-5)
User/visitor	Refer to old runs and turnout graphs	Database	Past simulation results(UC-6)
User/visitor	Download the resulting graphs	Database	Download simulation graph(UC-7)
Sub team 1 System(Internal Sub-System)	Get result of the simulation for the agents	Database	Compute Agent decisions(UC-4)

Use Cases

Casual Description:

1) UC#1 Initial simulation start:
   Users can first enter the necessary fields required by the application on the main page. The data entry page is the main page of the application when the user first enters the application.
2) UC#2 Data validation(sub-usecase):
   The data validation happens when the user enters the data into the application and presses the start button. The entered user data will be checked for any invalid parameters.
3) UC#3 Compute Agent strategy(sub-usecase):
   After the simulation has started the application will start by generating dummy agents and creating random strategies for the dummy agents. These strategies will be re-evaluated in a loop after executing UC-4.
4) UC#4 Compute Agent decisions(sub-usecase):
   The agents' decisions will be computed by evaluating the strategies. The current strategies will be evaluated and give a weight to the strategies and depending on the weight a binary decision will be made. This will happen for the given number of rounds
5) UC#5 Display simulation results:
   At the end of the computation the user will be able to see the simulated results at the end of each round. The user will be given a graph and the result of the last simulation.
6) UC#6 Past simulation results:
   The user will have the option to refer to the past simulations that have been created in the current session.
7) UC#7 Download simulation graph:
   When the system is done computing the simulations, the user will have the option to download the simulation as well as any past simulations.

Use Case Diagram:



Traceability Matrix:

REQ	PW	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC7
REQ-1	10	x						
REQ-2	10			x	x	x		
REQ-3	8							
REQ-4	7						x	
REQ-5	7						x	
REQ-6	7		x					
REQ-7	6							x

Maximum Weight	10	7	10	10	10	7	6
Total Weight	10	7	10	10	10	14	6

Fully-Dressed Description

Use Case UC-3: Compute Agent strategy

Related Requirements: Req-2
Initiating Actor: Uer/Visitor
Initiator's Goal: Get simulation result
Participating Actor: Simulation Creator

Preconditions: The input parameters are verified by the system

Minimal Guarantees: black strategy for each agent
Success Guarantees: A non-black strategy with conditioned weights

Plan of Action for main Success Scenario:

1) → The user enters the valid parameters and starts the simulation
2) ← The inputs are verified by the system
3) ← The system computes the strategies for each agent

Plan of Action for Contingencies:

The user enters Invalid data:
1) ← System detects the error and sends a message to the user
The system computes black Strategy:
2) ← System sends system error and give the user option to report the error

Use Case UC-1: Initial Simulation Start

Related Requirements: Req1
Initiating Actor: Uer/Visitor
Initiator's Goal: start the simulation
Participating Actor: None

Preconditions: The entered parameters must be valid

Minimal Guarantees: Invalid parameters
Success Guarantees: Parameters accepted and simulation start

Plan of Action for main Success Scenario:

1) → The user enters the required parameters and presses enter
2) ← The system validates the parameters
3) ← The system prints the successfully start simulation message

Plan of Action for Contingencies:

Invalid parameters:
1) ← System detects the invalid parameter and sends a message to the user

Use Case UC-6: Past Simulated Results

Related Requirements: Req4,Req-5
Initiating Actor: Uer/Visitor
Initiator's Goal: Get past simulation result
Participating Actor: None

Preconditions: At least one Iteration of simulation has been done

Minimal Guarantees: No simulations present
Success Guarantees: Graph of the each round minority decisions

