Report 1
Group 15

Beach Simulation

Jose Villamor Delgado
Sarid Shinwari
Yang Ren
Gegi Oniani
Steven Fisher
Tanzina Farzana

Daniel Davis

Table of Contents

1. a. Problem Statement

As a coastal business owner, running a profiting beachfront shop is already difficult. Running our business includes inventory, staffing, and many other factors that all depend upon the general population that is passing through our store. Like all businesses, the factors of profit are sales and costs. In order to keep costs down, we must be able to predict demand and adjust accordingly. If too many staff members are on hand, or too many supplies are purchased and not used/sold, it cuts into profit. It is an important skill to be able to predict the amount of customers and their spending habits. However, unlike a mall or mainland store, coastal markets are difficult to predict due to the variability of seasons, tides, and other factors.

Because this is an integral part of our business, we need a software that will simulate a large amount of these factors. By simulating the population, our business will be able to better predict the weeks' customer flow. This can help us cut unnecessary costs like overstaffing and overstocking.

Weather and tides are an important factor in a customer's decision to visit the beach. If it is raining, little to no visitors will come to the beach that day. Rough tides will also detract beachgoers that come for water sports, i.e. surfers.

In addition to weather, customers are also affected by their past experiences. A beachgoer may not visit the same beach again if it was too crowded to be enjoyable on a previous visit. This is a game theory problem that involves predicting the actions of others; if a beachgoer predicts that the beach is going to be too crowded, he/she may choose to stay home. On the opposite side of the spectrum, a beach can also be deserted, displeasing others (i.e people-watchers). The simulation should take this into account.

The simulation should also take into account the demographics of beachgoers. The population can be split into both gender and age groups. Each demographic has subtle preferences that affect their experience. Split into age groups, children may be more affected by rough tides than adults. Male young adults may prefer to go to beaches where young female adults frequent. The demographics for the simulation should be able to be set individually. For business purposes it is important to know what age groups affect the customer flow. Expanding our target audience to include a certain age group, as well as seeing how that age group effects the simulated crowd, will help with our marketing efforts.

Locations that will display real time population statistics as a default is another feature necessary for this software. This will let us simulate our multiple business locations, and in turn decrease the time it takes to simulate the population. We would like to avoid having to look up our population statistics to input.

When simulating the population, there should be a way to save any settings previously set. By allowing custom settings to be saved, it will reduce the amount of time to run a specific simulation multiple times.

In the simulation, agents (beachgoers) should aim to “win” by having an enjoyable experience at the beach. Their strategy should be based on past wins and future predictions. If an agent continues to lose, it should change its strategy. The simulation will return the simulated population percentage that chose to go to the beach on a given day.

1. b. Glossary

Population: Amount of people residing in a certain area. Can be based upon, city, county, state, etc. Also can be used in explaining an amount of people.

Agents: A simulated person that makes a decision based on their personal strategy, and contributes to the simulated population that the results are based upon.

Win: Each agent takes into account all the variables, and population results and determines if they would have an enjoyable time at the beach.

Strategy: Each agent will have set criteria on how they will base their decision. They can then pick a new strategy if they aren’t winning with one.

Self-Optimization: The process where agents will vary their strategy choices dependent on if they win.

Simulator: Running thousands of virtual days of simulations to end with an approximate answer of how many people would go to the beach. Resulting in an accurate representation of the projected population for the day.

Business User: The user who will be using the user interface. This is the main customer of our product. They will be able to simulate with variable control to gain information and data.

Variables: Different settings that are able to change to directly affect the population simulation. In this case, weather, tidal weather, time, date, and location are all variable in our program.

Simulated Population Percentage: This is the result of the simulation. After all the agents have repeatedly made a decision, there will be a stabilized average of what percent of the population would go according to the set variables.

Population to Agent Ratio: The number of people one agent represents. If a population is 500,000 people, instead of simulating 500,000 agents, we can define one agent to every 1,000 people. Resulting in only 500 agents.

2. a. Enumerated Functional Requirements

Identifier	Priority Weight (Low 1- 5 High)	Requirement Description
REQ-1	5	Agents decide if they go to the beach or not. The majority loses
REQ-2	5	Each agent has a personal strategy.
REQ-3	5	Agents with winning strategies repeat the winning strategi Agents with losing strategies may change them.
REQ-4	5	The more recent an Agents win/loss is, the more likely the agent is to consider it in its strategy.
REQ-5	5	The system initial conditions have a default setting.
REQ-6	3	The system default to the current day.
REQ-7	3	Weather forecasts used from the system are taken off of a website's 10 day forecast.
REQ-8	3	Agent action depends heavily on the weather conditions o the day.
REQ-9	5	The system updates graphs in real time depending on the users chosen actions.
REQ-10	3	If an area is selected with multiple beaches, it is possible f there to be multiple winning beaches.
REQ-11	4	The system runs the simulation multiple times so a good d set can be made.
REQ-12	1	Business analytics are included in the system to help

		beachside businesses
REQ-13	1	Actual area demographics are used to give a more accurate prediction.
REQ-14	2	Agent strategies should account for friend,family, and neighbor agents strategies.
REQ-15	2	Some agents should account for tidal forecast.
REQ-16	2	Agents should account for what day of the week it is.
REQ-17	2	An agent's strategy depends on personal agent factors, suc as age.
REQ-18	2	Agents should favor dates that are more convenient for the personal factors (such as various holidays, spring break fo students, etc..)
REQ-19	2	Some agents should have the ability to get sickness, preventing them from going to the beach.
REQ-I	5	The system will allow the user to run the software on diffe computers
REQ-II	4	The system shall allow the user to negotiate the software parameters
REQ-III	3	The system shall allow users to register with unique identi
REQ-IV	2	The system shall allow the user to specify their own parameters

REQ-(1-11) Defines what the consumer believes are the most important variables that need to be included in the application to run the simulation based off of.
REQ-(13-19) Defines the expected AI interaction within that region that the user requested.
REQ-I Is intended for better compatibility which will allow the user to access their subscription from anywhere.
REQ-II Is implemented so that if the user ever changes his preferences or the regions parameters have shifted, the user can adjust them as needed.
REQ-III This requirement is meant to allow separate users to keep their own individual settings, and allow the user to be more productive as a result.
REQ-IV This is a non essential requirement that will allow the user to alter the AI, as per their needs and only change it for that one individual user.

