Purpose: The analysis workflow revolves around use cases. What is the point of creating the product if it has no viable business model? What is the intention of the cellular automata simulator? Where will it be the most valuable? How can we design the product to serve high priority use cases?

Functional Modelling (Use case scenarios) 13.4

- **Purpose:** Delineate the role of the CA simulator application using specific examples along with application to real research problems. What is our target clientele and how will we suit the application to them? Describe each scenario in detail citing research or providing reliable demonstration. Also go over the use cases document and create scenarios for each use case.
- Create scenarios for each use case. Both normal scenarios and exceptional ones that may need special handling.
- 4 normal scenarios
- 2 exceptional scenarios

Normal Use Case Scenarios:

1. Basic Simulation Run:

Actor: Undergraduate Biology Student

Scenario:

- Sarah, a 3rd year biology student, opens the CA simulator application on her Windows laptop.
- She selects the "Conway's Game of Life" preset from the available CA models.
- Sarah sets the grid size to 50x50 cells and chooses a random initial configuration.
- She sets the simulation to run for 100 generations with a 0.5 second delay between generations.
- Sarah clicks "Start Simulation" and observes the patterns evolving on the grid.
- Midway through, she pauses the simulation to highlight a particular pattern.
- Sarah then uses the reset function to return the simulation to its initial state.
- Finally, she exports a comprehensive report including all simulation data, parameters, and visualizations for her students to review later.
- Sarah then exports the simulation data as a CSV file for further analysis in her lab report.

This scenario covers the use cases: Select CA Model, Modification and Configuration of Parameters and Cells, Pause/Play Simulation, Visualization Options, Reset Simulation, Simulation Analysis and Export/Import Grid State Results.

2. Custom Rule Creation and Testing:

Actor: Graduate Computer Science Researcher

Scenario:

- Alex, a CS grad student, launches the CA simulator to test a novel cellular automaton rule he developed.
- He selects "Create Custom Rule" and uses the rule editor to input his new transition rules.
- Alex names his rule "AlexCA-1" and saves it to the application's rule library.
- He sets up a 100x100 grid with a specific initial pattern he wants to test.
- Alex configures the simulation to run indefinitely and starts it.
- As the simulation runs, he uses the pause/resume controls to closely examine interesting patterns.
- Alex adjusts some parameters of his rule in real-time and observes the effects.
- Satisfied with his results, he stops the simulation and saves the rule for future use.

This scenario covers the use cases: Create Custom Rule, Present Rules/Scenarios, Configure Simulation Parameters, and Pause/Play Simulation.

3. Collaborative Research Scenario:

Actor: Multiple Researchers

Scenario:

- Dr. Chen, leading a research team, opens the CA simulator on her lab's main computer, which is connected to a large display.
- She loads a custom CA model representing tumor growth that her team has been working on.
- Dr. Chen configures a complex initial state representing actual patient data.
- As she starts the simulation, her team members connect to the application remotely from their own devices.
- The team observes the simulation together, discussing the results in real-time.
- One team member suggests a parameter change, which Dr. Chen implements on the fly.

- They run multiple iterations with different parameters, comparing results.
- At the end of the session, Dr. Chen exports a comprehensive report including all simulation data and parameters for the team's records.

This scenario covers the use cases: Load Saved Configuration, Run Simulation, Collaborate in Real-time, and Generate Reports.

4. AI-Assisted Prediction of CA Behaviour

Actor: Machine Learning Researcher

Scenario:

- David, an AI researcher, wants to train an AI model to predict the future of CA states
- He selects a complex CA model and runs multiple simulations, by pausing, he can collect and record simulation data to feed into his AI model
- David feeds this data into a machine-learning framework, categorizing it into training and testing sets for his machine-learning model. to predict the next steps of the simulation
- After refining the model, David can enhance his model to be faster and predict how the CA simulation might evolve in future steps or under new conditions

This scenario covers the use cases: Pause/Play Simulation, Configure Simulation Parameters, Generate Reports, and Simulation Analysis

Exceptional Use Case Scenarios:

1. System Resource Limitation:

Actor: High School Student

Scenario:

- Tim, a high school student, excitedly opens the CA simulator on his old laptop.
- He selects a complex 3D cellular automaton model and sets an extremely large grid size (1000x1000x1000).
- When Tim tries to start the simulation, the application displays a warning about insufficient system resources.
- The application suggests reducing the grid size or choosing a less complex model.
- Tim reduces the grid to 100x100x100, but the simulation still runs very slowly.

- The application detects the performance issue and offers to switch to a 2D version of the model or further reduce parameters.
- Tim chooses the 2D option and is able to run the simulation successfully.

This scenario covers error handling for system limitations and demonstrates the application's ability to adapt to user constraints.

2. Data Corruption During Export:

Actor: Research Assistant

Scenario:

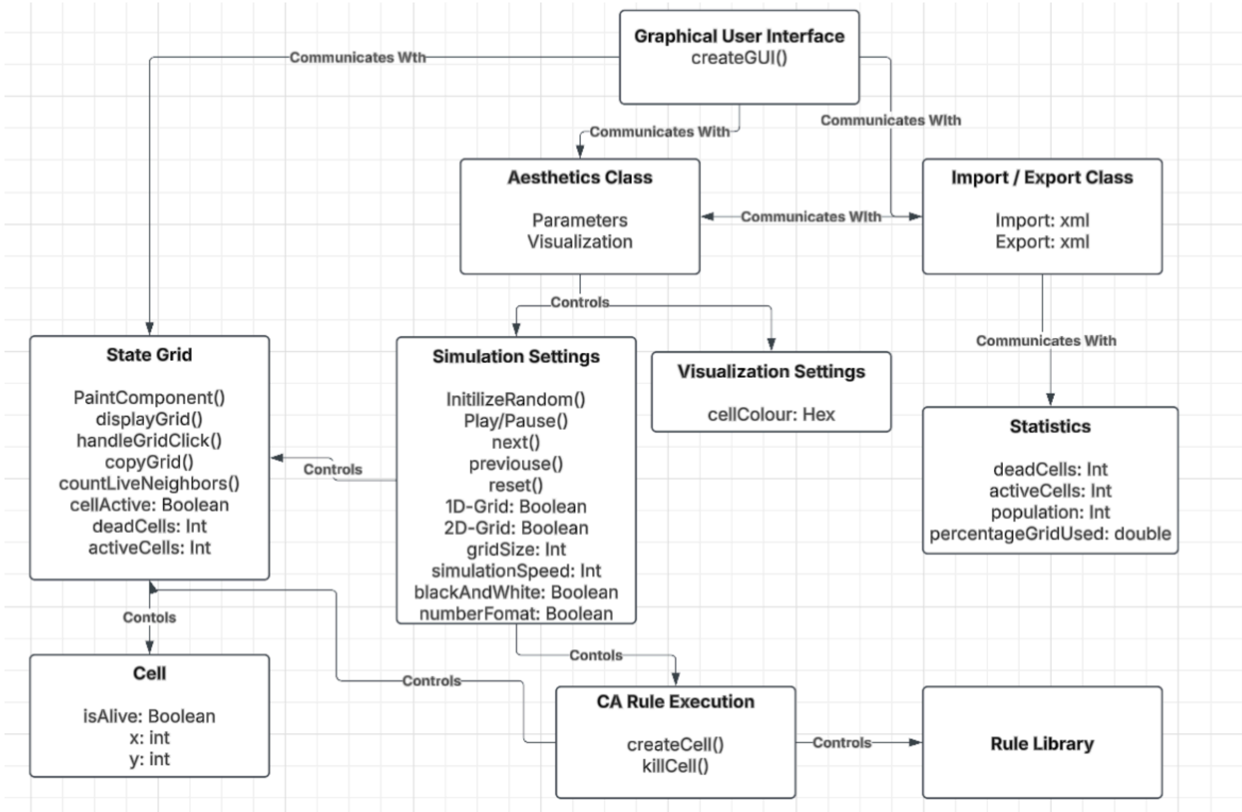
- Lisa, a research assistant, has just completed a long-running simulation of an ecological model.
- She initiates the export process to save the simulation data and results.
- During the export, there's a brief power fluctuation causing a system hiccup.
- The application detects that the exported file is corrupted and immediately notifies Lisa.
- It offers options to retry the export or recover data from the last auto-save point.
- Lisa chooses to recover from the auto-save and successfully exports the data.
- The application then prompts Lisa to send an anonymous error report to help improve future versions.