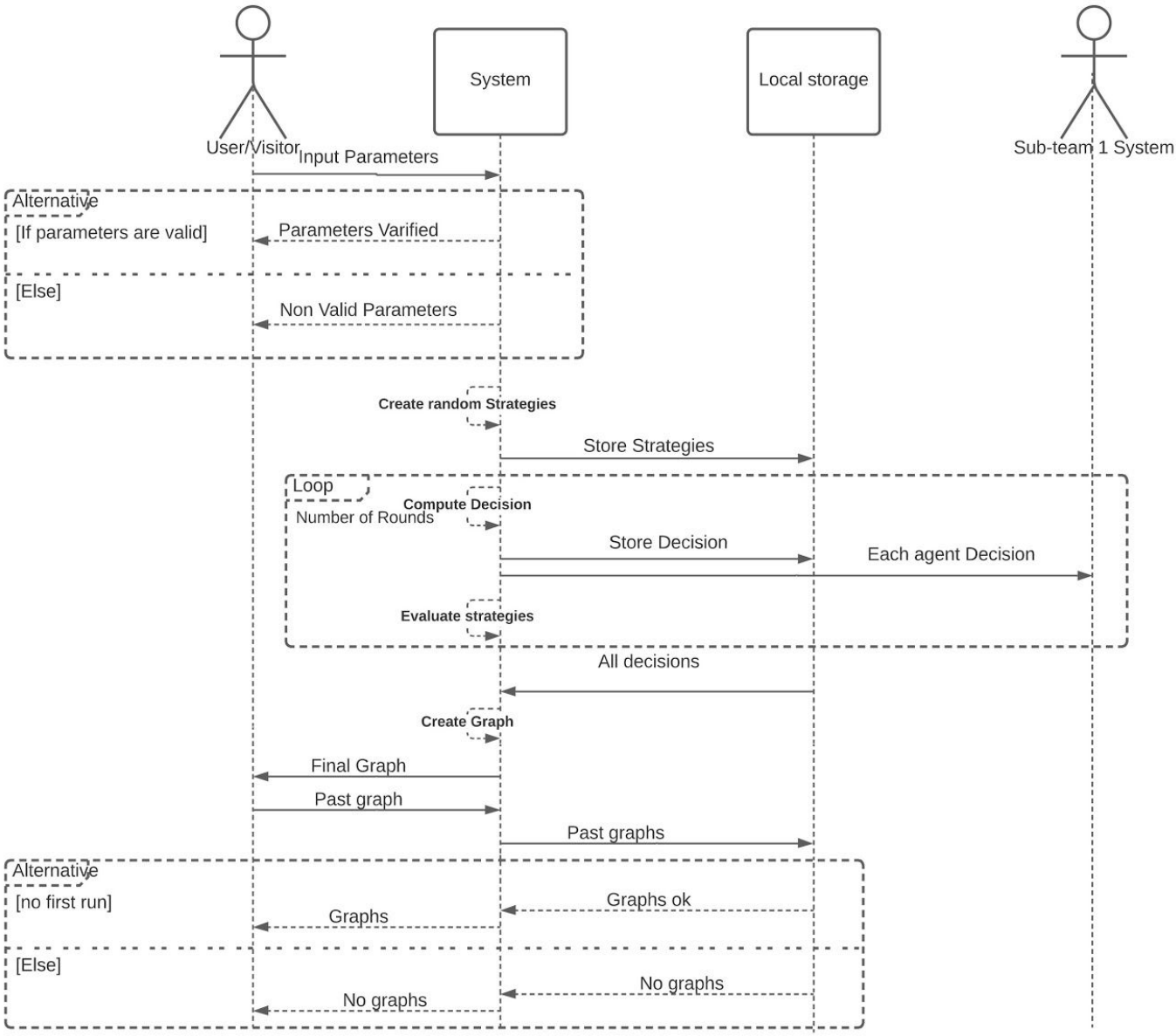
Plan of Action for main Success Scenario:

1) → The user presses the past simulation button
2) ← The system checks for the past simulation
3) ← The system prints the graphs of the past simulations

Plan of Action for Contingencies:

