Software Requirements Specification Document

Disaster Management Simulation Game

Team Name: *UG3_Team8*

Members: *Madhav, Akash, Kowshik, Jyotiraditya, Aarya, Sanjay*

Brief problem statement

The project addresses the need for a disaster management tool that educates people on how to react during disasters. The system simulates disaster situations where users can experience and practice making decisions in a safe, game-like environment. Incorrect decisions will prompt users to restart from the last successful checkpoint learning better disaster management strategies.

System requirements

- **Technologies [Front-end]**
  - React.js for responsive, dynamic UI
  - Three.js for 3D graphics and animations in simulation
  - Tailwind CSS (optional)
  - Axios for API calls
- **Technologies [Back-end]**
  - Node.js with Express.js for handling API requests
  - MongoDB as database

Users profile

Target Users**:** General public, disaster relief volunteers and students.

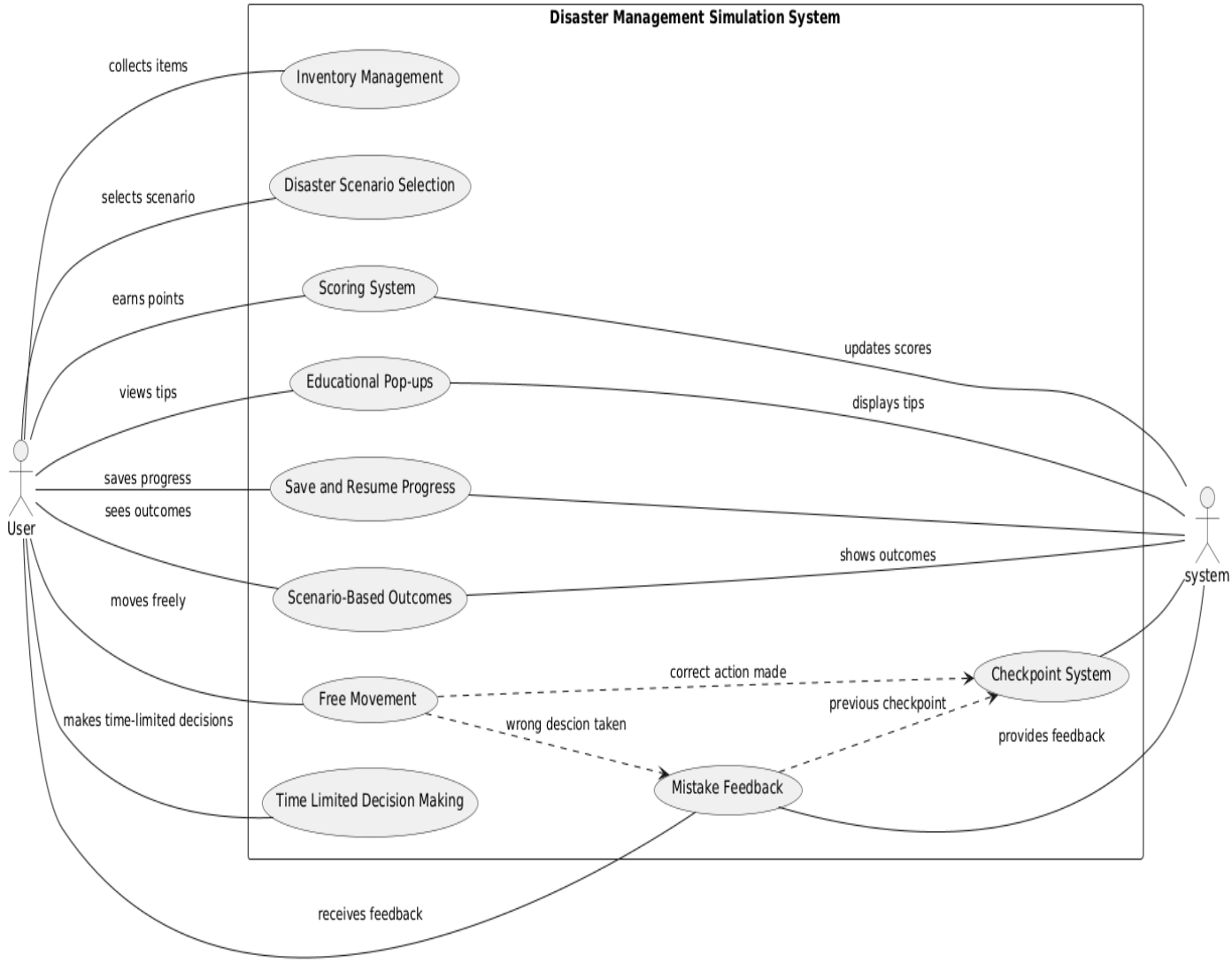
Mode of Interaction**:** Website accessible on both desktop and mobile.