This scenario demonstrates the application's error handling capabilities, data recovery features, and feedback mechanism for continuous improvement.

Entity Class Modelling 13.5

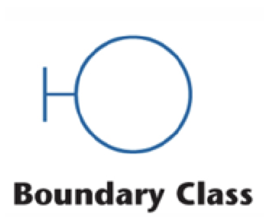
- **Purpose:** Determine the entity classes and their attributes. Then, determine the interrelationships and interactions between the entity classes. Present this information in the form of a class diagram.
- **Entity Class:** An entity class contains information that is persistent for a fairly long duration of time.
- **Boundary Class:** Object that specifies the interaction between the outside world and the internal backend of the product. These are considered I/O or widget classes.
- **Control Class:** The internal backend of the product responsible for most of the processing and sophisticated operation of the application.
- Use the description noun extraction method described in Chapter 13

- Create UML diagrams for the class and their relationships with each other



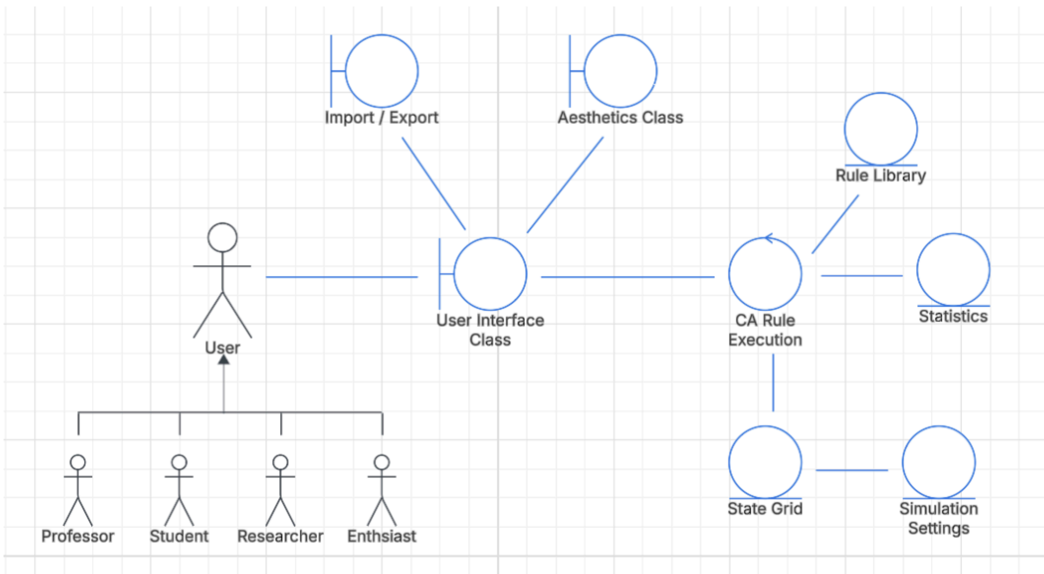
Possible Entity Classes:

eg. Statistic, State grid, Simulation Settings, Rule Library, Cell, Visualization Settings