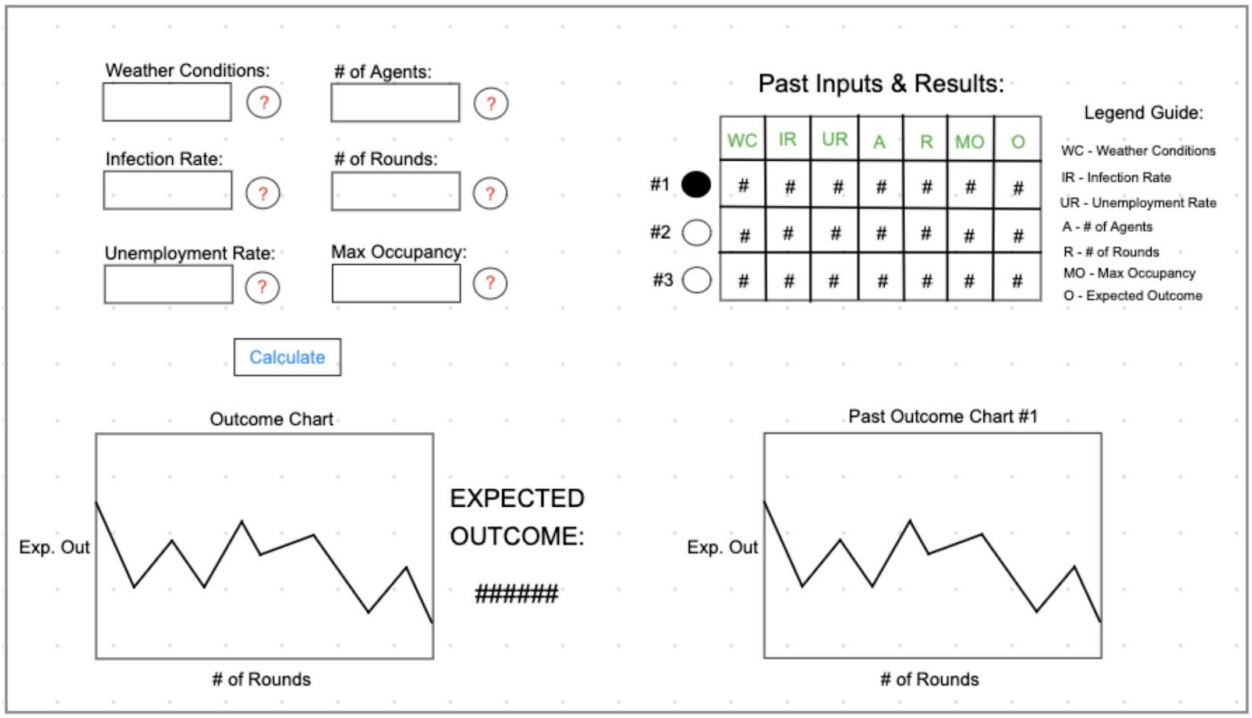
No past simulations:
1) ← System detects the error and sends a message to the user

System Sequence Diagram:



User Interface Specification

User Interface Design Example:



(*Note: Max Occupancy is really just the CDC guideline parameter, will probably rename it later, as well as some other parameters. This is just a general guide.)

Preliminary Design

- All input boxes, that is for the respective boxes under weather conditions(still unsure of what that input will even be, could just be a binary good or bad weather input), infections rate, unemployment rate, # of agents, # of rounds, and max occupancy(CDC parameter, think of as max allowed by law inside the restaurant) are to be clicked with the mouse pointer, allowing for the user to type in whatever value. Next input parameter box can be navigated to using the same method as before or by pressing tab. Next to each interactable input box is a question mark, that once a user hovers the mouse pointer over (or even clicks on), will display a hovering textbox that explains what the associated input parameter is. Once all parameter boxes have been filled, the user is then to click on the blue calculate button, which will calculate the expected outcome and display that result under the expected outcome label, as well as the associated graph that plots the expected outcome vs. a number of

rounds. The data is temporarily stored. If the user wishes to input new parameters and do a new calculation, the same steps as before following, the only difference is that once the calculate button is hit again, the temporarily saved data from the previous calculation gets displayed on the past inputs and results chart. The user can then click on the circle button next to the #1 row on the past inputs and results chart (a filled-in button represents a button that's been clicked on) to display the associated graph in the bottom right corner. As the user continues to make more calculations the chart will fill up to 3 saved slots/rows. If another calculation is made after that, then the data from row three will be deleted, row 1 and 2 data move down one row, and the now-empty row #1 will be filled with the latest saved calculation data.