2. b. Non-Functional Requirements

Non-functional requirements are requirements that describe how the system/system environment should be, rather than what the system needs to be able to do. A method of categorizing those requirements is FURPS+. FURPS+ stands for Functionality, Usability, Reliability, Performance, and Supportability. The + in FURPS+ stands for additional requirements such as design constraints, implementation

requirements, interface requirements, and physical requirements. Since we are focusing on the non-functional requirements, only the URPS+ will be taken into consideration.
Usability includes the ease of use, aesthetics, and documentation. For usability, we will focus on implementing the application on a web browser through the use of javascript. The purpose of implementing it online is to allow users to access their saved simulations from anywhere. In terms of aesthetics, the interface should be pleasing and simple allowing for ease of use. Documentation is a major part of usability as it gives the user the information necessary to navigate the application.
Reliability is defined by the system's uptime and ability to recover from errors. In order to have a high level of reliability, the system should address as many errors as possible that might occur.In addition, should any errors occur, the system should keep a log of such errors and fail gracefully and inform the user of the error.
Performance is relative to how fast the application runs. If the application takes too long to simulate, user demand for the application will drop. The application must have a low response time, low resource time, and be fast in performing calculations. To test performance, while programming the application, time counters should be implemented for all modules in order to find areas of delay.
Supportability includes compatibility which should be addressed to allow users to connect from many different browsers. Modularity to allow us, the programmer to make a system that has easy maintainability, and adaptability, as well as, making the program serviceable. On top of those requirements, the application should be configurable and allow the programmer to adapt the application to the needs of the user.
On top of the main requirements the implementation of the system should have readable code, and set limits on operation to prevent overflow. In addition, the system should be formatted so that all calculations/variables use the same system of measurement.

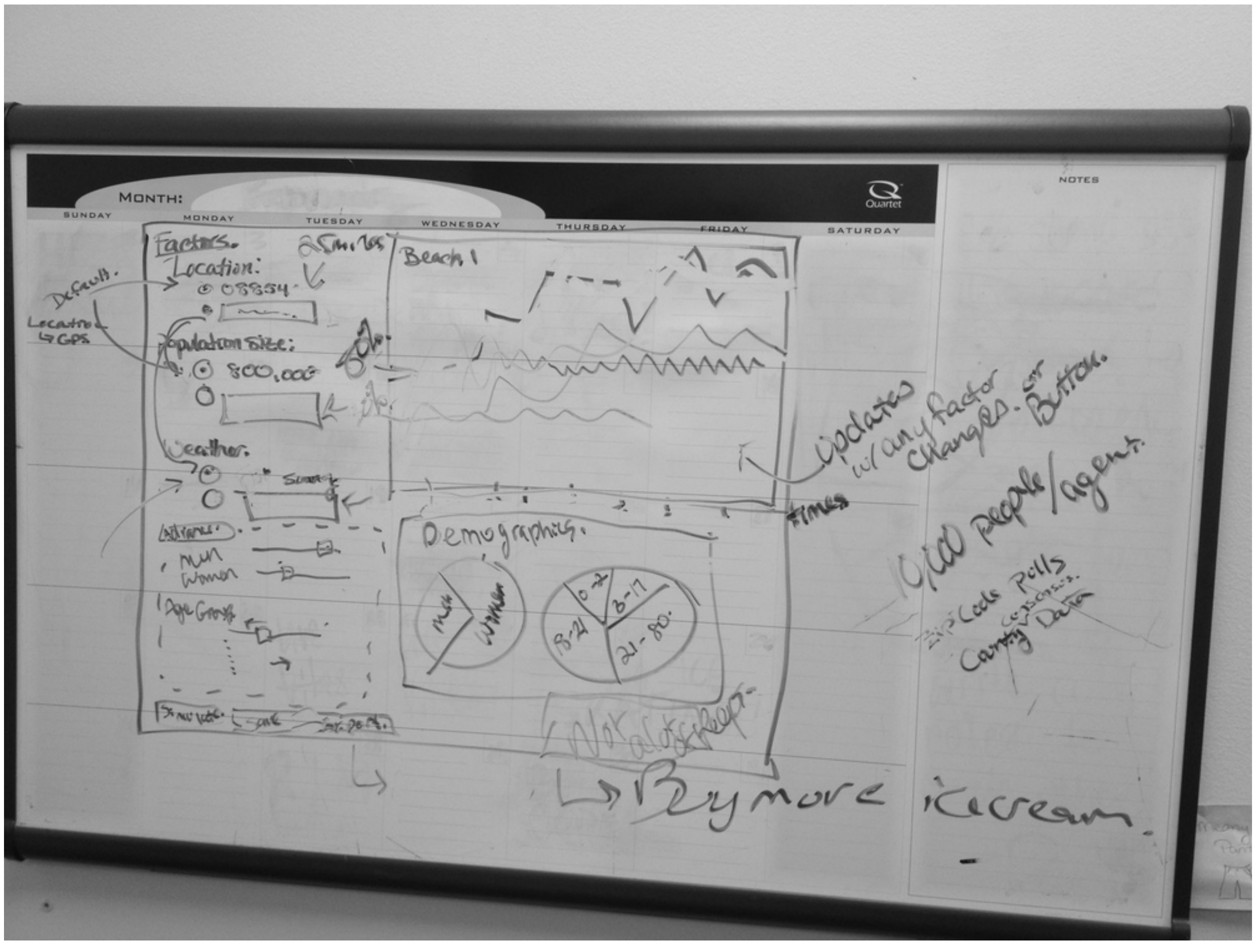
Identifier	**Priority Weight (Low 1- 5 High)**	**Requirement Description**
REQ-21	4	Help option should be available to aid the user (documentation).
REQ-22	5	The user should be able to access their account data from any locatio
REQ-23	5	System should account for any possible errors and set up a system to report the errors and fail gracefully.
REQ-24	3	System should implement runtime counters in order to allow the developer to find areas of delay.
REQ-25	5	System should ensure that there are no memory leaks to prevent needless resource usage.
REQ-26	5	System should be modular in order to allow easy maintenance and adaptation.

REQ-27	3	System should be compatible with multiple browsers.
REQ-28	2	System should be able to be configured on the admin end for specifi users and their needs.
REQ-29	5	System should use good coding practices such as readable code.
REQ-30	5	System should use one standard set of measurements to ensure interoperability.
REQ-31	4	System should have have default values based on the specific user.
REQ-32	4	The user should be able to navigate the application using a maximun three clicks from the root location.
REQ-33	4	The interface should be user friendly, and allow the user to see witho squinting.

These URPS+ requirements are meant to make navigation simpler for the user, and allow for the system manager to have easier access to altering the application without greatly impacting the users. The requirements should define a system with minimal downtime, and good customer satisfaction. The objective of the requirements is to make a system so pleasing to use, that there are no discouraging factors of interaction.

2. c. Onscreen Appearance Requirement

Identifier	**Priority Weight (Low 1- 5 High)**	**Requirement Description**
REQ-34	5	The user interface takes default values such as population size, weather, etc. based on the location given.
REQ-35	3	The user interface able to check the graph based on data input.
REQ-36	2	The user interface would be able to save data and graph , which the would get from the simulation.
REQ-37	1	Visual Demographics, the data we get from the location GPS.