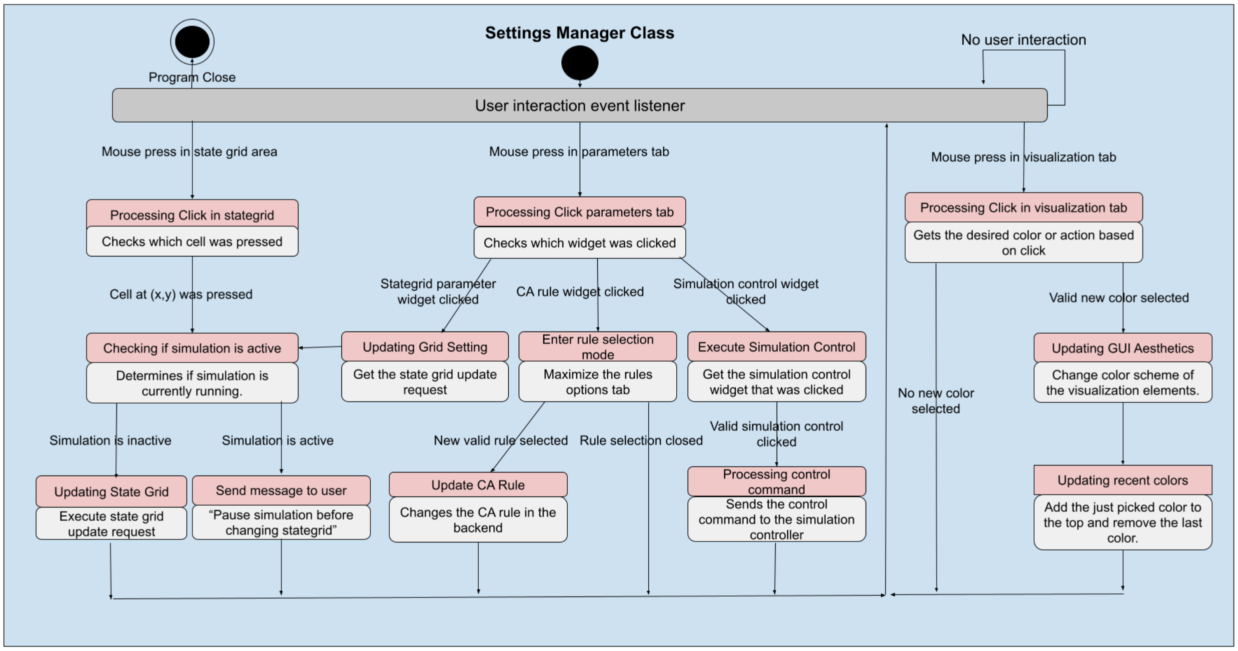
eg. Aesthetics class, Import/Export,

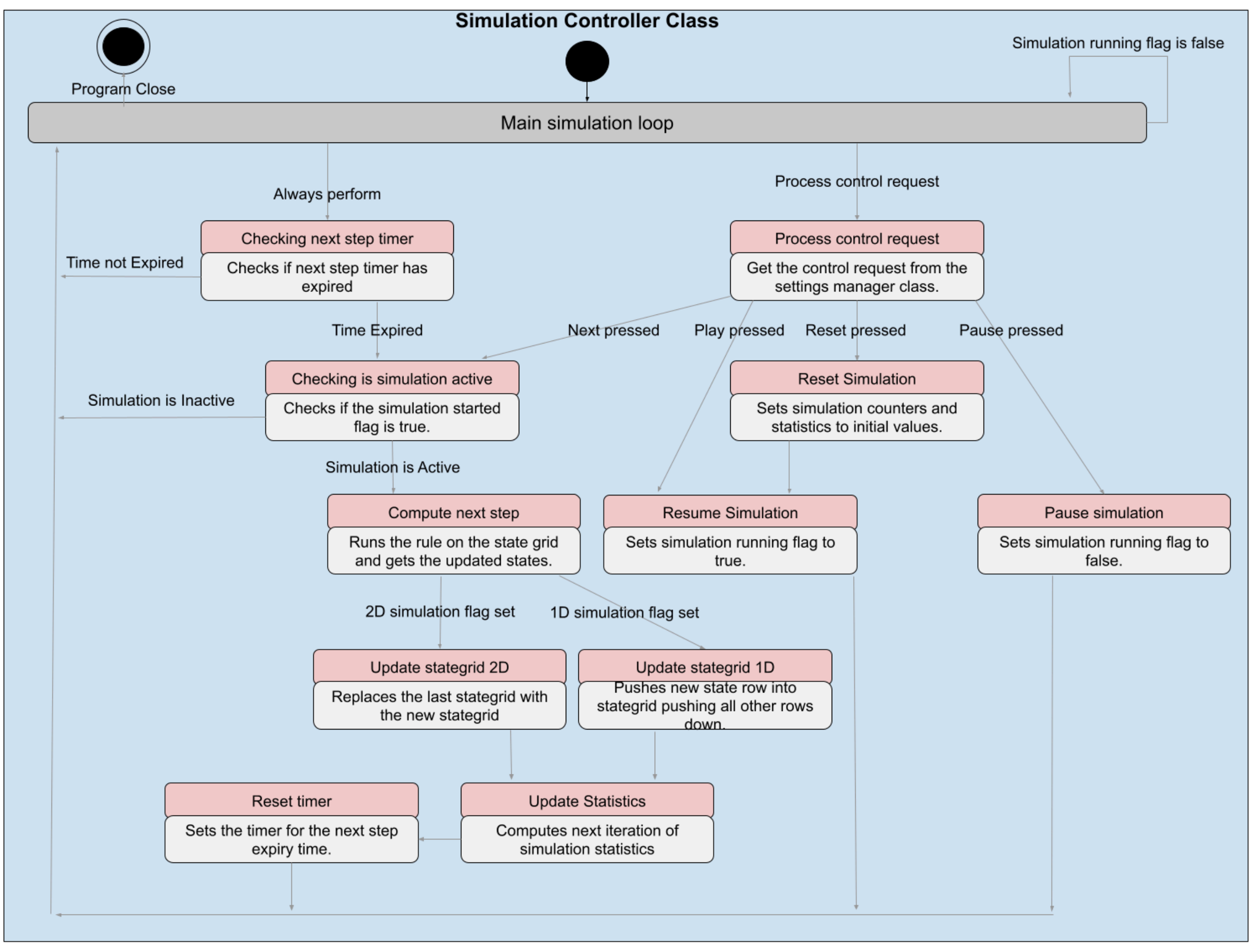
eg. CA rule execution



Statecharts (Dynamic Modelling) 13.6

- **Purpose:** Create a statechart depicting the evolution of the application state throughout operation.
- **Statechart:** A diagram of the state evolution for a given entity class. There are 3 aspects of state evolution, the initial state, event, and predicate which all lead to the next state. These are all organized on the statechart of a given entity class


Settings Manager Class
Program Close
No user interaction
User interaction event listener
Mouse press in state grid area
Mouse press in parameters tab
Mouse press in visualization tab
Processing Click in stategrid
Checks which cell was pressed
Processing Click parameters tab
Checks which widget was clicked
Processing Click in visualization tab
Gets the desired color or action based on click
Cell at (x,y) was pressed
Stategrid parameter widget clicked
CA rule widget clicked
Simulation control widget clicked
Valid new color selected
Checking if simulation is active
Determines if simulation is currently running.
Updating Grid Setting
Get the state grid update request
Enter rule selection mode
Maximize the rules options tab
Execute Simulation Control
Get the simulation control widget that was clicked
Updating GUI Aesthetics
Change color scheme of the visualization elements.
No new color selected
Simulation is inactive
Simulation is active
New valid rule selected
Rule selection closed
Valid simulation control clicked
Updating State Grid
Execute state grid update request
Send message to user
"Pause simulation before changing stategrid"
Update CA Rule
Changes the CA rule in the backend
Processing control command
Sends the control command to the simulation controller
Updating recent colors
Add the just picked color to the top and remove the last color.


Simulation Controller Class
Program Close
Simulation running flag is false
Main simulation loop
Always perform
Process control request
Checking next step timer
Checks if next step timer has expired
Process control request
Get the control request from the settings manager class.
Time not Expired
Time Expired
Next pressed
Play pressed
Reset pressed
Pause pressed
Checking is simulation active
Checks if the simulation started flag is true.
Reset Simulation
Sets simulation counters and statistics to initial values.
Simulation is Inactive
Simulation is Active
Compute next step
Runs the rule on the state grid and gets the updated states.
Resume Simulation
Sets simulation running flag to true.
Pause simulation
Sets simulation running flag to false.
2D simulation flag set
1D simulation flag set
Update stategrid 2D
Replaces the last stategrid with the new stategrid
Update stategrid 1D
Pushes new state row into stategrid pushing all other rows down.
Reset timer
Sets the timer for the next step expiry time.
Update Statistics
Computes next iteration of simulation statistics

Boundary Classes

Within the cellular automata project, there are four main tabs that the user can toggle between to interact with the simulation. These include the Parameters tab, Visualization tab, Statistics tab, and Import/Export tab, each of which has corresponding interfaces that provide different insights and allow users to set various parameters.

The Parameters and Visualization tabs can be considered input screens, as they allow users to define variables for different use cases. For example, the Parameters tab includes options such as Initialize Random, Play/Pause, Next, Previous, Reset, Grid Dimension, Grid Size, Simulation Speed, and Cell Display Format, all of which users can modify.

The Statistics tab serves as an output screen, automatically generating a report for each generation. This report includes information such as the number of dead cells, alive cells, total population, and the percentage of the grid that is occupied.

The Import/Export tab functions as both an input and output screen, as it allows users to import and export data—processes that inherently involve both input and output operations.

Additionally, the State Grid class functions as both an input and output component, allowing users to interact with the grid by selecting cells while also displaying real-time updates as the simulation progresses. This class manages grid rendering, user interactions, and state tracking, making it essential for both visual representation and simulation feedback.

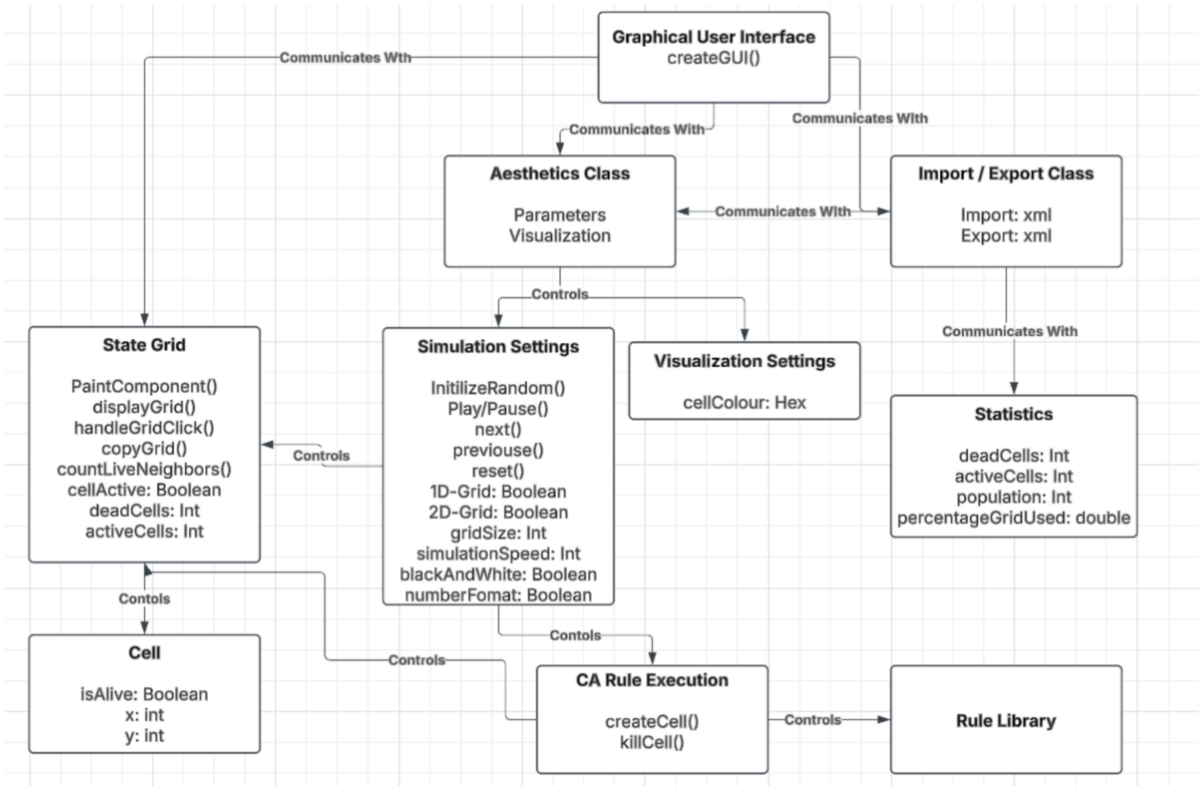
Thus, there are six boundary classes in the project:

