

**Project: The Calculator Mod**

**Project Report#1 Part 2**

**Professor:** Dr. Mike Mireku Kwakye

**CSCI441:**Software Engineering

Sept 13. 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

### **Individual Contributions Breakdown**

Every member in this group has contributed equally to the project. This is a breakdown of contributions in a table:

Sections	Uong Sovandara	Oziel Martinez	Thol Ucca Kool	Alexander Redinger
Cover Page				
Individual Contribution Breakdown				
Table of Content				
Work Assignment				
1a. Problem Statement				
1b. Decomposition into Sub-Problems				
1c. Glossary of Terms				
2a. Business Goals				
2b. Enumerated Functional Requirements				
2c. Enumerated Nonfunctional Requirements				
2d. User interface Requirements				
3a. Stakeholders				
3b. Actors and Goals				
3c. Use Cases				
3d. System Sequence Diagrams				
4a. Preliminary Design				
4b. User Effort Estimation				
5a. Identifying Subsystems				
5b. Architecture Styles				
5c. Mapping Subsystems to Hardware				

5d. Connectors and Network Protocols			Yellow	
5e. Global Control Flow				Green
5f. Hardware Requirements		Red		
Project Management	Blue			
Reference	Blue			

## Table of Contents

<u>Section</u>	<u>Page #</u>
Cover Page.....	1
Individual Contributions Breakdown.....	2
Table of Contents.....	4
Work Assignment.....	5
<b>1. Customer Problem Statement of Requirement.....</b>	<b>6</b>
a. Problem Statement.....	6
b. Decomposition into Sub-Problem.....	9
c. Glossary of Terms.....	10
<b>2. Goals, Requirements and Analysis.....</b>	<b>11</b>
a. Business Goals.....	11
b. Enumerated Functional Requirements.....	12
c. Enumerated Nonfunctional Requirements.....	13
d. User Interface Requirements.....	15
<b>3. Functional Requirement Specification and Use Cases.....</b>	<b>17</b>
a. Stakeholders.....	17
b. Actors and Goals.....	20
c. Use Cases.....	21
i. Casual Description.....	21
ii. Use Case Diagram.....	22
iii. Traceability Matrix.....	23
iv. Fully Dressed Description.....	24

d. System Sequence Diagram.....	26
<b>4. User Interface Specification.....</b>	<b>30</b>
a. Preliminary Design.....	30
b. User Effort Estimation.....	32
<b>5. System Architecture and System Design.....</b>	<b>34</b>
a. Identifying Subsystems.....	34
b. Architecture Styles.....	35
c. Mapping Subsystems to Hardware.....	36
d. Connectors and Network Protocols.....	37
e. Global Control Flow.....	38
f. Hardware Requirements.....	39
<b>6. Project Management.....</b>	<b>40</b>
<b>7. References.....</b>	<b>43</b>

## **Work Assignment**

Uong SovanDara and Thol Ucca Kool have been assigned to mainly do most of the frontend on the website in W3School while Oziel Martinez is designing the UI and UX of the website. Alexander Redinger is linking the excel sheet calculation with the website while also refining the modding process of the master mod itself.

### **Individual Student Competence**

**Thol Ucca Kool:** Strong team player and problem solver with practical software development and system analysis skills. adept at generating ideas, debugging, and confirming functionality in practical tasks. The main programming languages include HTML/CSS, Java, JavaScript, Python, C++, and Java. knowledgeable about database administration, Prolog, Oracle APEX, and assembly (MASM). knowledgeable about software engineering projects, web and app development, and internship verification procedures.

**Oziel Martinez:** Skill set includes Python for scripting and automation, SQL with hands-on experience in CRUD operations, and Java for object-oriented programming projects. Proficient in Java, C++, Python, JavaScript, HTML, and Node.js for backend development, including building and consuming APIs. Skilled in database integration and backend system design, with a foundation in web technologies and software development principles.

**Uong SovanDara:** dependable team player, resourceful problem solver, and skilled communicator with strong organizational awareness and developing leadership skills. knowledgeable about HTML, CSS, SQL, Python, Java, C++, and JavaScript. Proficiency in Oracle APEX and Assembly Language, along with an awareness of web development and server upkeep.

**Alexander Redinger:** Currently, I can program in Python, C++, C#, Java, Javascript, HTML, CSS. I am extremely familiar with Excel and Google Spreadsheets. I am experienced in data analysis, and modding video games.

