Project: The Calculator Mod

Project Report: Technical Documentation

Professor: Dr. Mike Mireku Kwakye

CSCI441: Software Engineering

December 31th. 2025

Project website: https://spaces.w3schools.com/space/the-calculator-mod/editor

Team D Group Members:

Alexander Redinger

Thol Ucca Kool

Oziel Martinez

Uong SovanDara

Table of Contents

1. Introduction

An architectural and design overview of the project is given in this publication. UML diagrams and data representations are used to illustrate the fundamental system structure, object relationships, and component interactions. grasp how the system functions and how each part adds to its overall functionality requires a grasp of these diagrams.

2. System Overview

Our project is designed to simplify the balancing and management of units and factions for a strategy game called Total War Room 2: ROME. The system enables users to edit, compare, and manage unit data stored in JSON format. It uses a modular structure, where each core part of the program handles a specific function such as loading data, validating input, or managing unit information.

3. Use Case Diagram

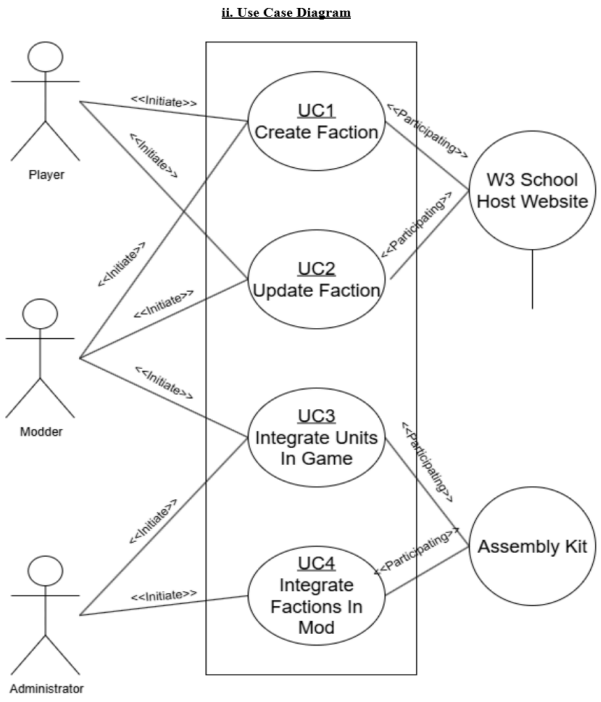

Figure 1. Use Case Diagram for the TWR2 Calculator Mod System

Description: The primary user system interactions are depicted in the use case diagram. It demonstrates how the platform is used by players, modders, and administrators to manage faction integration within mods, create and maintain factions, and include units into the game. These activities are supported by external systems that offer hosting and game integration features, such as the Assembly Kit and the W3 School Host Website.

4. Sequence Diagram

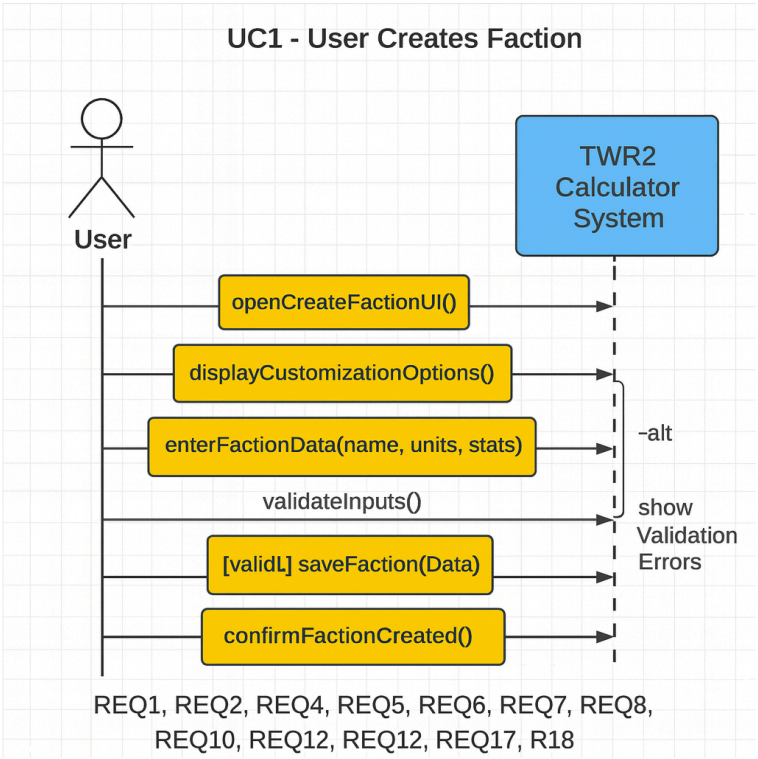

Figure 2. Sequence Diagram for UC1: Create Faction

Description: The sequence diagram illustrates how the user interacts with the TWR2 Calculator System to create a new faction. It shows each step in the process, from opening the creation interface and entering faction details to validating inputs and saving the new faction. The diagram highlights the validation flow, ensuring that incorrect inputs trigger error messages, while valid data is stored successfully, confirming faction creation.