3a. Stakeholders, Actors, and Goals

Stakeholders:
1. Beach business owner/investor
2. Township (Beach Owner)

The beach is a natural resource that can be used for both relaxation and profit. Local businesses count on the beach's popularity for providing customers. Local municipalities are concerned with the tax revenues and the upkeep of the beach itself.

Local businesses have incentive to maximize profit and minimize loss. This means maximizing not only the number of customers, but also the amount spent per customer. Customers are more likely to spend money if they are having a good time. Also, beachgoers will stay longer at the beach if they are having fun. If a visitor stays long at the beach, they are more likely to make food and other purchases. In turn, the business has incentive to keep prices low to attract customers to the general area. Local businesses will use the simulation to evaluate the effect of pricing schemes and marketing on the consumer demand. Businesses will also use the simulation to predict the need for extra staff and other purchases. Since beach use patterns are highly affected by weather patterns, businesses may be hurt by a “bad season”. It is useful to be able to simulate custom weather patterns to get an estimate of how the business will perform in the best and worst case scenarios.

Simulating the beach-going population is useful for event planning and tax purposes. Extra lifeguards must be stationed when the beach is more crowded. This also means more supplies, equipment, and training. In addition, the township is responsible for keeping the beach clean with the use of beach combing equipment and sanitation workers. As such, the township needs taxes to fund these operations. Therefore, the township will use the simulation to gauge operation costs and tax revenue. Some municipalities charge for parking, and the simulation can be used to show how changes in pricing affect beach use patterns and the tax revenue needed to be directed towards upkeep. The township has incentive to increase tax revenue and support local businesses. In addition, the township has incentive to attract new businesses (and therefore, jobs).

3b. Actors and Goals

Actor 1: User [UC-1][UC-2][UC-3][UC-4][UC-5]
[Roles]: Provides data to the system and memory to be interpreted
[Type]: Initiating
[Goals]:
-To register and login to the system, run the simulation, adjust simulation parameters, adjust simulation algorithms

Side: System [UC 1-5]
[Roles]: Provides data to the system and memory to be interpreted
[Type]: Participating

3.c.i. Use Cases Casual Description:

The summary use-cases are as follows:

UC-1: Simulate -- Allow the user to run the simulation using the default settings.
Derived from REQ I-IV. *Only available to registered users.

UC-2: Register -- Allows a user to fill out a registration form to obtain access to the software.

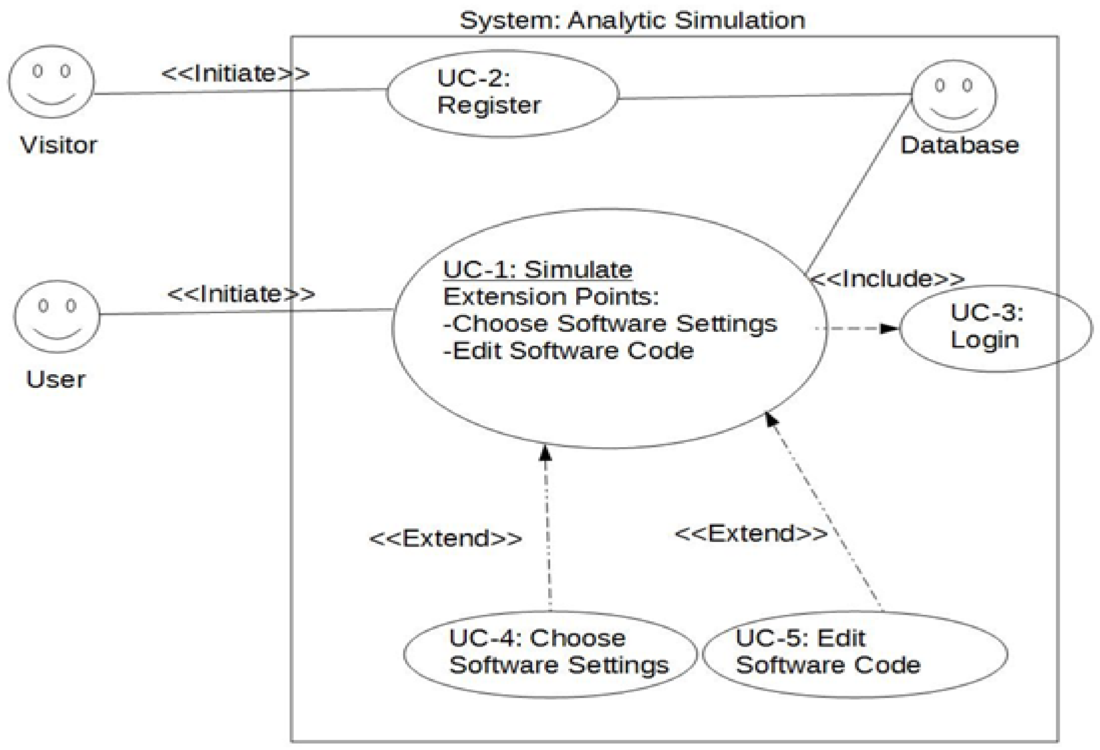
Derived from REQ III.

UC-3: Login -- Allows a user to login to access their saved settings, and simulations.
Derived from REQ III.

UC-4: ChooseSoftwareSettings -- Allows the user to edit the default settings to create new variations in parameters for the simulation to run.
Derived from REQ II.

UC-5: EditSoftwareCode -- Allows the user to modify the underlying software code, so that they can add their own algorithms, and only implements it for that specific user. (Includes a restore to default option and optionally* storage of the last 5 edits)
Derived from REQ IV.

3.c.ii. Use Case Diagram:



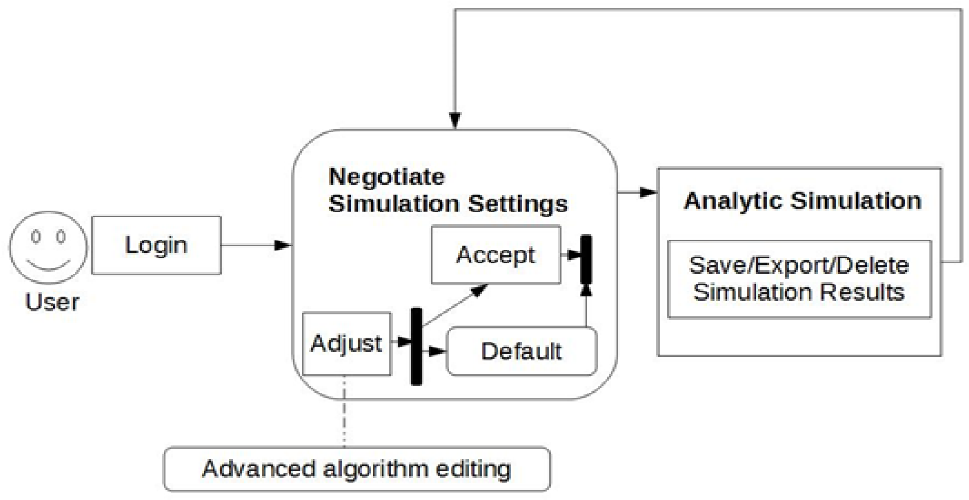
3.c.iii. Traceability Matrix

The traceability matrix shows the relationship between the use cases and requirements.