- Simulation Settings class
- Visualization class
- Statistics class
- Import/Export class
- State Grid class

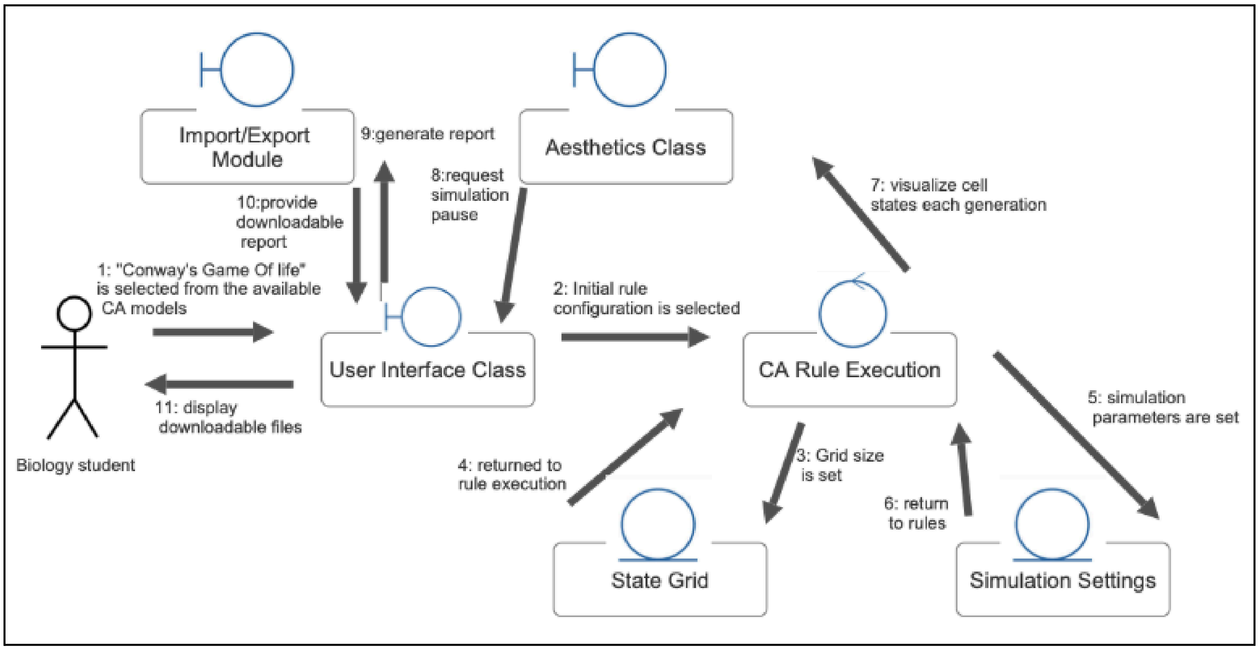
Control Classes

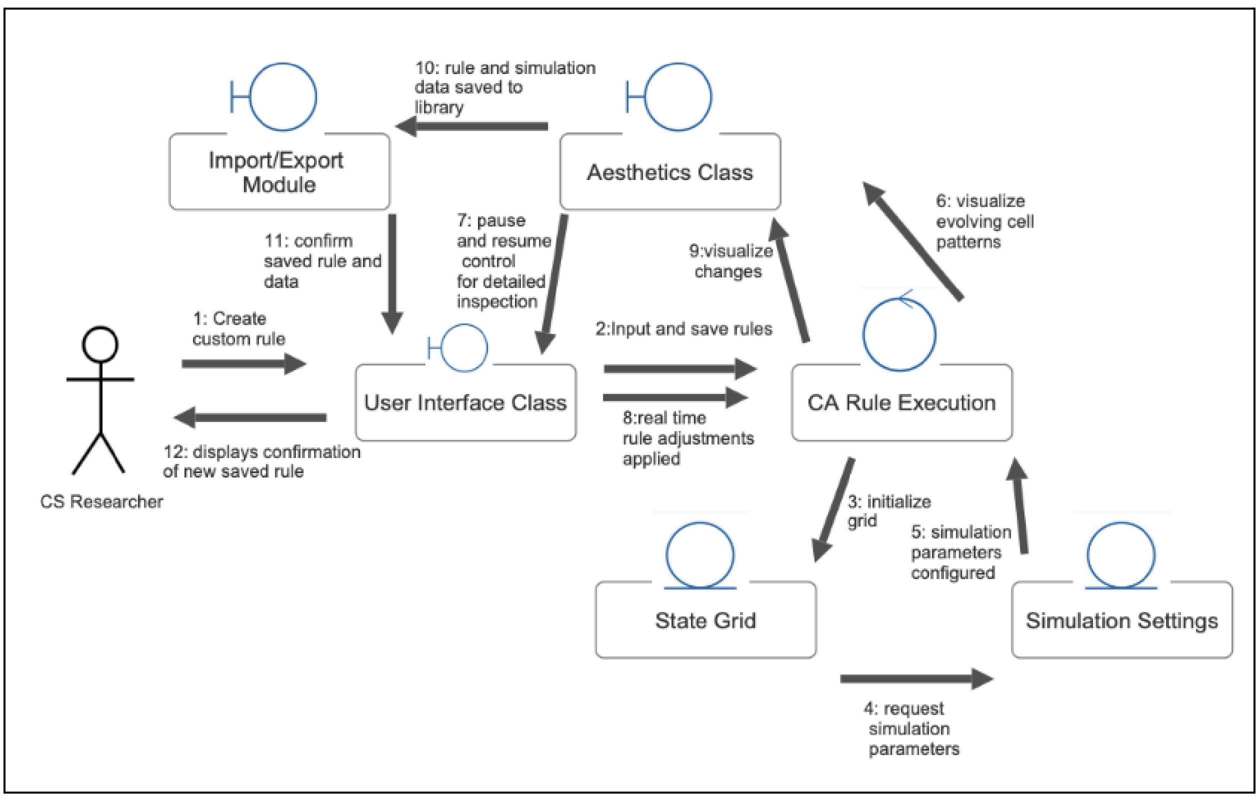
A non-trivial computation is required to generate the next state of the grid, which can be modeled by a CA Rule Execution class. This class is responsible for executing and coordinating the automation steps, with each step being governed by a predefined set of rules.