## **1. Customer Problem Statement of Requirement**

### **1a. Problem Statement.**

#### The Challenge of Balance and Customization in Total War: Rome II

The Total War: Rome II multiplayer community is experiencing a resurgence, largely thanks to a renewed interest in game modding. As more players venture into creating their own content, a significant problem has emerged: a lack of balance in the new mods. These imbalances often arise from a fundamental misunderstanding of unit value, a complex issue in a game where a unit's effectiveness is highly dependent on the specific in-game situation. A modder might see a unit perform exceptionally well in one engagement and, as a result, mistakenly increase its cost or downgrade its abilities, leading to a mod where units are either overpriced for their utility or so powerful for their cost that they become "overpowered" and ruin the competitive integrity of the game.

This challenge extends beyond simple balance. Many players find it difficult to find a faction in the vanilla game that perfectly aligns with their personal playstyle. While Rome II offers a diverse range of factions, each with its own set of strengths and weaknesses, players often feel they must compromise. For instance, a player might admire the defensive strength and discipline of Roman infantry but wish they could combine it with the rapid, shock-and-awe cavalry tactics of the Seleucids. However, without modding, this kind of hybrid faction is simply not possible. This forces players to choose between their preferred unit types, preventing them from experiencing the game with a truly tailored army.

The process of creating a custom, balanced faction is a formidable obstacle for most players. Total War: Rome II is an older game, and the resources available for new modders are few and far between. Even for those who manage to navigate the steep learning curve, the work itself is tedious and incredibly time-consuming. The bulk of the process involves manually editing large spreadsheets, a task that can take hours just to create a single faction. This combination of a lack of support and a high time commitment discourages many players from even attempting to create the custom content they so desperately want. The current system is a bottleneck, where a great idea for a custom faction often dies before it can even begin to be built.

## Customer Stories

### Negan's Quest for a Balanced Fantasy

Negan is a dedicated and highly skilled Total War: Rome II player, deeply embedded in the game's multiplayer community. While he is proficient with all of the base game's factions, his true passion is to merge his love for Rome II with his deep appreciation for the world of Lord of the Rings. He's scoured the internet for mods that do this, but has been consistently disappointed. The existing mods are often incomplete, and many suffer from poor balance, where certain units are laughably overpowered for their cost, which completely undermines the fun and competitive spirit of the game.

Negan's ultimate fantasy is to command an army of Elves in an epic confrontation against the Romans. He doesn't just want to participate in these battles; he wants to create the Elven faction himself, with units that accurately reflect his extensive knowledge of Lord of the Rings lore. He is confident that his vision is the most faithful and discerning. However, Negan is a busy individual with limited free time. He simply lacks the time and the desire to learn the complex and painstaking process of modding Rome II. As a player, not a developer, he wants a solution that allows him to bring his custom faction to life without spending hours on manual spreadsheet editing and complex development work.

### Graikos' Struggle with Subjective Modding

Graikos is a talented mod developer for Total War: Rome II who has created a mod that allows players to build their own custom factions. His process involves having players select 15 of their favorite units from the game, which he then compiles into a unique faction. To ensure some level of balance, he asks players to choose a specific culture, which provides a framework of strengths and weaknesses. However, the pricing of each unit is determined entirely by Graikos' subjective "gut feeling." This highly personal approach to balancing means that his personal biases can affect the mod; he may unintentionally lower the prices for a player he likes or unfairly inflate them for someone he doesn't.

This subjectivity has become a major source of stress for Graikos. The players who use his mod frequently complain about the lack of balance and the inconsistent pricing, which puts a

significant strain on his already busy schedule. The manual nature of his work only adds to the pressure. He must change every single unit stat by hand, one at a time. A single mistake requires a long, frustrating debugging process to pinpoint the error. To attract more players, he has even started taking on requests for custom visual designs, which is an even more labor-intensive and time-consuming process than the stat changes. To make matters worse, there are even fewer resources available to help with the visual modding of Rome II, leaving Graikos to navigate this complex process almost entirely on his own. The joy he once found in modding has been replaced by the stress of managing player complaints and the tedious, manual work of development.

Can you write a conclusion that sums up the two user stories?

### Conclusion: The Pain Points of Rome II Modding

Both Negan and Graikos represent the dual challenges facing the Total War: Rome II modding community. On one hand, you have the player who wants custom content but lacks the time and technical expertise to create it. Negan's story highlights the frustration of a passionate gamer who is forced to settle for poorly balanced, generic mods because the tools to build his perfect, lore-accurate faction simply aren't accessible to him. He is a creative user with a clear vision, but the complex and manual nature of the modding process is a massive barrier.