Req’t	PW	UC1	UC2	UC3	UC4	UC5
REQ I	5	X				
REQ II	4	X			X	
REQ III	3	X	X	X		
REQ IV	2	X				X
Max PW		5	3	3	4	2
Total PW		14	3	3	4	2

Looking at the priority weights:
UC1>UC4>UC3,UC2>UC5
Using this weighting we will choose the use cases with the highest priority and implement them first.

3.c.iv. Fully-Dressed Description:


User
Login
Negotiate Simulation Settings
Accept
Adjust
Default
Advanced algorithm editing
Analytic Simulation
Save/Export/Delete Simulation Results

*Operational model for the analytical simulation software

Use Case UC-1: Simulate

Related Requirements: REQ I-REQ IV
Initiating Actor: User
Actor's Goal: To simulate the expected business
Participating Actors: Database
Preconditions: User is registered
Success End Condition: If the simulation is completed, user is given a choice to save sim.
Failed End Condition: If inputted parameters are outside of range, sim is not run.
Extension Points: Choose Software Settings (UC-4) in step 2
Flow of Events for Main Success Scenario:
include::Login (UC-3)
← 1. **System** displays the default software parameters, as well as, an advanced option button
→ 2. **User** accepts the default conditions
← 3. **System** simulates the results based upon the parameters and displays the results
→ 4. **User** can then choose to save or export the simulation results
--- If user chooses to simulate again based on default settings, steps 2 to 4 are repeated ---
→ 4. a. **User** chooses to save/export
← 5. a. **System** saves the results/exports them
--- Steps 2 → 5 are repeated until the user exits ---

Flow of Events for Alternate Scenario:
→ 2. a. **User** chooses to modify default conditions, and must accept them
← 3. a. **System** checks modified parameters for legality and simulates if legal
--- User is informed if values are not acceptable / Simulation is not run ---
→ 4. a. **User** can then choose to save or export the simulation results

Use Case UC-4: Choose Simulation Settings

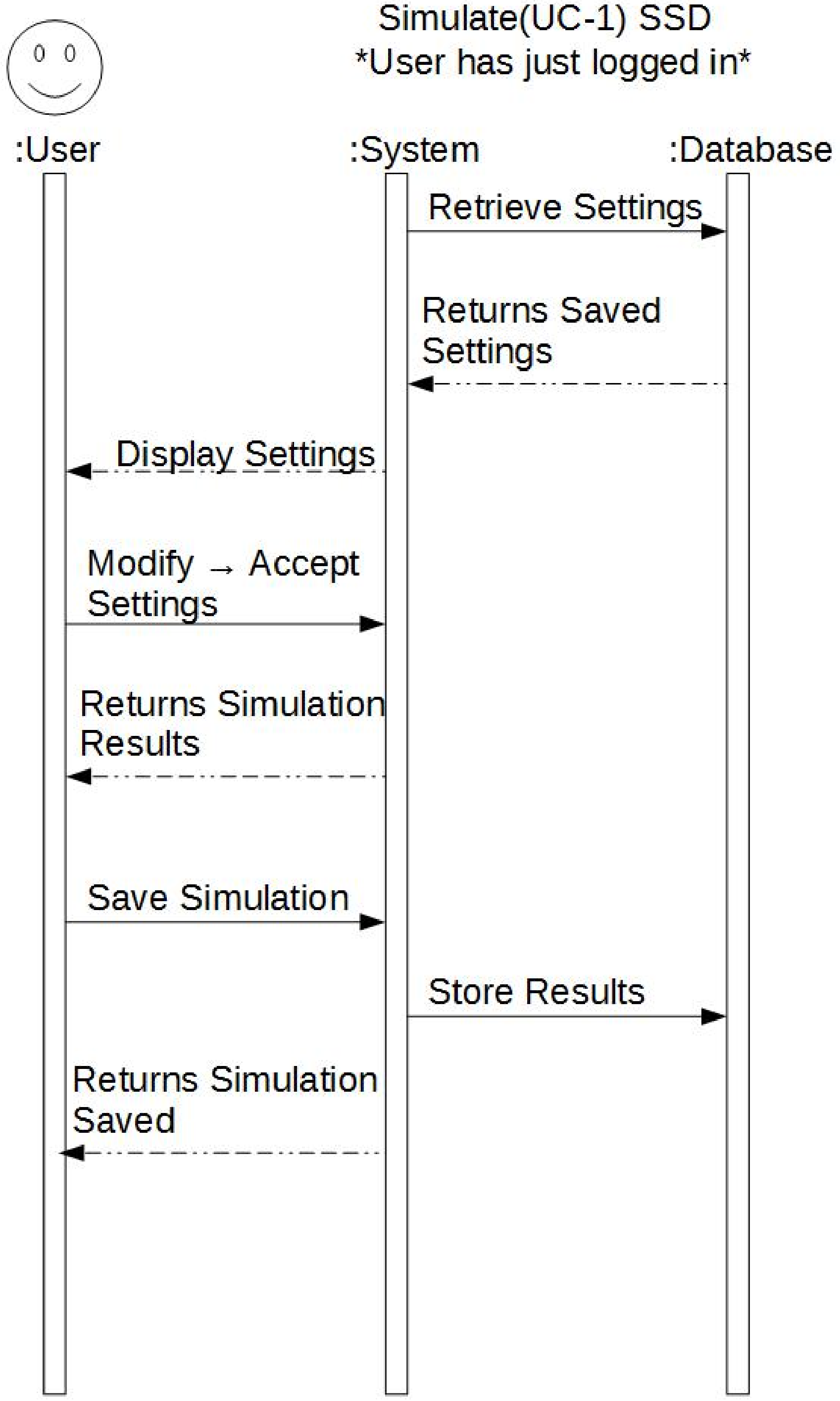
Related Requirements: REQ II
Initiating Actor: User
Actor's Goal: To adjust simulation settings
Participating Actors: Database
Preconditions: User must be logged in
Success End Condition: If parameters are acceptable, settings can be saved
Failed End Condition: If inputted parameters are outside of range, must be reentered
Extension Points: Edit Software Code (UC-5) in step 3. a. → 5. b.
Flow of Events for Main Success Scenario:
← 1. **System** displays the default software parameters, as well as, an advanced option button