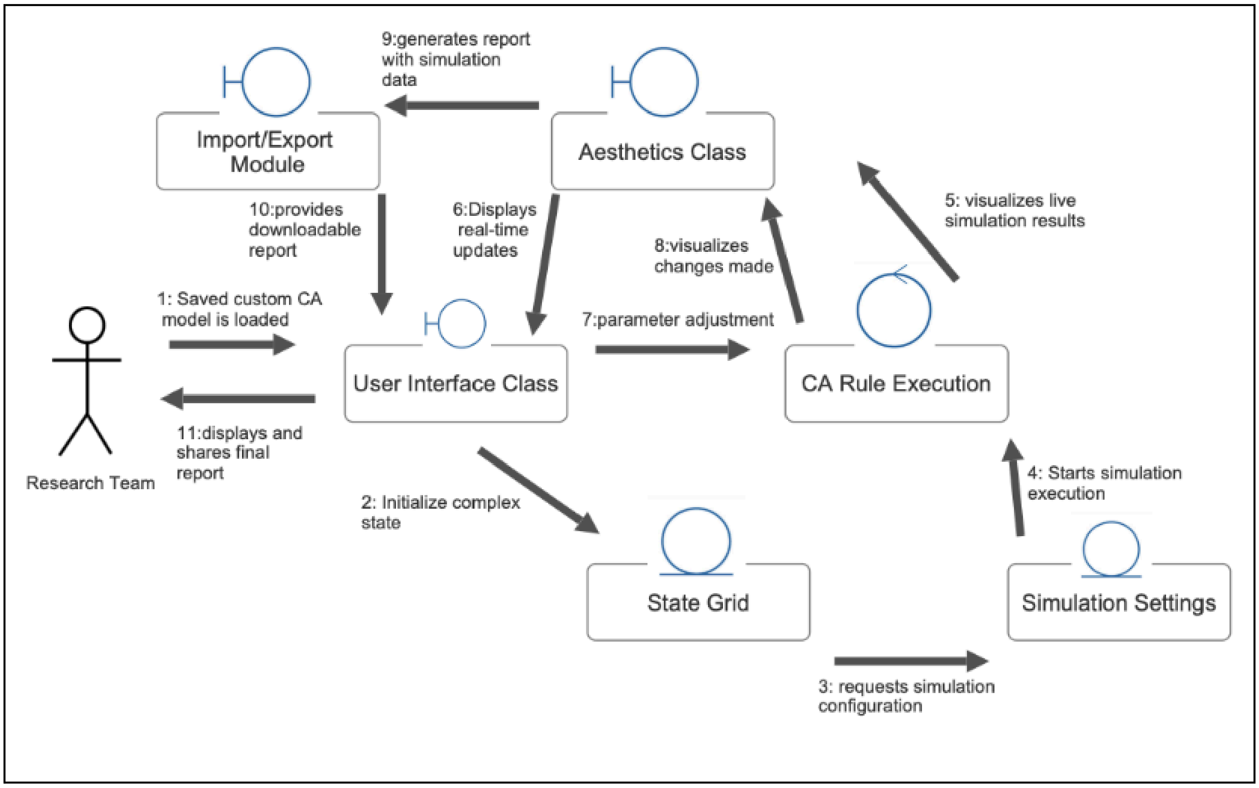
Class Diagram

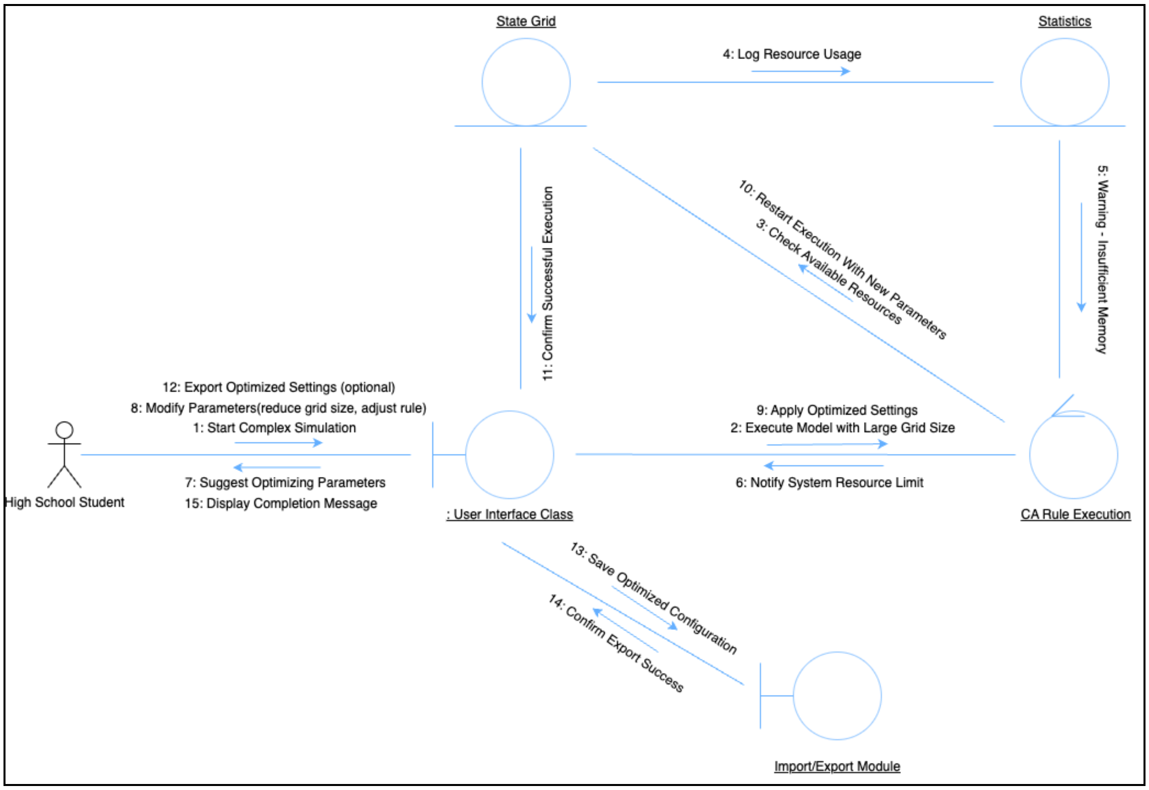


Communication Diagram

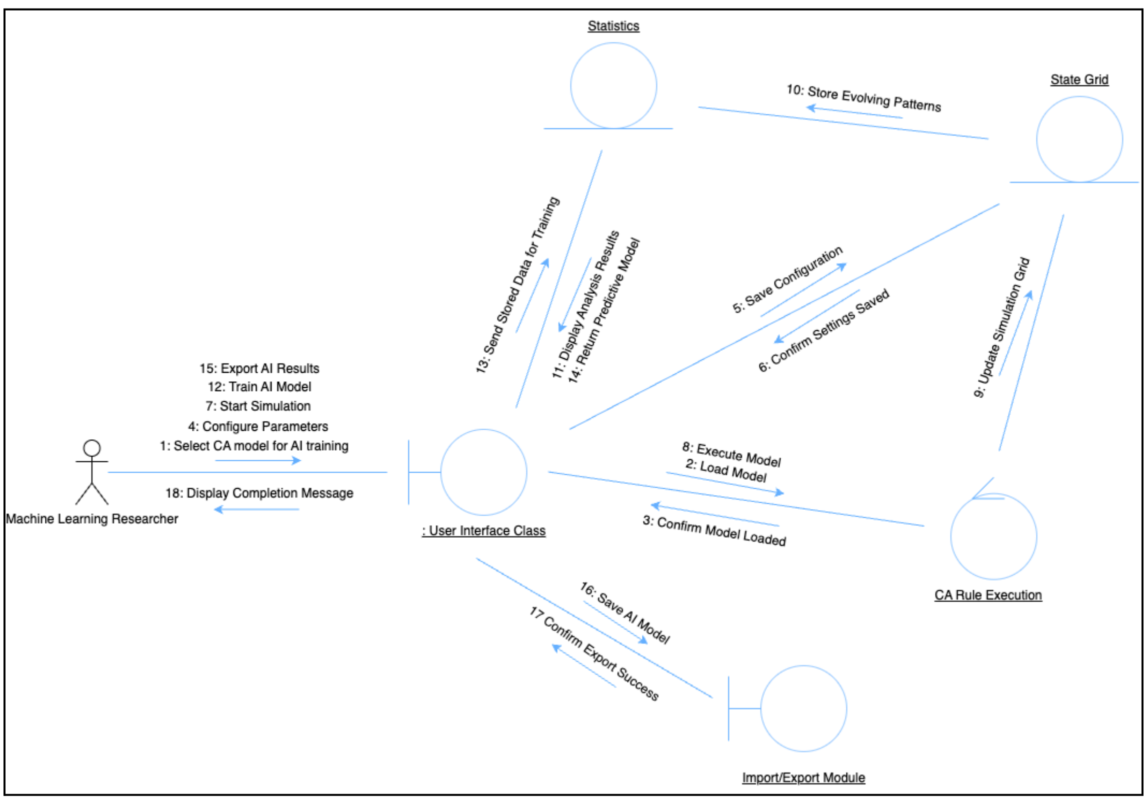




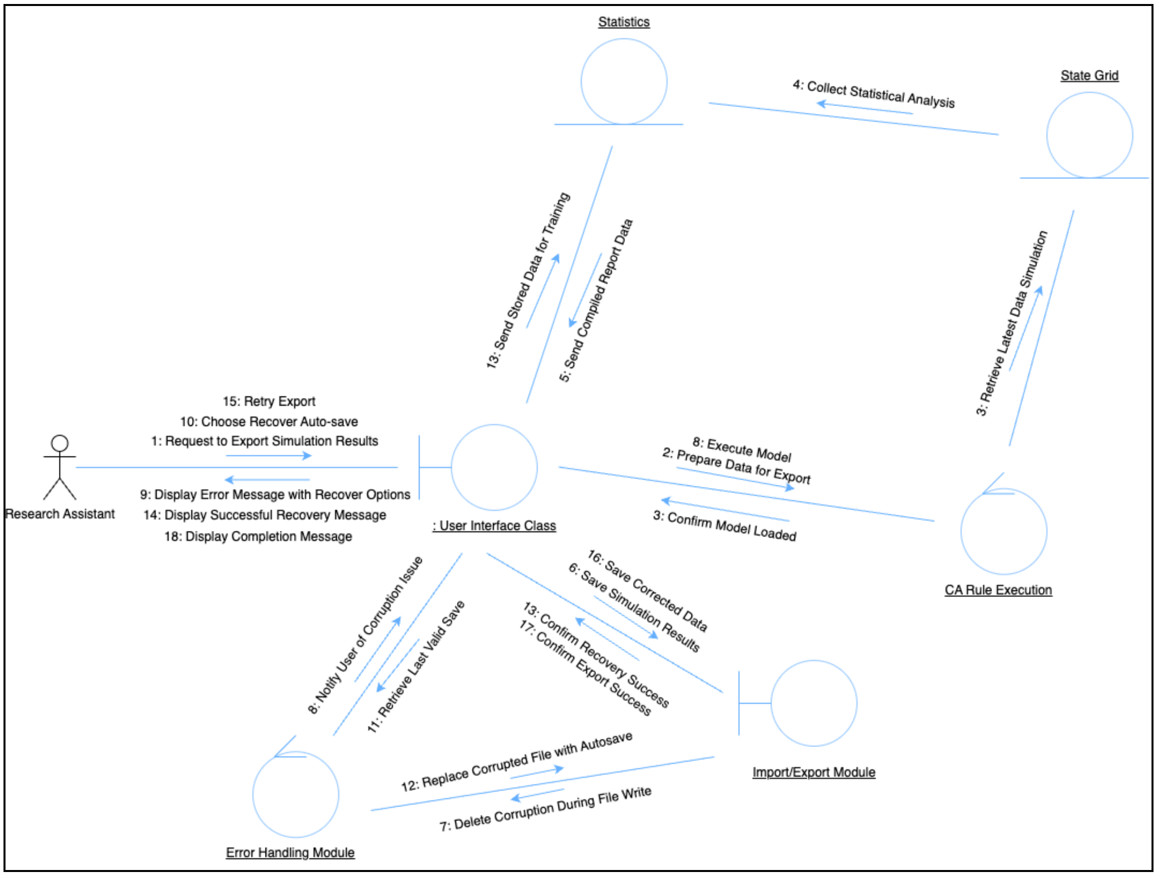

9:generates report with simulation data
Import/Export Module
Aesthetics Class
10:provides downloadable report
6:Displays real-time updates
5: visualizes live simulation results
8:visualizes changes made
1: Saved custom CA model is loaded
7:parameter adjustment
User Interface Class
CA Rule Execution
11:displays and shares final report
Research Team
2: Initialize complex state
4: Starts simulation execution
State Grid
Simulation Settings