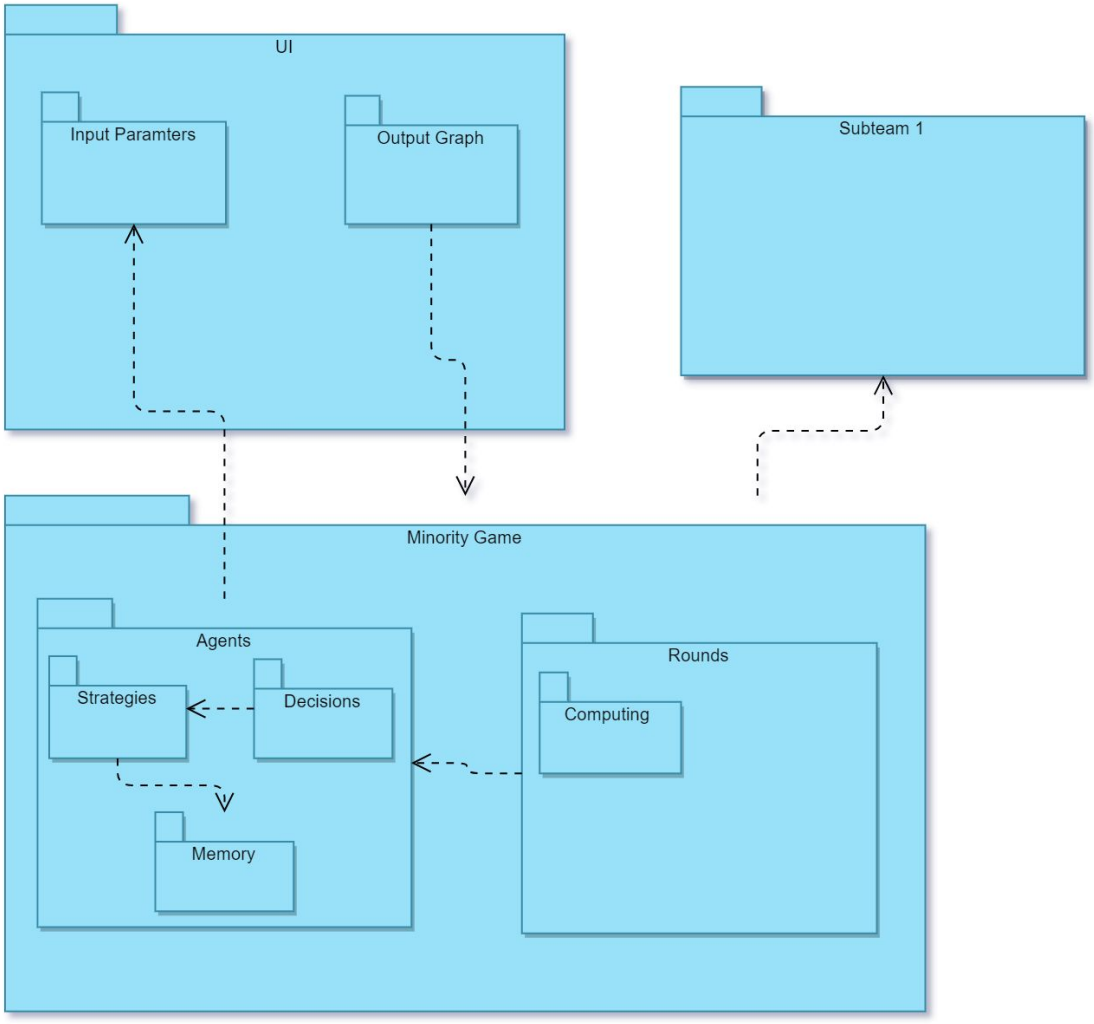
User effort estimation

- Based on what was said in the preliminary design, for someone who wants to calculate the expected output for a set of parameters (all value 1), the navigational clicks and clerical clicks are as follow:
    - Navigation
        - Click on the first box
        - Click on the calculate button
    - Clerical
        - Type “1”
        - Tab to move to 2nd box
        - Type “2”
        - Tab to move to 3rd box
        - Type “3”
        - Tab to 4th
        - Type “4”
        - Tab to 5th
        - Type “5”
        - Tab to 6th
        - Type “6”
    - In total it's about 13 clicks/keystrokes for this simple case, though I have devised an approximate formula (if inputs have the same number of integers)
    - c(6*i)+g, c=calculations, i=integers per input box, g=times user clicks on past data graph button
    - Overall not a lot of clicks or keystrokes required to use this program.