On the other hand, you have the mod developer who is overwhelmed by the manual, subjective, and time-consuming nature of the work. Graikos's story shows that even someone with the skills to create mods struggles under the weight of manual data entry, subjective balancing decisions, and the stress of dealing with a demanding community. The current process forces him to make every change by hand, leaving no room for a systematic, objective approach to balance. He is a bottleneck, and his personal feelings and limited time directly impact the quality and consistency of his work.

In essence, the core problem is a broken ecosystem: modding Rome II is a difficult and laborious process that is neither user-friendly for players nor efficient for developers. This leads to a vicious cycle where players can't get the custom, balanced content they crave, and the developers who try to provide it are pushed to their limits by the manual, tedious nature of the work.

### **1 b.) Decomposition into Sub-problems**

1. Unit data management
  - a. How do we gather, store and present unit stats in a structured way?
  - b. Solution: build a backend/database that holds unit stats in accessible format
2. Custom Faction Builder
  - a. How do we allow users to select units and form custom factions without the need for complex spreadsheets?
  - b. Solution: Create a User Interface for players to create custom factions and automatically generate faction data
3. Balance Calculator
  - a. How do we prevent subjective balancing?
  - b. Solution: Develop functions that will assign costs to the units based on weighted stats and balanced formulas
4. Lore and Customization Layer
  - a. How do we let users name their custom lore accurate factions into them game while still maintaining balance?
  - b. Solution: Provide options for faction themes and allow users to visually customize their faction's identity
5. Data editing
  - a. How to remove manual data entry into a spreadsheet?
  - b. Solution: Export mod-ready files directly from the calculator with minimal human editing
6. Error Prevention and Debugging
  - a. How do we prevent hours of debugging to find errors?
  - b. Solution: Implement validation( checking for missing stats, invalid costs), before export
7. User Accessibility and Time Efficiency
  - a. How do we make this usable for casual players with little technical knowledge?
  - b. Solution: Create a user interface(web app) that allows user to select units and click "Generate Mod"

### **1c. Glossary of Terms**

**Administrator:** An individual with the authority and responsibility to configure and manage a system.

**API (Application Programming Interface):** An API is a set of rules that enable different applications to communicate with each other. In this system's context, an API is used for data transfer and to utilize the functionality of external third-party systems.

**Application:** A software package that performs a specific function for an end user or another Application.

**Developer:** An individual that creates software using code.

**User:** An individual that interacts with or utilizes a systems resources and services

**Graphical User Interface (GUI):** An interface that allows users to interact with a computer or electronic device.

**Faction:** A collection of units a player picks from to create an army

**Unit:** A group of soldiers the player commands in-game.

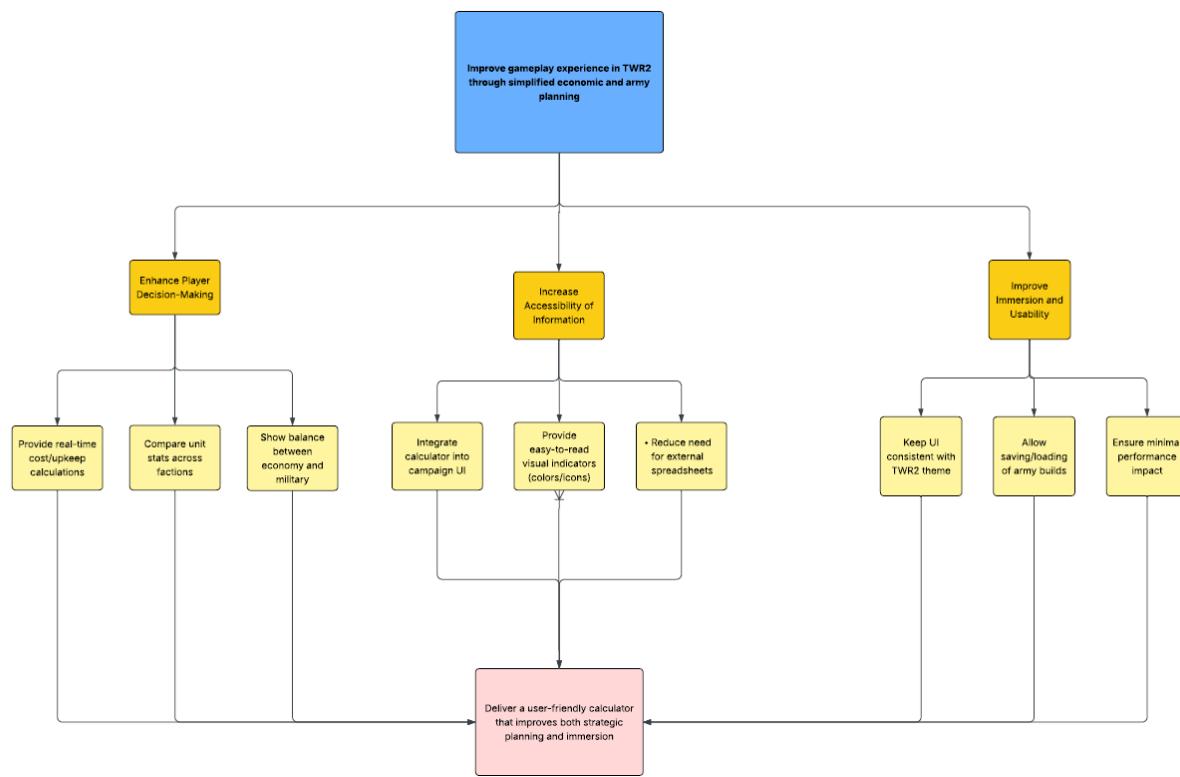
**Balancing (of units):** Correctly determining the price a unit is worth.