3: requests simulation configuration



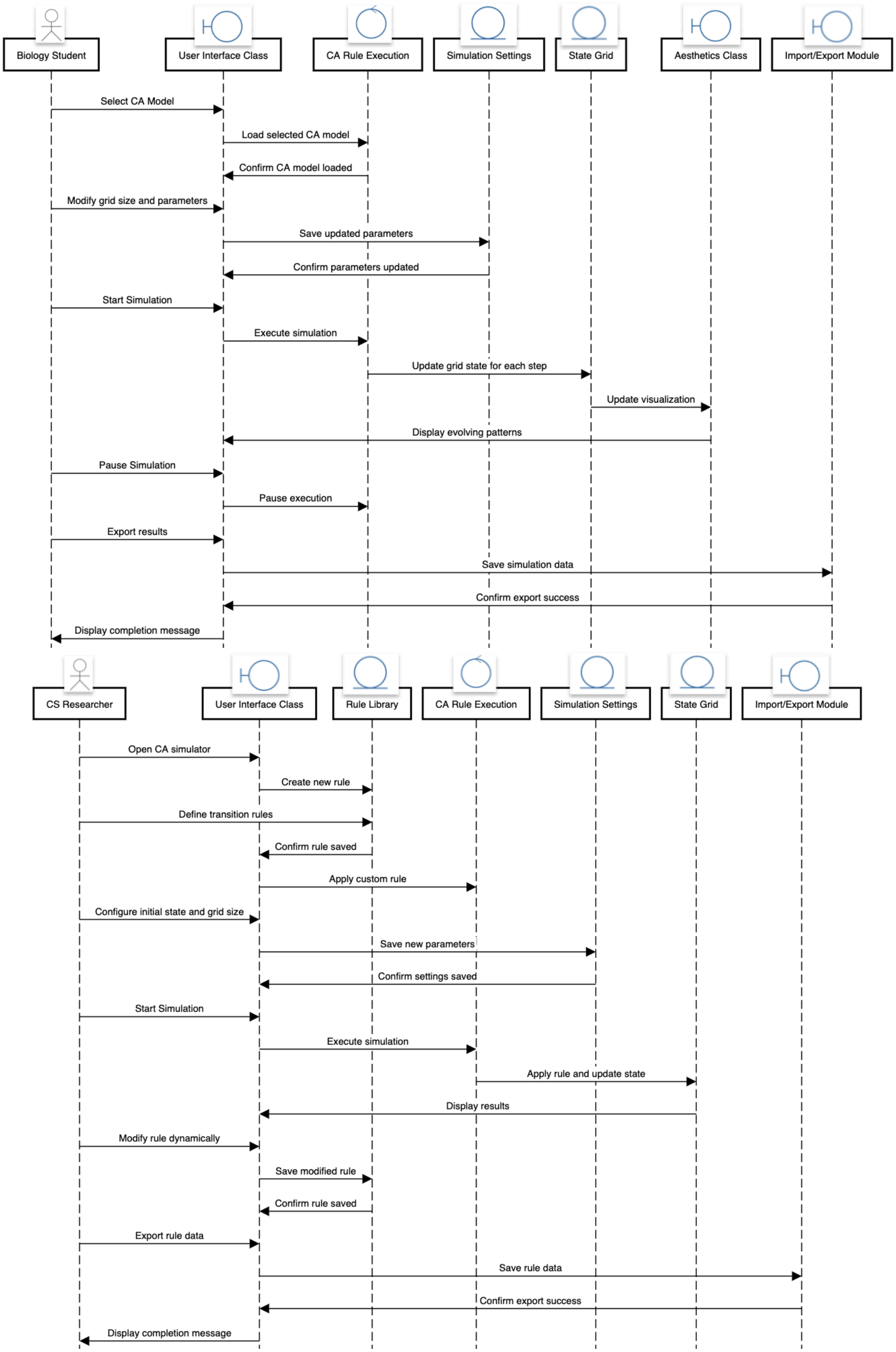
State Grid
Statistics
4: Log Resource Usage
10: Restart Execution With New Parameters
3: Check Available Resources
5: Warning - Insufficient Memory
11: Confirm Successful Execution
12: Export Optimized Settings (optional)
8: Modify Parameters(reduce grid size, adjust rule)
1: Start Complex Simulation
9: Apply Optimized Settings
2: Execute Model with Large Grid Size
7: Suggest Optimizing Parameters
15: Display Completion Message
6: Notify System Resource Limit
High School Student
: User Interface Class
CA Rule Execution
13: Save Optimized Configuration
14: Confirm Export Success
Import/Export Module