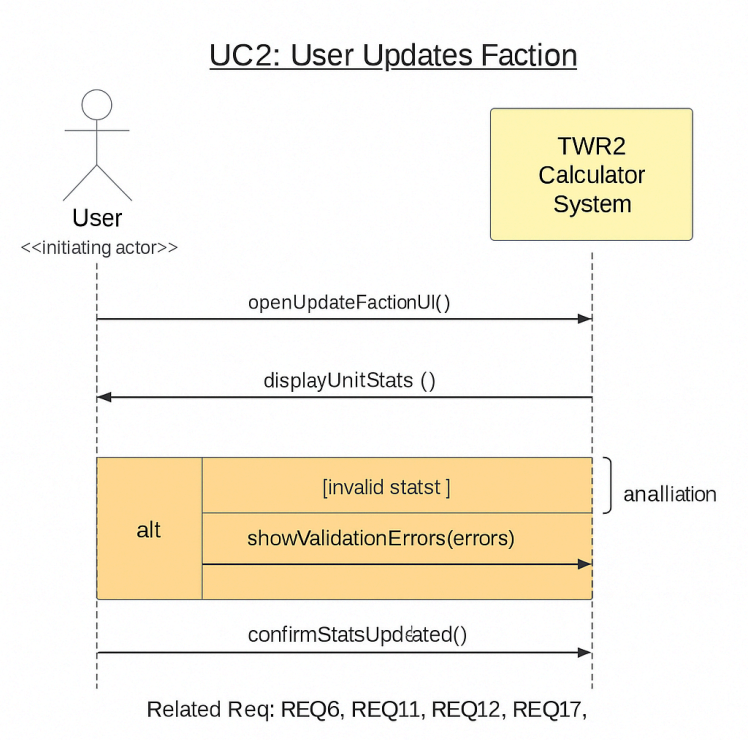

Figure 3. Sequence Diagram for UC2: Update Faction

Description: The sequence diagram illustrates the process of a user updating an existing faction in the TWR2 Calculator System. It shows how the user accesses the update interface, views the current unit stats, and modifies data as needed. The system validates the updated inputs—displaying error messages if invalid stats are entered—and confirms the update once all changes are valid. This ensures that faction data remains consistent and accurate across all updates.

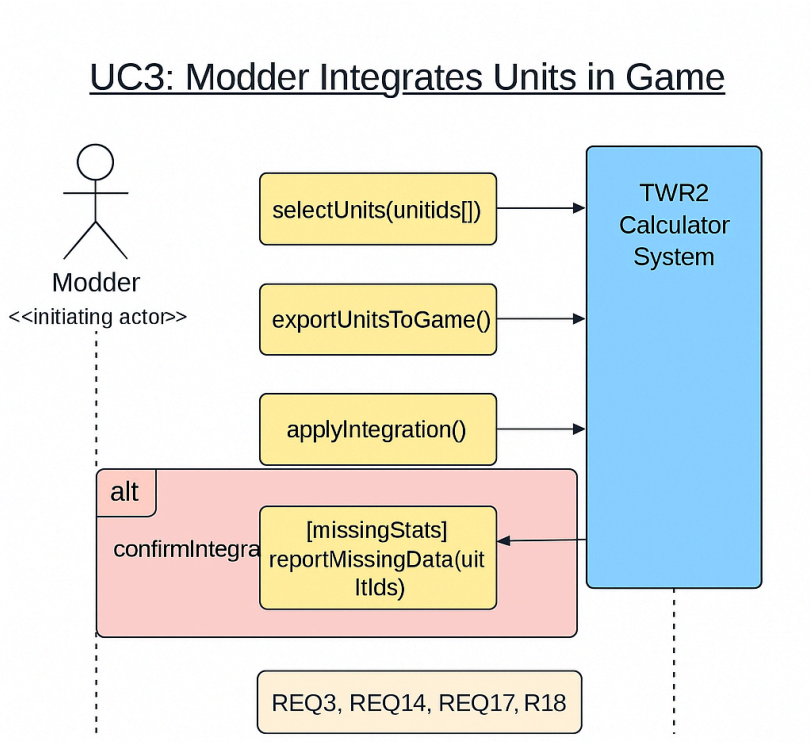

Figure 4. Sequence Diagram for UC3: Integrate Units in Game

Description: The sequence diagram illustrates how a modder integrates selected units into the game using the TWR2 Calculator System. The process begins when the modder selects specific units, exports them to the game, and applies the integration. The system checks for any missing data or stats before confirming the integration. If any unit information is incomplete, the system reports the missing details, ensuring that all integrated units are properly validated and ready for gameplay.