**Vanilla:** Playing the game without any mods.

**Mod:** A 3rd party modification of the game.

## **2. Goals, Requirements, and Analysis**

### **2a. Business Goals**



## **2b. Enumerated Functional Requirements**

<b>Identifier</b>	<b>PW</b>	<b>Requirement</b>
REQ1	5	The system shall calculate and display the total cost of a unit given certain properties of the unit.
REQ2	5	The system shall provide the user all the tools necessary to create their own faction.
REQ3	5	The system shall format the information given by the user into a way identical to the game files for easy integration into the game.
REQ4	4	The system shall provide the user with Error Codes, and methods to fix them to ensure certain rules are unbroken.
REQ5	3	The system shall provide a method to keep every users' calculator up to date.
REQ6	3	The system shall keep track of a users' faction across sessions, and allow the admin to access their faction.
REQ7	2	The system should allow user to customize the graphical aspects of their units.
REQ8	4	The system shall provide the user with all necessary information in order to create their units.

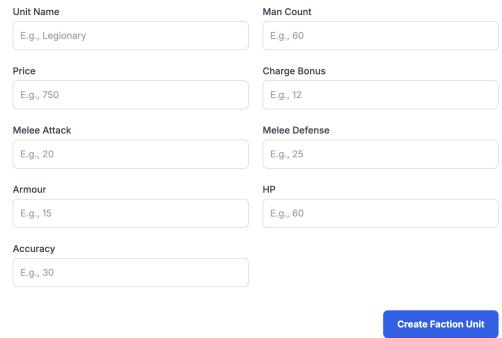
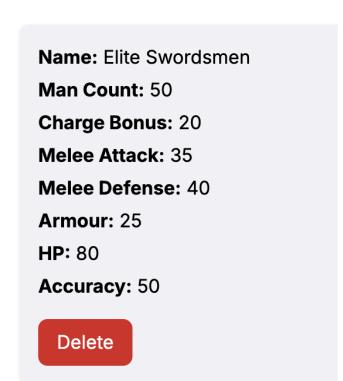
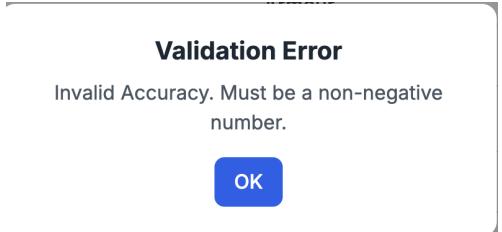
## **2c. Enumerated Nonfunctional Requirements**

<b>Identifier</b>	<b>PW</b>	<b>Requirement</b>
REQ9	5	The faction builder shall provide a simple and intuitive user interface with clearly labeled input fields.
REQ10	3	The system shall comply with WCAG 2.1 accessibility standards for usability and inclusivity.
REQ11	5	All submitted faction data shall be reliably stored without corruption or loss
REQ12	3	The system shall display clear error messages and prevent crashes when invalid data is entered.
REQ13	5	The system shall process player submissions and update Sheets within 2 seconds
REQ14	5	The system shall allow easy integration of a new faction into the master mod.
REQ15	3	The system shall scale to support at least 100 players created factions without significant performance issues.
REQ16	5	The project shall be maintained in GitHub with documentation and remain compatible with the latest version of Total War:Rome II
REQ17	3	The web builder shall run reliably on modern browsers (Chrome, Firefox, Edge etc.)
REQ18	2	The system shall require internet connectivity for faction creation and integration.