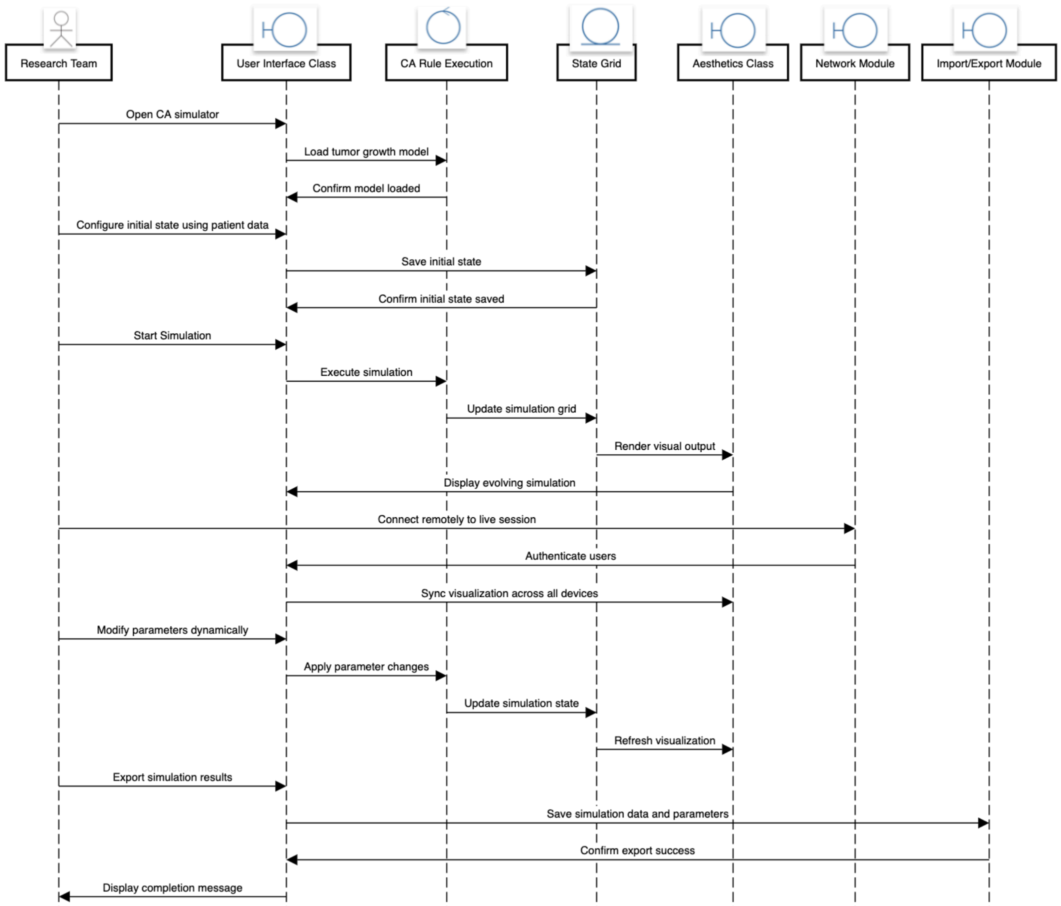


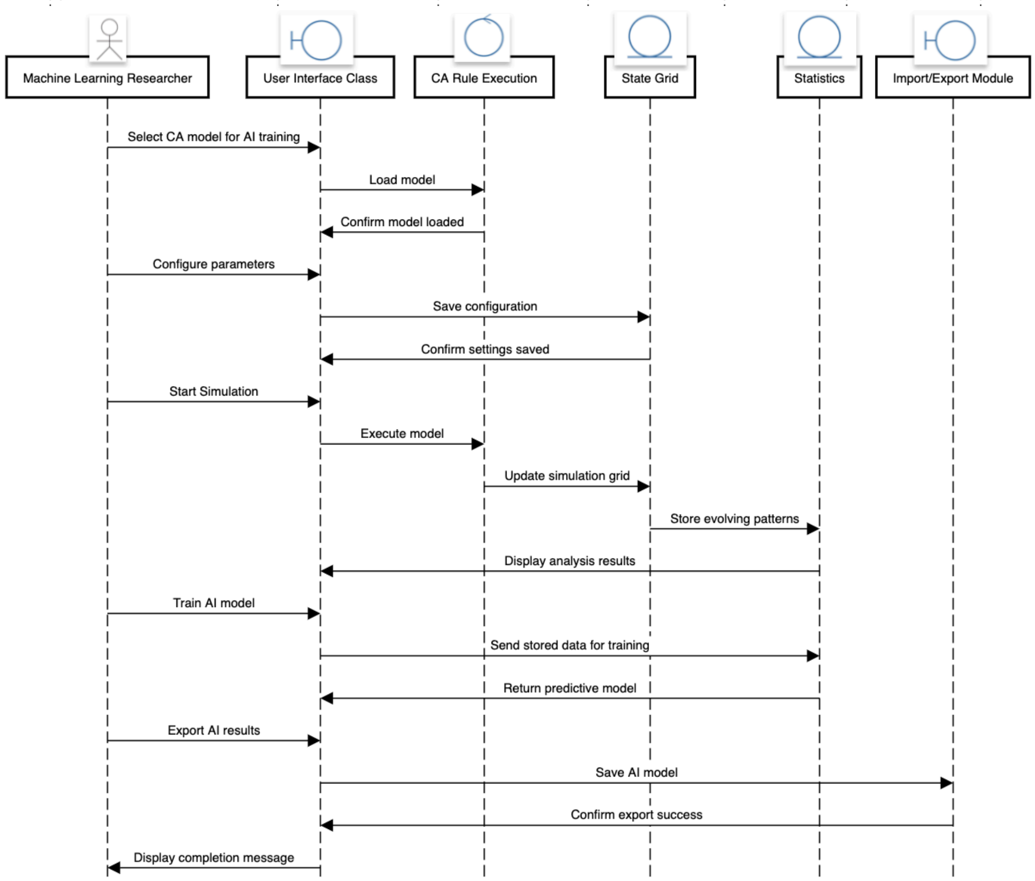
Statistics
10: Store Evolving Patterns
State Grid
13: Send Stored Data for Training
11: Display Analysis Results
14: Return Predictive Model
5: Save Configuration
6: Confirm Settings Saved
9: Update Simulation Grid
15: Export AI Results
12: Train AI Model
7: Start Simulation
4: Configure Parameters
1: Select CA model for AI training
18: Display Completion Message
Machine Learning Researcher
: User Interface Class
8: Execute Model
2: Load Model
3: Confirm Model Loaded
CA Rule Execution
16: Save AI Model
17 Confirm Export Success
Import/Export Module