→ 2. **User** chooses the advanced option button
← 3. **System** displays all the implemented variables that may be edited
→ 4. **User** can then choose to alter the variables to suit their own needs
← 5. **System** checks the variables to see if they are legal and informs the user if they are not
--- If variables are not legal, user is returned to step 4 ---
← 6. **System** prompts user with option to save their preferences
→ 7. **User** chooses to save
← 8. **System** stores preferences in the database

Flow of Events for Extensions:
→ 3. a. **System** gives the user the choice of directly editing their algorithms
← 4. a. **User** enters their own algorithm and attempts to run simulation
→ 5. a. **System** displays error if algorithm is not correctly formatted/unusable
→ 5. b. **System** accepts input and sim is run

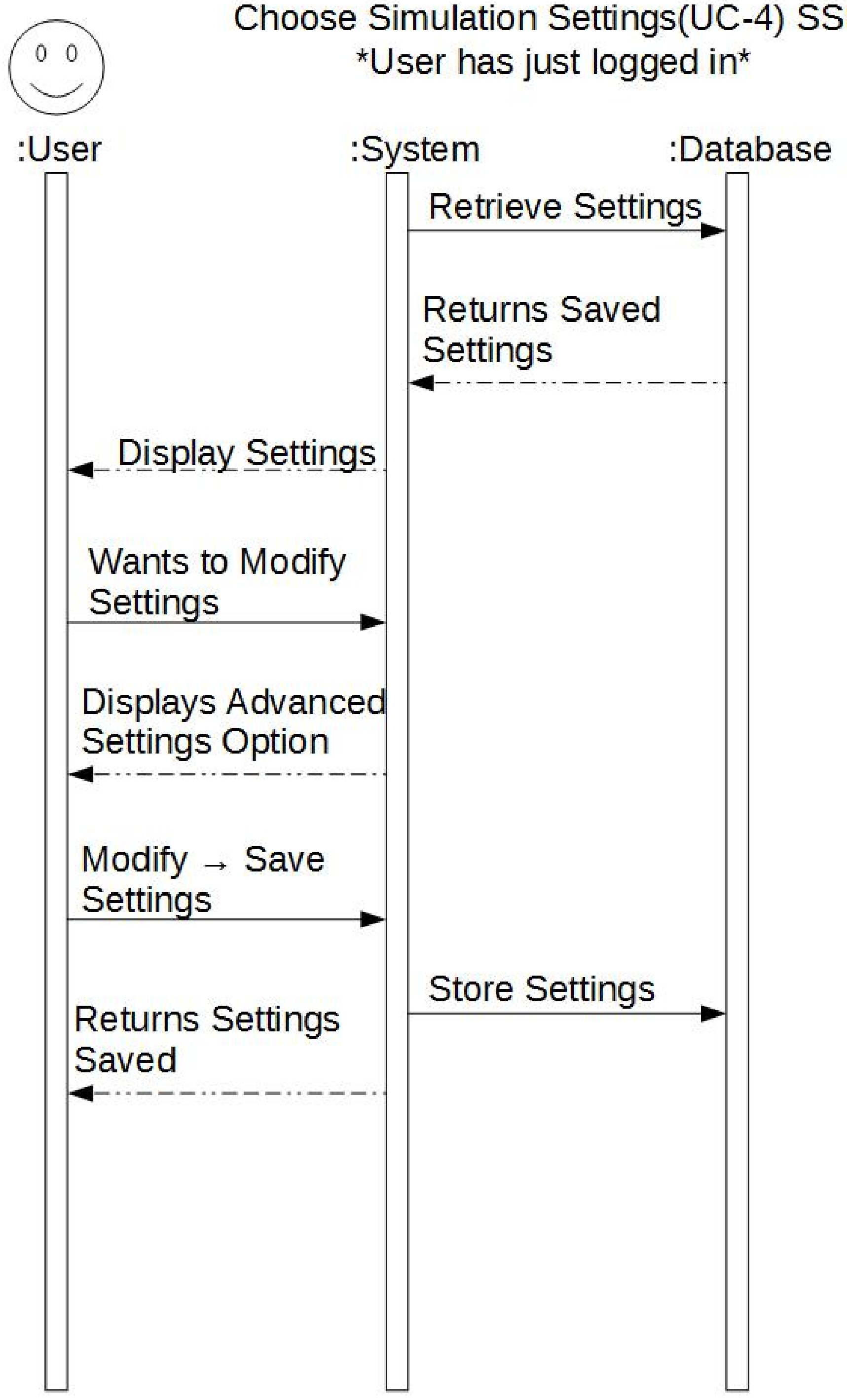
3.d. System State Diagram

Simulate(UC-1) SSD
User has just logged in



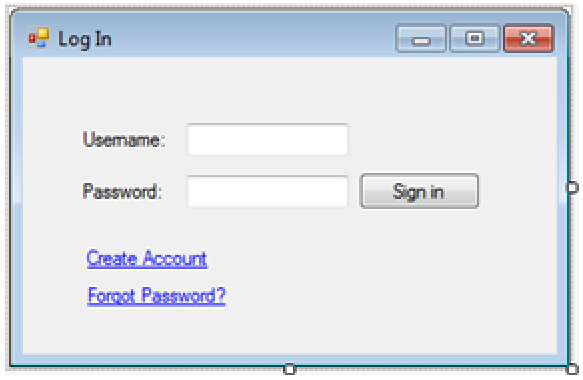
Choose Simulation Settings(UC-4) SSD

User has just logged in



4. a. Preliminary Design

Initial Screen:

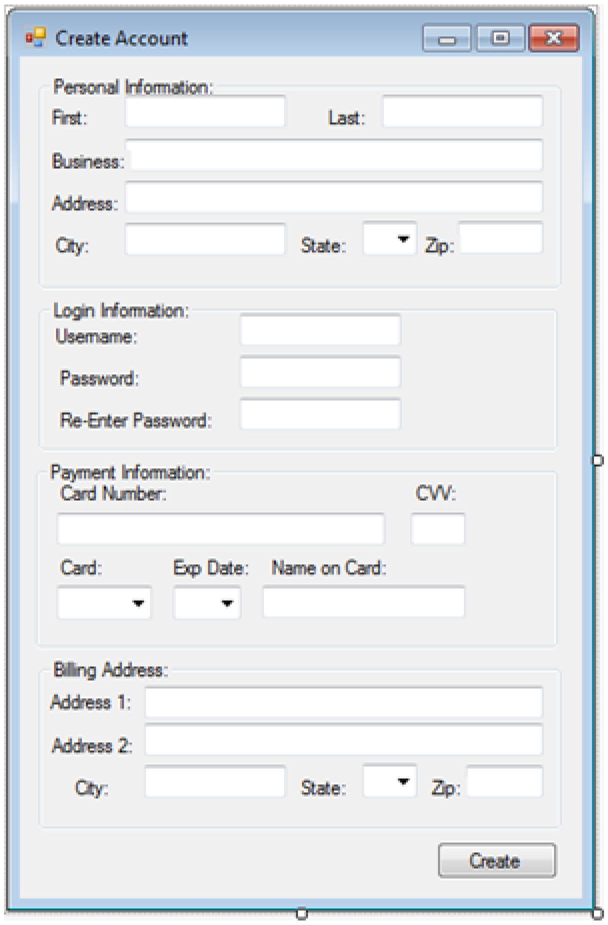


Just like a standard log-in, the user will enter the username (his e-mail) and the password. There are three buttons to click on this initial screen.