Technical Familiarity**:** Users with basic computer and smartphone usage knowledge, no prior game experience required.

Feature requirements (described using use cases)

No.	Use Case Name	Description	Release
1.	Disaster Scenario Selection	User selects a disaster scenario (e.g., flood, earthquake) to begin learning and interacting.	R1
2.	Free Movement	Users can move freely in the environment, exploring different disaster areas and learning points.	R1
3.	Checkpoint System	When users make a correct decision, they hit a checkpoint that allows them to restart from that point if they fail in the next step.	R1
4.	Mistake Feedback	If the user makes a wrong decision, they are reset to the last checkpoint and given educational feedback.	R1
5.	Scoring System	Users earn points based on correct actions.	R1
6.	Educational Pop-ups	Relevant disaster management tips and resources appear during gameplay.	R1
7.	Save and Resume Progress	Users can save progress and resume later from their last checkpoint.	R2
8.	Scenario-Based Outcomes	Different disaster outcomes based on the user's decisions .	R1
9.	Time limited Decision making	During critical moments, users are required to make decisions within a time limit, simulating real-time pressure in a disaster situation.	R1
10.	Inventory Management	Users collect essential items (e.g., food, water, first aid kits) that could help them survive during the disaster.	R2

Use case diagram :



Use case description

Use Case Number:	UC- 01
Use Case Name:	Disaster Scenario Selection
Overview:	The user selects a specific disaster scenario (e.g., flood, earthquake) from a list to begin interacting with and learning about that disaster type..
Actors:	**User**: The individual selecting a disaster scenario. **System**: The platform or application presenting disaster scenarios.
Pre condition:	The system must display a list of available disaster scenarios, and the user must be authenticated (if required).
Flow:	1. The user opens the application or platform. 2. The system displays a list of disaster scenarios (e.g., flood, earthquake). 3. The user selects a disaster scenario from the list. 4. The system loads the selected disaster scenario and provides relevant information or interactions. 5. The user begins interacting with the scenario.
	Alternate Flows: **Scenario Not Available** 1. The system detects that the selected scenario is unavailable. 2. The user is notified that the scenario is unavailable, and they are prompted to select another one from the list of available scenarios.
Post Condition:	The selected disaster scenario is loaded and ready for user interaction.

Use Case Number:	UC- 02
Use Case Name:	Free Movement
Overview:	The user can move freely within the simulated environment, exploring different disaster areas and interacting with various learning points.
Actors:	**User**: The individual navigating through the environment. **System**: The platform enabling free movement within the disaster scenario.
Pre condition:	The user must have selected a disaster scenario, and the scenario must be fully loaded for exploration.
Flow:	Main (success) Flow: 1. The user enters the selected disaster scenario.

	2. The system enables free movement within the environment. 3. The user moves through the disaster areas, exploring different regions and learning points. 4. The system presents information as the user approaches different areas of learning points. 5. The user completes their exploration and exits the free movement mode.
	Alternate Flows: **User Exits Prematurely** 1. The user decides to exit free movement mode before completing exploration. 2. The system saves the user's progress, and they can re-enter the scenario or continue later.
Post Condition:	The user has freely explored the disaster scenario and interacted with key learning points or areas.