System Architecture

a. Identify Subsystems

i. Subteam 2 will work on 2 different subsystems which will be the UI and Minority Game subsystems. The UI subsystem will include the input parameters and the output graph. The inputs will be sent to the Minority Game subsystem in which Agents will form strategies and decisions. These decisions will be sent to the Rounds subsystem in order to compute and create graphs, which will then be outputted in the UI.



b. Architectural Styles

i. Our designs architectural style relies mostly on a loosely layered structural format. We chose this format due to its inherent simplicity, notably how the structure behaves somewhat like an easy to manage top down design, except with the added flexibility of being able to move data between each adjacent component bidirectionally. The reason we went with a loosely based design instead of a strict one is that some components of our program that are lower level require direct connection to higher level components. A layered system also makes it easier to integrate subteam 1's integratable component from any point. Our system will have two tiers, one for the user interface, one for the system functions. The reason for this multi-tiered architecture is to be able to clearly distinguish the unique functionalities of each structure, and to avoid any mix-ups.

c. Mapping Subsystems to Hardware - N/A

d. Connectors and Network Protocols - N/A

e. Global Control

i. Execution orderness is procedure-driven, as the user has to input values into the parameter boxes and click the calculate button every time the user wants to see a new expected output. The user must go through these same steps everytime without exception if they wish to see a newly calculated expected value. The user can check previous 3 input histories at any time, however this has no effect on the calculation of a new output calculation.

ii. There are no timers in our system, as its design is of event-response type, since all calculations are done without any concern for real time.

f. Hardware Requirements

i. The user will need a 640x480 resolution display capable of outputting color to see the program's user interface, a mouse and keyboard to navigate through the program, at least 4 GB of RAM (just to even be able to run the Windows OS), at least a dual core cpu (with integrated graphics if no dedicated gpu), at least 10 MB of allocated disk space to store the program and save data, and some form of internet connection to be able to download the program.

Project Size Estimation Based on use case points

Each trial will simulate a certain sample size, this is represented by an array. The project would need to take into account the multiple agent structure, and will use mixed strategies for each agent involved. Assuming 100 trials, there would need to be 100 separate arrays each with a different step per person. The processing would need to take into account every parameter (as many as 5) for each agent. These parameters would affect the weight of each strategy, and in each step the results of the previous step would be taken into account.

Plan of Work

The plan would be to use the “divide-and-conquer” approach when dealing with the entire sub system. The UI, as well as the system, are two separate parts and should be handled individually with each subteam dedicated to one part. The next couple of weeks, our team will be dedicated to using python and scripting the various game theory scenarios. For every agent, the program will calculate winners and losers based on pure chance and mixed strategy, as well as the input parameters (which will be finalized later). These models will be graphically represented, with the graphical information eventually fed back to each and every user of the program.

Team 1: Eric, Tatsat

Team2 : Nitin, Taranvir


Week 1
Week 2
Week 3(First Demo)
Week 4
Week 5
Week 6
Week 7
Week 8(Second Demo)
Brain storm strategies
Finalize UI Design
Input parameters UI implementation
Simulation graphs UI Implementation
Past Simulation graphs UI implementation
Input varification
Agents and strategies setup
Decision Making Algorithm Implementation
Strategy Evaluation Algorithm Implementation
Final Simulation Graph Implementation
Additional Simmulation completion tasks
Minority game simulation testing
End to End UI testing
Sub-teams Integration
Legend:
Team 1
Team 2

Breakdown of responsibilities:

Team Member	Done	Doing	Will Do
Tatsat Vyas	BrainStorm project algorithm Ideas, Research relevant minority game topic and Initial separation of overall project into sub-projects	Over all project management(github repo, task assignment, weekly meeting coordination and more)	Continue project management, sub-project integration.
Nitin Ragavan			
Taranvir Singh			
Erik Robles	Some minor work, constant reading of textbook for guidance	Small edits, formatting some stuff	Coding

References

https://www.ece.rutgers.edu/~marsic/books/SE/projects/MinorityGame/ElFarolBar.pdf

https://www.youtube.com/watch?v=PJNmlfqh1J0&ab_channel=ComplexityExplorer