1. Sign-In: Will then launch the basic UI Screen (See below)
2. Create Account: Which will launch the Registration UI Screen (see below)
3. Forgot Password: Which will launch standard password recovery (Not shown below)

Registration UI Screen:

On this screen user will enter required data to start his account. Most fields are just plain text boxes, however there are a few check boxes and drop downs for state. You can then click create which will enter you right into the program's Basic UI Screen.



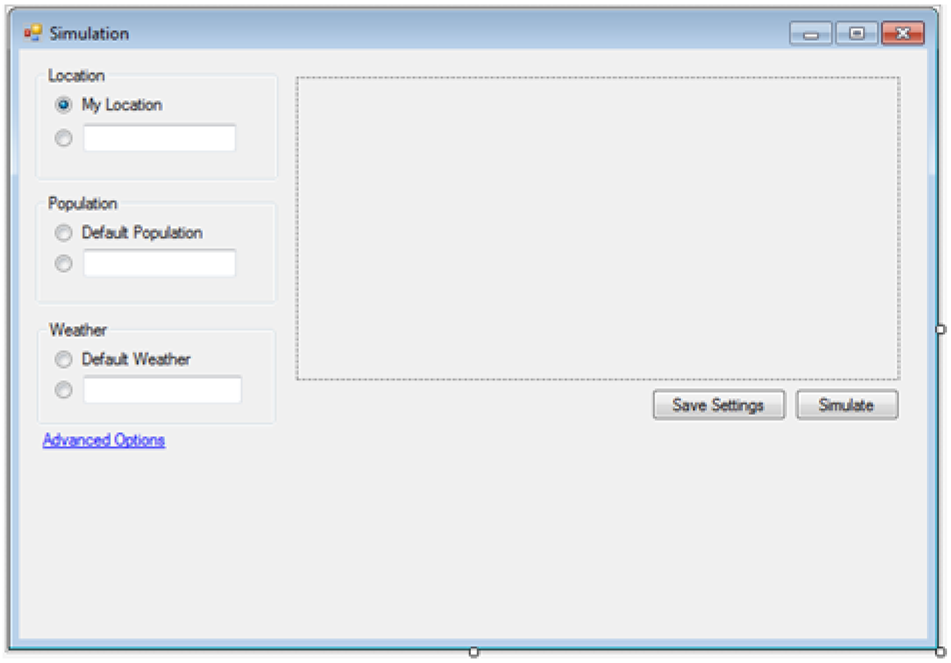
Basic UI:

The basic UI will look like like this. On the left you will have your basic variables you can alter, and on the right will be your simulation graph report. The variables will be always previously set to default values, and are represented as checkbox groups, therefore something will always be selected. This also cuts down on the minimum number of clicks necessary to run the program. Default values are based upon current location or saved settings of the user.
On this you have three action clicks.

Action clicks:

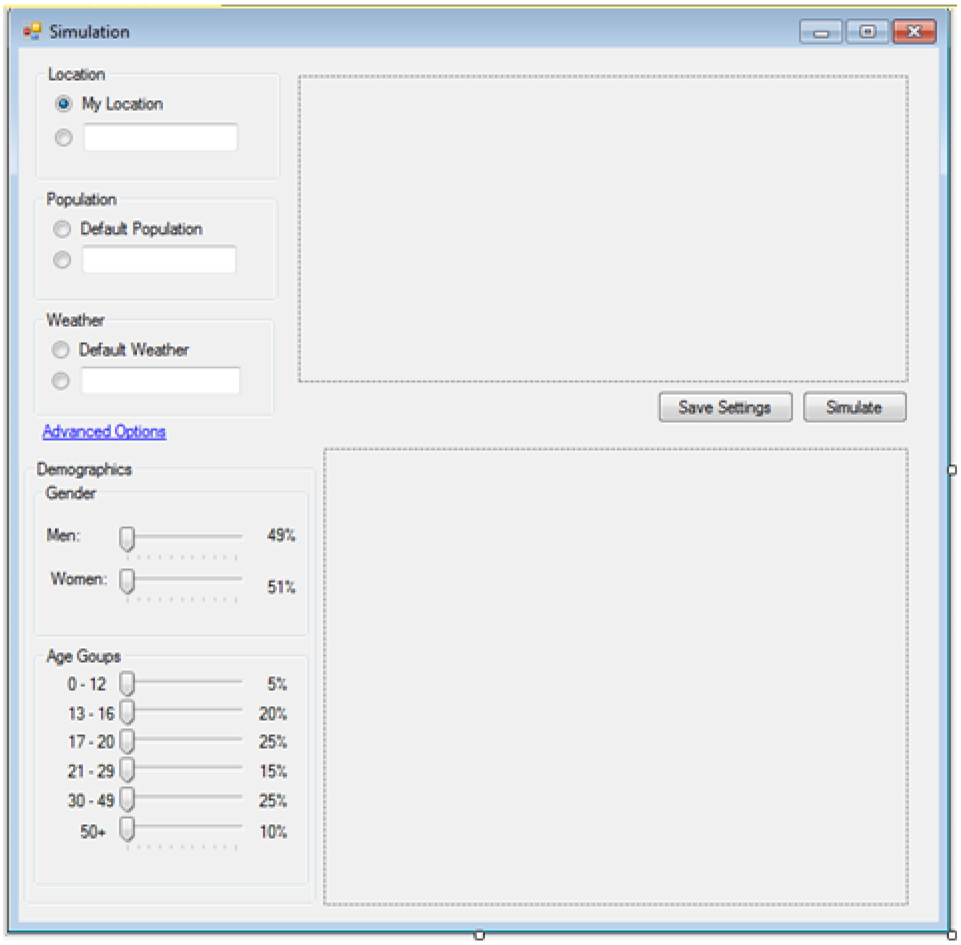
1. Save Settings: This option will take current settings and set them as the default. Once the user logs on they will be automatically selected. If this button is never clicked, default settings will be set as the programs default settings.
2. Simulate: This button refreshes the graph by re-running the simulation inputting the new settings selected. If there is no change it will just re-run the simulation with the same settings selected.
3. Advanced Options: This hyperlinked phrase, will expand the basic UI view to the Advanced UI

(see below).



