CS 2340 – Milestone 3: Project Iteration 1

New Game Configuration, Use Case Diagram, and Domain Modeling

BACKGROUND: You will be designing a remake of the classic strategy video game **Space Trader.** Space Trader, as the name suggests, is a game about traveling and trading in a galactic environment. While Space Trader used a galactic environment backdrop, you may change the theme of the game to a different environment, although core gameplay elements must remain the same.

Within a game of Space Trader, players typically begin the game in a randomly generated galaxy with a small spaceship and a small number of "credits", the currency of the galaxy. Going forward, players are free to explore and trade as they please to acquire more credits, upgrade their ship, and meet new NPCs. Due to the brevity of the course, we will not require you to implement all the features of the original game. Instead, future milestones will detail what new features must be added, with the potential of some extra credit for groups who pursue certain stretch goals, which will also be explicitly detailed.

PURPOSE: There are two primary goals for this project. The first goal is to provide students with experience collaborating with a team of peers in order to develop a product with specific requirements. The second goal is to increase student aptitudes with several key software engineering principles and technologies, namely Git, SOLID, and GRASP. Over the course of the project, these items will be introduced to you so that they can be incorporated into your future milestones.

TASK: For this milestone, you are asked to create two design deliverables, which will be submitted to canvas, in addition to the first portion of your app. Your app implementation/functionality will be graded during a demo, which will occur the week after milestones are due.

Deliverable 1: Use Case Diagram

1. Categorize the following actors as *Primary, Supporting,* or *Off-Stage* (Turn in this and number 2's categorization)
    - Player
    - Game Admin
    - Third-party database
    - Shareholder
    - Price calculation service for market goods
2. Brainstorm one additional actor and categorize it like above
3. Draw a Use Case Diagram for the game application
    - Include the player and at least three other actors
    - Place *Primary Actors* on the left of the system boundary
    - Place *Secondary Actors* on the right of the system boundary
    - Place *Off-Stage Actors* near but unconnected to the system boundary
4. Include six or more *Functional Requirements* within the system boundary

- *Functional Requirements* should define the ways in which *Primary* and *Secondary Actors* may interact
- Example: players configure skill points, admins add additional market items

Deliverable 2: Domain Model

1. Identify and list at least ten potential nouns which could be used in your project
    - Example: player, ship, region
2. Sort the ten nouns into two categories (a *minimum of 5 nouns must be classes)*
    - Game Objects (*classes*): require their own methods and attributes
        - Example: Player, Region
    - Attributes: do not require a whole class
        - Example: Price, Credits
3. Draw a Domain Model for the nouns you brainstormed
4. Connect each *class* within your Domain Model to at least one other class using *associations*
    - Example: Player "travels-to" Region
5. Include *multiplicities* for each association – one on each side of the association

App Implementation Requirements

1. The application should be implemented as a JavaFX desktop application
2. Establish your project in version control using Gatech GitHub.
    - All group members must be contributors on the GitHub repository
3. Display a welcome screen for the application
    - Must include a way to start the game (i.e. a start button)
4. The method of starting the game should take the player to an **initial configuration screen** which has the following requirements:
    - Allows the player to enter a character name
        - **Players should not be allowed to pass in a null or empty name**
    - Allows the player to select a game difficulty from at least three options
    - Allows the player to allocate a set number of skill points to four different skills: *pilot, fighter, merchant,* and *engineer.*
        - Each difficulty should change the number of starting skill points
        - **It should not be possible to allocate more points than the given amount**
    - Provides functionality to allow players to proceed to the next screen
5. Implement a **character sheet screen** which displays the following attributes:
    - Displays the name, difficulty, and skill point allocations
    - Displays the player character's number of credits
        - This starting number of credits should vary by difficulty, like skill points

Checkstyle

During demo your team will be required to run the checkstyle script (located under files>checkstyle>Java Guide.pdf). This script will give your project a score out of 10 and will account for 10 points of your M3 final grade. Be sure to run the checkstyle script prior to submission to avoid unforeseen deductions.

Milestone Tagging

Tags are a way of marking a specific commit and are typically used to mark new versions of software. To do this, use "`git tag`" to list tags and "`git tag –a tag_name –m description`". You are required to tag the latest commit before the deadline which is to be graded during demo. You will be required to pull this commit during demo. Remember, if you have made changes after the submission deadline and demo a later version, you will be subject to a 20 point deduction.

Submission Requirements

In addition to your diagrams, ensure that you include a link to your GitHub repository in your submission. Also, ensure that you have added your grading TA(s) as collaborators so that they may view your private repository. **Repositories must be located on the Georgia Tech GitHub and must be set to private!** Points may be deducted if these guidelines are not followed!

CRITERIA: You will be graded according to the rubrics on the final pages of this assignment document. Please note, 70 points of the 110 total points are dependent on your group's demo. **Groups are required to demo in order to receive credit for the features they have implemented.**

Use Case Diagram Rubric (20 points)

The 5 provided actors are correctly categorized.	5 points. (1 point for each correctly categorized actor)
One additional actor is listed and has a correct category.	1 point.
The Use Case diagram contains the Player and at least 3 other actors.	4 points. (1 point for Player and 1 point for each other actor in the diagram)
Each of the four actors are placed correctly in relation to the system boundary.	4 points. (1 point for each correctly placed actor)
At least 6 Functional Requirements are listed within the system boundary.	6 points. (1 point for each functional requirement)

Domain Model Rubric (20 points)

10 potential nouns are brainstormed and written on a submitted document.	5 points. (1 point for each *reasonable* noun)
Each of the nouns is categorized as a game object or an attribute on the document.	5 points. (1 point for each correctly identified noun)
At least five associations are drawn between classes.	5 points. (1 point for each correctly drawn association)
Each association includes multiplicities (on both sides of the association).	5 points. (1 point for each association with multiplicities on both sides)

Implementation / Demo Rubric (70 points)

A link to the group's project GitHub repository is provided in the Canvas submission.	5 points.
Each group member is a *contributor* (not just a collaborator) on the repository.	5 points. (1 point for each group member that is a contributor)
The application displays a welcome screen and includes functionality for moving to the next screen.	5 points. (3 points for the screen and 2 points for a description of how to proceed)
The functionality for moving to the player configuration screen works properly.	5 points.
It is possible to enter a valid character name within the configuration screen.	5 points. (3 points for a field to enter a name and 2 points for checking for invalid names)
It is possible to select a game difficulty from at least 3 difficulty options.	5 points. (3 points for the first difficulty, 1 for each of the other two)
It is possible to allocate a skill points to four different skills.	5 points. (2 points for 1 skill, 1 point for each of the other three)
Each difficulty should modify the starting number of skill points.	5 points.
It is not possible to allocate more skill points than the starting amount.	5 points.
The player configuration screen includes functionality to proceed to the character sheet screen.	5 points.
The character sheet screen displays character attributes and a starting number of credits.	5 points. (3 points for the character attributes, 2 points for the credits)
The starting number of credits is variable based upon the selected difficulty.	5 points.
Checkstyle Script	10 points