Use Case Number:	UC- 03
Use Case Name:	Checkpoint System
Overview:	When users make a correct decision, they hit a checkpoint that allows them to restart from that point if they fail in the next step.
Actors:	**User**: The individual navigating the scenario and making decisions. **System**: The platform that tracks decisions and manages checkpoints.
Pre condition:	The user must be actively interacting with a disaster scenario, and the system must have checkpoints configured.
Flow:	Main (success) Flow: 1. The user navigates through the disaster scenario and makes a decision at a decision point. 2. The system evaluates the user's decision. 3. If the decision is correct, the system records a checkpoint. 4. The user continues to the next phase of the scenario. 5. If the user makes a wrong decision later, they are reset to the last checkpoint.
	Alternate Flows: **User Skips a Decision Point** 1. The user skips an interaction or decision point without triggering a checkpoint. 2. The user continues through the scenario without a checkpoint, and if they fail, they are returned to the last recorded checkpoint (before the skipped point).
Post Condition:	The user reaches a checkpoint after a correct decision, and the system saves their progress for future restarts if needed.

Use Case Number:	UC- 04
Use Case Name:	Mistake Feedback
Overview:	If the user makes a wrong decision, they are reset to the last checkpoint and given educational feedback.
Actors:	**User**: The individual making decisions in the scenario. **System**: The platform that evaluates decisions and provides feedback.
Pre condition:	The user must have passed at least one checkpoint and be actively interacting with the disaster scenario.
Flow:	Main (success) Flow: 1. The user navigates through the disaster scenario and makes a decision at a decision point. 2. The system evaluates the user's decision. 3. If the decision is wrong, the user is reset to the last checkpoint. 4. The system provides the user with educational feedback explaining the mistake. 5. The user continues from the checkpoint (or the beginning) informed about the mistake.
	Alternate Flows: **User Continues Without Checkpoint** 1. The user makes a wrong decision, but no checkpoint was reached before the error. 2. The system resets the user to the beginning of the scenario instead of a checkpoint. 3. The system provides educational feedback explaining the mistake. 4. The user must restart the scenario from the beginning, losing any progress made. 5. The user proceeds with greater caution, now informed by the feedback provided.
Post Condition:	The user is reset to the last checkpoint or the start of the scenario and given educational feedback to help them understand and correct their mistake for future decision-making.

Use Case Number:	UC- 05
Use Case Name:	Scoring System
Overview:	Users earn points based on correct actions.
Actors:	**User**: The primary actor who makes decisions in the simulation and earns points based on their actions. **Scoring Engine:** Responsible for calculating and updating the user's score based on their decisions and actions during the simulation.
Pre condition:	The simulation engine is running, and the user has started making decisions that can impact their score. The scoring criteria and rules have been defined and are accessible to the scoring engine.
Flow:	1. The user makes a decision during the disaster simulation. 2. The Scoring Engine evaluates the decision based on predefined criteria. 3. If the decision is correct or effective, the user earns points. 4. The Scoring Engine updates the user's score in real time.
	Alternate Flows: **Incorrect Decision** 1. The user makes an incorrect decision during the simulation. 2. The Scoring Engine evaluates the decision and determines that it is ineffective. 3. The user loses points based on the scoring criteria.
Post Condition:	The user's score is updated based on the points earned for correct actions, and the user receives real-time feedback on their performance, encouraging further engagement in the simulation.

Use Case Number:	UC- 06
Use Case Name:	Educational Pop-ups
Overview:	This use case displays educational pop-ups during gameplay, providing users with critical information and tips related to disaster management.
Actors:	User, System
Pre condition:	Users must commit some mistake or reach a checkpoint.

Flow:	1. The user interacts with specific elements in the simulation. 2. The system displays an educational pop-up with information or tips. 3. The user reads the information and can close the pop-up to continue.
Post Condition:	Educational content is successfully displayed to the user, enhancing their knowledge.

Use Case Number:	UC- 07
Use Case Name:	Save and Resume Progress
Overview:	Users can save their progress at any point during the simulation and resume from their last checkpoint later. This feature ensures that users do not lose their progress and can continue learning.
Actors:	User, System
Pre condition:	The user has already been through a disaster simulation.
Flow:	1. The user selects the option to save progress from the menu. 2. The system saves the user's current state and checkpoint. 3. The user exits the simulation and can return later. 4. When the user returns, they select the option to resume. 5. The system restores the user's last saved state.
Alternate Flow:	1. The user attempts to save progress, but the system is unable to save due to a technical error. 2. The system displays a message indicating that the save was unsuccessful. 3. The user is prompted to try saving again.
Post Condition:	The user must continue with saving,without risking loss of progress.