Advanced UI:

The change in this UI is the addition of population demographics. These are originally set at a default relative to the inputted location. Once this section is opened you are presented with two slider groups and another graphic. The two slider groups will always equal 100%. When sliding the men's slider lower, the women's slider will increase. Same relative to the age groups. Same with the basic, the simulation and the pie chart graphics will update corresponding to the inputted values.



4b. User Effort Estimation

Non-Registered User:

25 Clicks to Register. Includes typing in the initial information in the sign-on screen and then being directed to the registration page.

Registered User:
It will take a registered user 3 clicks to enter the program.

First time basic simulation:

This will take a user 25 clicks to register and enter, and one click to simulate the basic settings. This would go down to 4 clicks if the user has been previously registered.

Running Advanced options:

A new user would take 25 clicks clicks plus 7 clicks to enter and alter all the demographics, and simulate. A registered user would take 3 clicks plus the 7 to set and simulate a determined simulation.

General:

Most clicks in this program are to select a data and run the simulation. Only a few clicks are designated to input data. Especially when a user need to register their account.

5. Domain Analysis

We derive the domain model concept starting from the responsibilities mentioned in the detailed use cases. Following table lists the responsibilities and the assigned concept.

Concept Definitions

The concepts and their responsibilities are discuss below.

Deriving concepts from responsibilities identified in the detailed use cases .

Responsibility Description	Type	Concept Name
Runs the game once user succeeds in inputting correct parameter detailed in UC-1 Main Success Scenario	D	Simulator
If user modifies default conditions, checks if modified condition legal	D	Simulator
A simple and effective display for the user to interact and obtain information from the simulation	D	Interface
Registers Visitor into database and the registered Visitor is now considered a User	K	Database
Stores past simulation results, user registration information and default values	K	Database
Chooses whether to go to the beach or not depending on various conditions set by the user	K	Agents
Decides whether to change decision in the next round based on previous outcomes	K	Agents
Keeps track of how many agents went to the beach for the curre round	K	Beach
Shows default values to the user	D	Interface
Allows the user to select advanced options using a button interfa	D	Interface
Displays all of the advanced options available to be modified by user	D	Interface
Checks if the changes made to the advanced menu are legal	D	Simulator

Displays an error if the changes are not legal	D	Interface
The medium that the user uses to access the simulation online	D	Website
Keeps track of a user’s login information, status, and simulation data	D	User Profile

Association Definitions

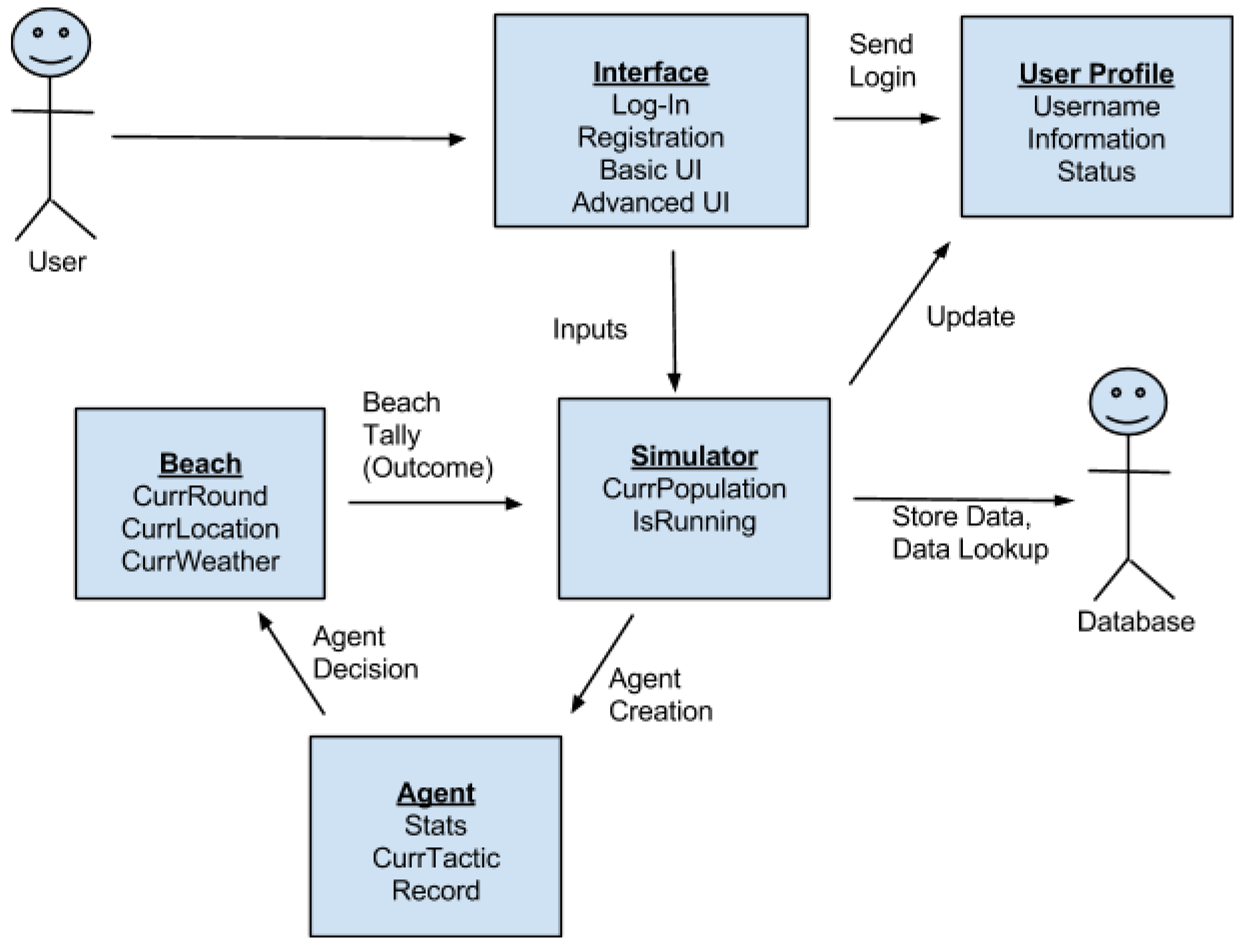
Deriving the association of concepts listed in concept definitions table.