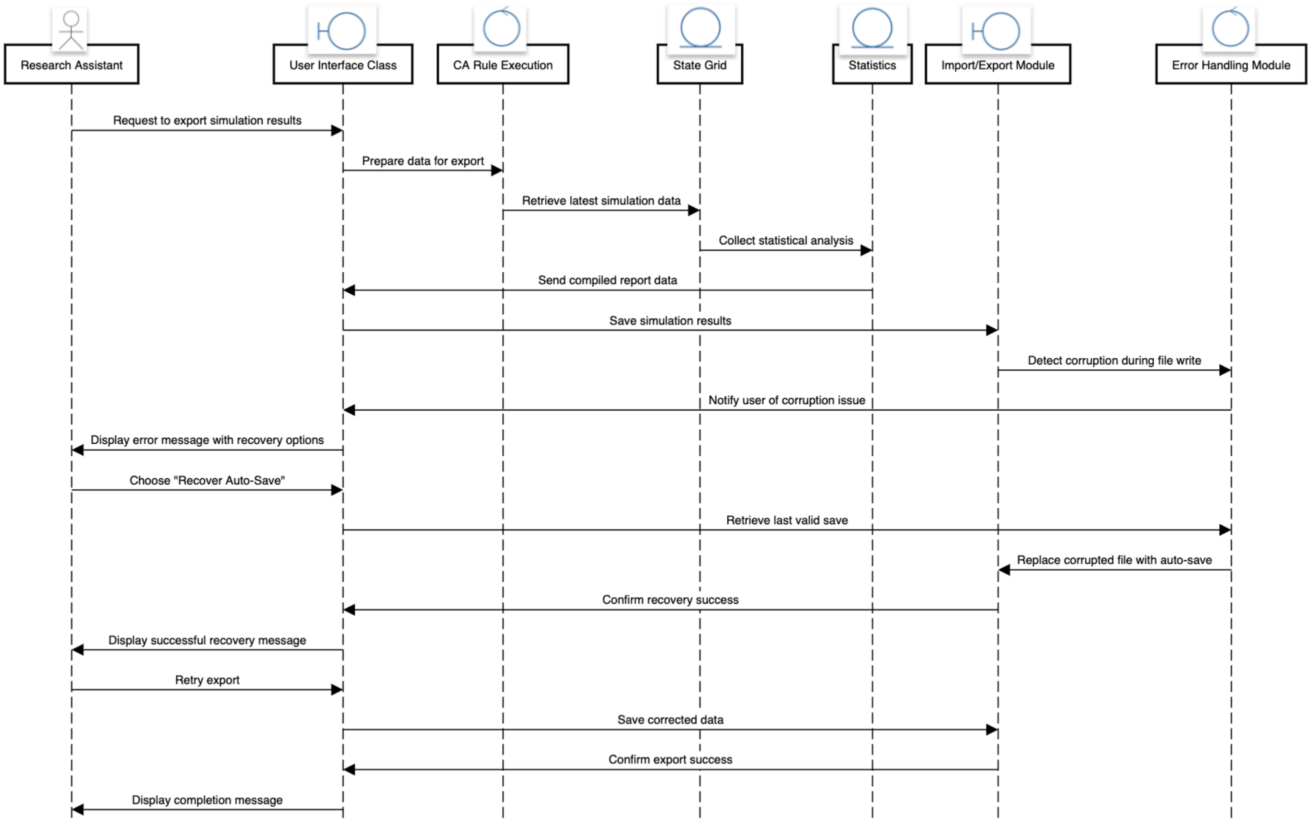

Statistics
State Grid
4: Collect Statistical Analysis
13: Send Stored Data for Training
5: Send Compiled Report Data
3: Retrieve Latest Data Simulation
15: Retry Export
10: Choose Recover Auto-save
1: Request to Export Simulation Results
Research Assistant
9: Display Error Message with Recover Options
14: Display Successful Recovery Message
18: Display Completion Message
: User Interface Class
8: Execute Model
2: Prepare Data for Export
3: Confirm Model Loaded
CA Rule Execution
8: Notify User of Corruption Issue
11: Retrieve Last Valid Save
16: Save Corrected Data
6: Save Simulation Results
13: Confirm Recovery Success
17: Confirm Export Success
Import/Export Module
12: Replace Corrupted File with Autosave
7: Delete Corruption During File Write
Error Handling Module

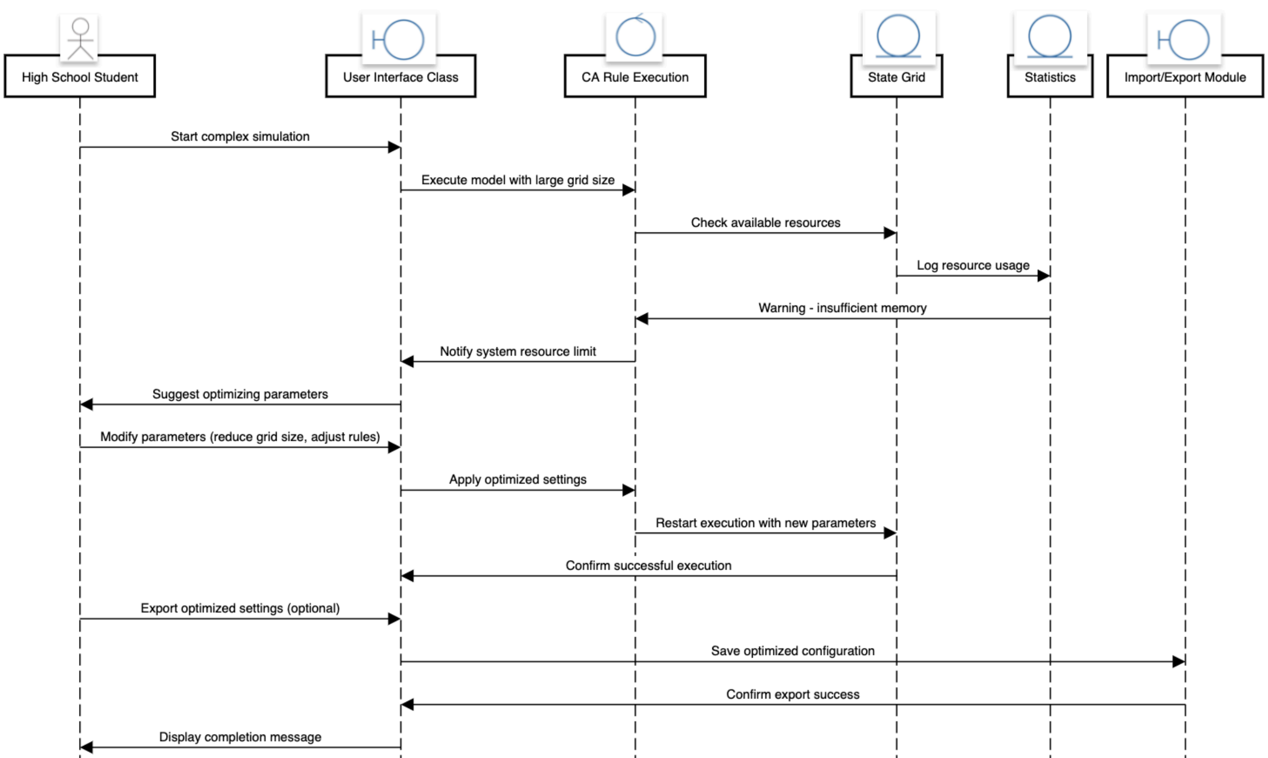
Sequence Diagram




Research Team
User Interface Class
CA Rule Execution
State Grid
Aesthetics Class
Network Module
Import/Export Module
Open CA simulator
Load tumor growth model
Confirm model loaded
Configure initial state using patient data
Save initial state
Confirm initial state saved
Start Simulation
Execute simulation
Update simulation grid
Render visual output
Display evolving simulation
Connect remotely to live session
Authenticate users
Sync visualization across all devices
Modify parameters dynamically
Apply parameter changes
Update simulation state
Refresh visualization
Export simulation results
Save simulation data and parameters
Confirm export success
Display completion message


Machine Learning Researcher
User Interface Class
CA Rule Execution
State Grid
Statistics
Import/Export Module
Select CA model for AI training
Load model
Confirm model loaded
Configure parameters
Save configuration
Confirm settings saved
Start Simulation
Execute model
Update simulation grid
Store evolving patterns
Display analysis results
Train AI model
Send stored data for training
Return predictive model
Export AI results
Save AI model
Confirm export success
Display completion message


Research Assistant
User Interface Class
CA Rule Execution
State Grid
Statistics
Import/Export Module
Error Handling Module
Request to export simulation results
Prepare data for export
Retrieve latest simulation data
Collect statistical analysis
Send compiled report data
Save simulation results
Detect corruption during file write
Notify user of corruption issue
Display error message with recovery options
Choose "Recover Auto-Save"
Retrieve last valid save
Replace corrupted file with auto-save
Confirm recovery success
Display successful recovery message
Retry export
Save corrected data
Confirm export success
Display completion message


High School Student
User Interface Class
CA Rule Execution
State Grid
Statistics
Import/Export Module
Start complex simulation
Execute model with large grid size
Check available resources
Log resource usage
Warning - insufficient memory
Notify system resource limit
Suggest optimizing parameters
Modify parameters (reduce grid size, adjust rules)
Apply optimized settings
Restart execution with new parameters
Confirm successful execution
Export optimized settings (optional)
Save optimized configuration
Confirm export success
Display completion message