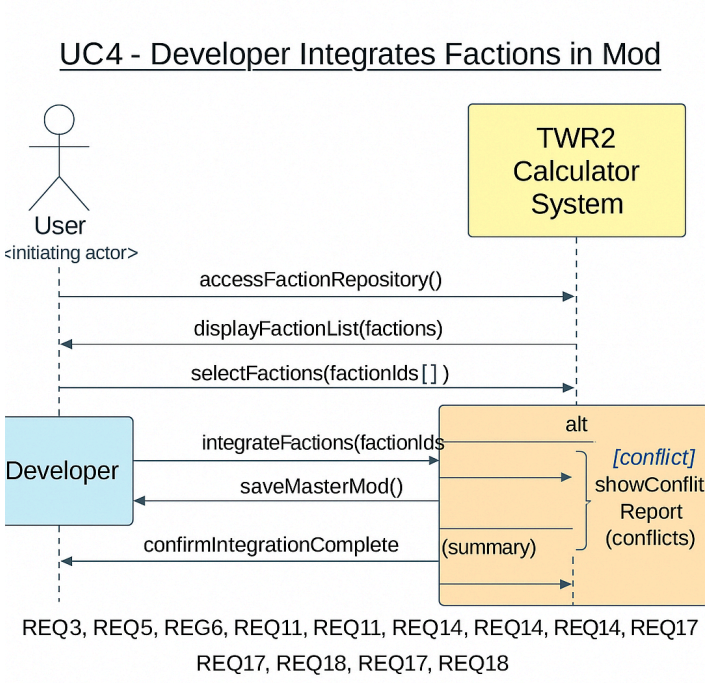

Figure 5. Sequence Diagram for UC4: Integrate Factions in Mod

Description: The sequence diagram demonstrates how a developer integrates multiple factions into a mod using the TWR2 Calculator System. The developer begins by accessing the faction repository, viewing available factions, and selecting which ones to integrate. The system then processes the selected factions, checks for any data conflicts, and generates a conflict report if necessary. Once all integrations are verified, the final mod is saved, and the system confirms successful integration, ensuring all factions work together without overlap or data issues.