Concept Pair	Association Description	Association Name
Interface↔Simulator	The Interface sends the input ,given by the user, t the simulator	Inputs
Simulator↔Interface	The Simulator sends simulation data to the interface to be displayed to the user	Simulator Updates
Simulator ↔Database	The Simulator sends data to the Database where all the data about each simulation is stored for future use and reference	Store data
Database↔Simulator	The Simulator may use past simulation data store in the Database to influence current simulations	Data Lookup
Simulator↔Agent	Simulator creates a number of agents based on population data	Agent Creation
Agent↔Beach	Agent sends its decision to the Beach whether the are going or not	Agent Decision
Beach↔Simulator	The Beach gives the data of how many went to th beach and how many did not go to the beach	Beach Tally (Outcome)
Interface↔User Profile	The interface sends login information to the user profile	Send login
Simulator↔User profile	Updates the information in the account after each simulation	Update

Attribute Definitions
Deriving the attributes of concepts in concept definitions table from responsibilities identified in detailed use cases.

Concept	Attributes	Attribute Description
Interface	Log-In	Allows users to log in to the website
	Registration	Allows visitors to register for an account
	Basic UI	Contains basic input interface for users that are looking for a quick and easy way to get simulatio data. Also contains the Advanced UI button.
	Advanced UI	Contains the advanced input interface for users looking for more in-depth control of the simulatio
Beach	CurrRound	Keeps track of the current round of simulation
	CurrLocation	Keeps track of the location of the beach
	CurrWeather	Keeps track of the weather at the beach's location
Agent	Stats	Each agent keeps track of their own gender, age group, etc.
	CurrTactic	Keeps track of the agent's current strategy
	Record	Keeps track of the agent's win/loss record
Simulator	CurrPopulation	Keeps track of the current population based on the location given. The number of active agents is bas on the population
	IsRunning	Keeps track of whether the game is running or no
User Profile	Username	Keeps track of a user's login information
	Information	Keeps a record of a user's activity and past simulations
	Status	Keeps track of whether a user is logged in and/or whether they are running a simulation or not

5a. Domain Model Diagram



Traceability Matrix

			Domain Model			
		Interface	Beach	Agent	Simulator	User Profile
Use Case	PW					
UC1	5	X	X	X	X	X
UC2	3	X				X
UC3	3	X				X
UC4	4	X			X	
UC5						
Max PW		5	5	5	5	5
Total PW		15	5	5	9	11

5b. System Operation Contacts

Operation	**Name of operation and parameter**
Cross reference	UC-1
Precondition	User login Input the default values based on the location given.
Postcondition	Data valid, simulate the system. Save the simulation. Store the result.

Operation	**Name of operation and parameter**
Cross reference	UC-1
Precondition	User login Input the default values based on the location give.
Postcondition	If the inputted parameter are outside of range, Data invalid. Simulate does not run.

Operation	**Name of the operation and parameter**
Cross reference	UC-4
Precondition	User log in. User can modify inputted default values based location. User can choose the advance option button.
Postcondition	Data valid, simulate the system. Analysis graphs and visual Demographics. Save the simulation. Store the result..

6. Plan of Work

Plan of Work

- Week of 2/23

Begin to study framework that we will be using for the whole rest of the project.
Everyone should run a few practice .js scripts to get used to the language and how the language works. Each individual group member should explore alternative options that might be available, just in case any of those implementations might be better. The group in charge of algorithms should begin thinking of the initial parameters that will be applied based on the input.

- Week of 3/2

Everyone should now be somewhat proficient with .js so we will begin implementation of the scripts. The group in charge of UI should draw up their final draft of the interface. Based on their draft, the other groups should start writing their own individual drafts for the items included on the UI.

- Week of 3/9

The groups should all have their schedules drafted for what they want to implement by now. Now that they have addressed exactly what they want to implement, the groups should design their own test cases. At this point we should make certain that each groups scripts can be compiled together, and that no one is going along their own route. Any sections that are found to be too difficult should be edited out, and hardcoded, until a reasonable replacement can be found in order to have the whole program implemented in a timely fashion.

- Week of 3/16

At this point the basic structure of the application should be up and running. From here the groups will now work on optimizing their sections of the program and we will meet up to allocated the remaining time to see what else can be implemented successfully. Then we will divide up the new responsibilities and move onto implementing those as well.

- Weeks after

From here on out the goal is to implement “new things” that will make our project stand out from the rest. We will work on what additional elements can be implemented without changing the core of our original implementation.

Product Ownership

Member #	1 Danny Davis	2 Tanzina Farzana	3 Steven Fisher	4 Yang Ren	5 Gegi Oniani	6 Sarid Shinwari	7 Jose Delgado

Project management (10 points)		33%			33%		34%
Sec 1: Customer Statement of Requirements (9 points)	50%		50%				
Sec 2: System Requirements (6 points)	16%	12%	17%	17%	11%	16%	11%
Sec 3: Functional Requirements Spec (30 points)			33%	67%			
Sec 4: User Interface Specs (15 points)	100%						
Sec 5: Domain Analysis (25 points)		34%			33%		33%
Sec 6: Plan of Work (5 points)		20%			15%	50%	15%

*Groups
*Group 1: Yang, Sarid
*Group 2: Danny, Steven
*Group 3: Tanzina, Jose, Gegi
*Notes -- Group 3 took care of their responsibilities timely, while there were difficulties in getting Sarid and Steven to do their parts. As such, the distribution of points should not penalize group 3. Since the distribution favors Yang, and Danny in group 1 and 2 for having to take over their portions.

Work Coordination

The group will meet up once a week to discuss ongoing project requirements and divide up the requirements for that week. However, the workflow will be split into 2 week segments, with the meetings acting as terminals to determine the status of the work for that week.

Technical Workflow

Our web application will be full-stack javascript to minimize development time, iterate faster in an agile environment, and maximize scalability. We will be using the MEAN stack (MongoDB, Express.js, Angular.js and Node.js):

- MongoDB: http://www.mongodb.org/
    - document database (noSQL)
    - Binary JSON format (does queries with JS)
    - Schemaless, indexing support, querying
- Express.js: http://expressjs.com/
    - web framework for Node.js
    - routing, templating
- Angular.js: http://angularjs.org/
    - client MVW framework (binds server to UX)
    - data-binding, syncing, templating, component building, filters, testing
- Node.js: http://nodejs.org/
    - server-side JS platform
    - Built on Chrome's javascript runtime
    - event-driven non-blocking IO

For unit testing:

- karma (http://karma-runner.github.io/0.10/index.html)
    - test runner
- mocha (http://visionmedia.github.io/mocha/)
    - testing framework
- Chai (http://chaijs.com/)
    - assertion library
- Sinon (http://sinonjs.org/)
    - mocking/faking

We will be using git for our version control system. Specifically, it will be hosted at BitBucket. It also contains a wiki, a bug & task tracker, and a code review system that we will be using.

The application will be deployed live on Heroku at http://beachsim.herokuapp.com/

7. References

No outside references were used.