Use Case Number:	UC- 08
Use Case Name:	Scenario-Based Outcomes
Overview:	Different disaster outcomes are based on the user's decisions.
Actors:	**User**: The primary actor making decisions that influence the scenario and its outcomes.

	Simulation Engine: The core system responsible for running the disaster simulation and generating outcomes based on the user's decisions. **Outcome Generator:** Determines the final outcome of the scenario based on the user's decisions, dynamically generating disaster results.
Pre condition:	The disaster scenario has been initialized, and the user has started making decisions. The simulation environment is actively tracking the user's decisions, and the Decision Engine is prepared to evaluate them.
Flow:	1. The user interacts with the disaster simulation. 2. The user faces multiple decision points. 3. The user selects options (e.g., evacuate, secure resources). 4. The Decision Engine evaluates choices in real time, affecting the disaster's progression. 5. Decisions accumulate, shaping the disaster's outcome. 6. The Outcome Generator calculates the final result (e.g., successful evacuation, resource shortages).
	Alternate Flows: **Incorrect Decision** 1. The user makes an incorrect decision during a critical moment. 2. The Decision Engine identifies the negative impact (e.g., failure to secure supplies). 3. The Outcome Generator adjusts the scenario to reflect the negative outcome
Post Condition:	The user completes the simulation, and the final outcome (positive, negative, or neutral) is presented based on the decisions made throughout the scenario.

Use Case Number:	UC- 09
Use Case Name:	Time limited Decision making
Overview:	During critical moments, users are required to make decisions within a time limit, simulating real-time pressure in a disaster situation.
Actors:	**User**: Making critical decisions during the disaster simulation within a limited time frame. **Timer System:** Responsible for displaying the countdown timer to the user, initiating when the decision-making scenario begins, and enforcing the time limit.
Pre condition:	The simulation engine is running, and the Timer System is prepared to initiate the countdown when the critical decision moment begins.
Flow:	1. The user is presented with a critical decision point during the disaster scenario. 2. The Timer System initiates a countdown (e.g., 30 seconds) visible to the user. 3. The user is given multiple choices for how to respond to the critical situation. 4. The user makes a decision within the allowed time.

	5. The Decision Engine evaluates the user's choice based on the simulation rules.
	Alternate Flows: **Decision Made After Time Expires** 1. The user is presented with a critical decision point. 2. The Timer System starts counting down. 3. The user hesitates or takes too long, and the timer reaches zero before a decision is made. 4. The system triggers a default outcome or penalty (e.g., the user fails the scenario, or a negative consequence occurs in the simulation).
Post Condition:	The user successfully makes a decision within the time limit, and the simulation progresses based on the chosen action.

Use Case Number:	UC- 10
Use Case Name:	Inventory Management
Overview:	Users collect essential items (e.g., food, water, first aid kits) that could help them survive during the disaster.
Actors:	**User**: Interacts with the system to collect and manage essential survival items **Inventory System:** The internal system responsible for managing the user's inventory
Pre condition:	The user must be actively participating in a disaster simulation scenario.
Flow:	1. The simulation engine places essential items (e.g., food, water) in the environment. 2. The user navigates and identifies an item to collect. 3. The user interacts with the item to collect it. 4. The system adds the item to the inventory and updates the display. 5. The user continues the simulation with the updated inventory.
	Alternate Flows: **Inventory Full** 1. The system provides feedback to the user indicating that no more items can be collected ("Inventory full"). 2. The user must use, drop, or manage existing items to free up space in the inventory. 3. Once space is available, the user can collect the item.
Post Condition:	The user has successfully collected one or more essential survival items, and the items are now available in the user's inventory for use during the simulation. The inventory system is updated to reflect the newly collected items. The user continues to progress through the simulation with the updated inventory.