5. Domain Model Diagram

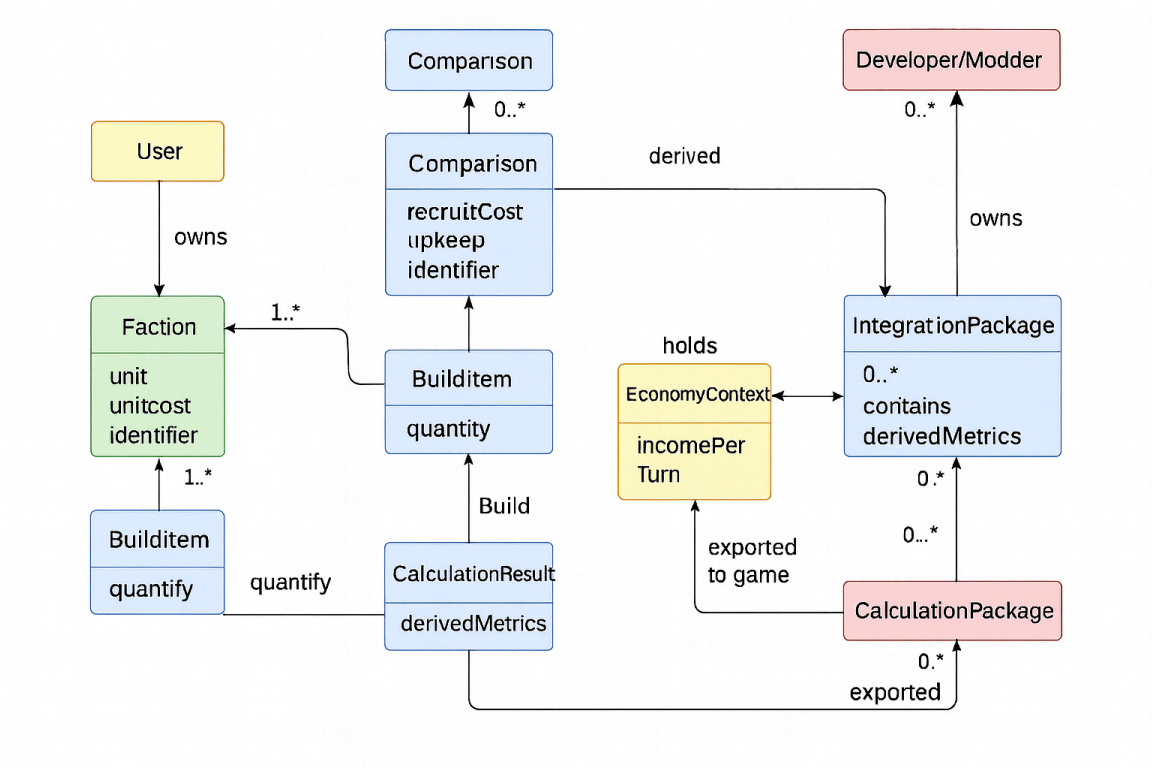

Figure 6. Domain Model for the TWR2 Calculator Mod System

Description: The domain model represents the core entities and their relationships within the TWR2 Calculator Mod System. It outlines how users, factions, and build items connect to form the foundation of gameplay data. Users own one or more factions, each containing multiple build items identified by cost and unit attributes. Calculation results and comparisons are derived from these items to evaluate performance and balance. Developers or modders manage integration packages that bundle these results and export them as calculation packages for in-game use. This model defines how data moves logically through the system, ensuring clear structure between user input, analysis, and integration.

6. Class Diagram

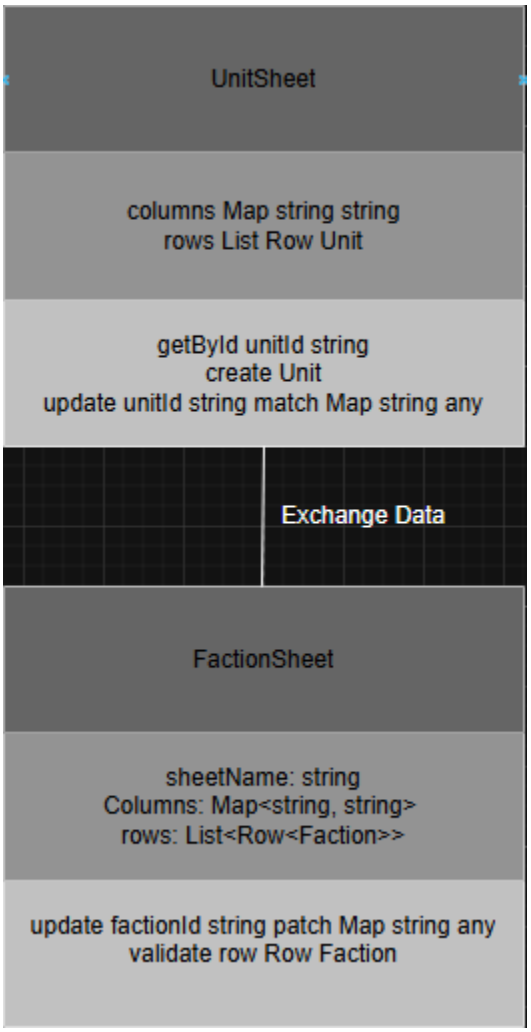

Figure 7. Class Diagram (Overview of Core Classes)

The class diagram illustrates the relationship between the main classes, UnitSheet and FactionSheet, which handle the management of unit and faction data respectively.

The UnitSheet class manages a list of units and their related data. It contains attributes for storing column definitions and row data, and provides operations such as getById, create, and update to handle unit creation and modification based on specific identifiers and parameters.

The FactionSheet class organizes and validates faction-level data. It stores attributes such as the sheet name, column mappings, and rows of faction data. Its methods, including update and validate, ensure that faction data is consistent and correctly formatted when updated.

The “Exchange Data” layer between the two classes indicates a shared data interaction, where updates made in either class can exchange or synchronize information, ensuring that units and factions remain aligned within the system.

7. Data Model

Our system uses JSON files to store and manage data instead of a traditional database. Each JSON file contains structured data representing units, factions, and stats.


```
{
  "unit_id": "infantry_basic",
  "name": "Infantry",
  "cost": 250,
  "damage": 40,
  "armor": 15,
  "faction": "Empire"
}
```


Description: This lightweight structure allows quick loading, easy editing, and fast exporting without complex database setup. It was chosen for its simplicity, flexibility, and ease of integration.

8. Summary

The TWR2 Calculator Mod System's logical and structural framework is described in this technical documentation. The system keeps its architecture structured and flexible by using JSON-based data management and UML diagrams. This method guarantees maintainability, facilitates seamless updates, and makes it simple to incorporate upcoming features like ability editing and unit visual customization.