## **FURPS Table**

<b>Category</b>	<b>Description</b>
Functionality	<ul style="list-style-type: none"><li>• Provides a web based faction builder with input fields for unit composition, resources, and attributes.</li><li>• Automates calculations in and integrates results into the master mod.</li><li>• Enforces balancing rules such as unit caps and upkeep thresholds.</li></ul>
Usability	<ul style="list-style-type: none"><li>• The interface will be clear, simple, and accessible to new players.</li><li>• Input fields will be labeled, with tooltips for guidance.</li><li>• WCAG 2.1 accessibility standards will be followed.</li></ul>
Reliability	<ul style="list-style-type: none"><li>• Data will be stored securely and without corruption.</li><li>• Invalid inputs will generate error messages rather than crashes.</li><li>• Versioning ensures backup and recovery.</li></ul>
Performance	<ul style="list-style-type: none"><li>• Factions integrate into the master mod with minimal effort</li><li>• The system scales to 100+ factions without major slowdowns.</li></ul>
Supportability	<ul style="list-style-type: none"><li>• Source code will be documented and versioned in GitHub.</li><li>• Compatible with the latest Total War: Rome II Assembly Kit.</li><li>• Supported on modern browsers across platforms.</li></ul>

## 2 d) User Interface Requirements

Identifier	Priority	Requirement
REQ9	5	<p><b>Faction Builder</b></p>  <p>A user should be able to input custom faction</p>
REQ11	5	<p><b>Saved Factions</b></p>  <p>Factions should be reliably stored without corruption or loss</p>
REQ12	3	<p><b>Validation Error</b></p>  <p>A user should be able to see clear error messages</p>

REQ13	5	<b>Create Faction Unit</b>  System should process player submissions
REQ14	5	<b>Export to Mod File</b>  System should allow for easy integration to export to master mod

### **3. Functional Requirements Specification and Use Cases**

#### **3a. Stakeholders**

The Calculator Mod provides value not only to players but also to the wider Total War modding community and supporting platforms. Having a clear picture of stakeholders ensures the system meets the needs of all groups who have a vested interest in its success.

##### **Primary Stakeholders**

- **Players (Casual and Competitive):** End-users who want to customize their Rome II experience by creating balanced factions. They gain an easy-to-use builder that removes technical barriers.
- **Community Modders:** Contributors who add content, test balance, and expand the master mod with creative new factions. Their input ensures replayability and long-term community engagement.
- **Developers (Team D):** The project team members who design, code, test, and integrate the system. Their main interest is delivering a functional and innovative modding tool that works reliably.

##### **Secondary Stakeholders**

- **Administrators:** Individuals responsible for maintaining the shared mod, ensuring updates, managing version control, and troubleshooting errors.
- **Total War Community (Broader Audience):** Players and online communities who benefit indirectly from a growing pool of balanced factions available for download.
- **Creative Assembly (Game Developer):** While not directly involved, their Assembly Kit and modding framework define the technical boundaries within which this project operates.

##### **Supporting Platforms**

- **Google Sheets API:** Provides backend computations for unit balance and upkeep.
- **W3Schools Spaces:** Hosts the faction builder web app, ensuring reliable online access.

- **GitHub Repository:** Maintains version control, collaboration, and distribution of the master mod.

### **3b. Actors and Goals**

<b>Actors</b>	<b>Participating/ Initiating</b>	<b>Role</b>	<b>Goal</b>
Player	Initiating	End-user who wants to build a custom faction.	To create a personalized, balanced faction quickly without technical modding knowledge.
Community Modder	Initiating/Participating	Experienced modder who may both use and test the system.	To add creative factions, test balance, and expand the master mod for replayability.
Administrator	Participating	Maintains the system, reviews submissions, and manages the shared mod/master mod	To ensure the system runs smoothly, manage errors, and coordinate integration of factions.
Rome II Assembly Kit	Participating	Official creative assembly modding tool	To compile and integrate newly created factions into the game environment.
GitHub Repository	Participating	Source control and distribution platform	To store master mod files, provide version control, and make updates accessible to the community.
W3School Spaces	Participating	Hosting platform for the web application	To provide stable, accessible online hosting so players can use the faction builder anywhere.

### **3c. Use Cases**

#### **i. Casual Description**

##### **UC1: User Creates Faction**

Description: User fills in every customization option and stat slot for their units.

Responds to Requirements: REQ1, REQ2, REQ4, REQ5, REQ6, REQ7, REQ8, REQ9, REQ10, REQ11, REQ12, REQ17, REQ18

##### **UC2: User Updates their Faction**

Description: User makes changes to their already-existing faction.

Responds to Requirements: REQ5, REQ6, REQ11, REQ12, REQ17, REQ18

##### **UC3: Modder Integrates Units into Game**

Description: Modder puts the units from the calculator into the game.

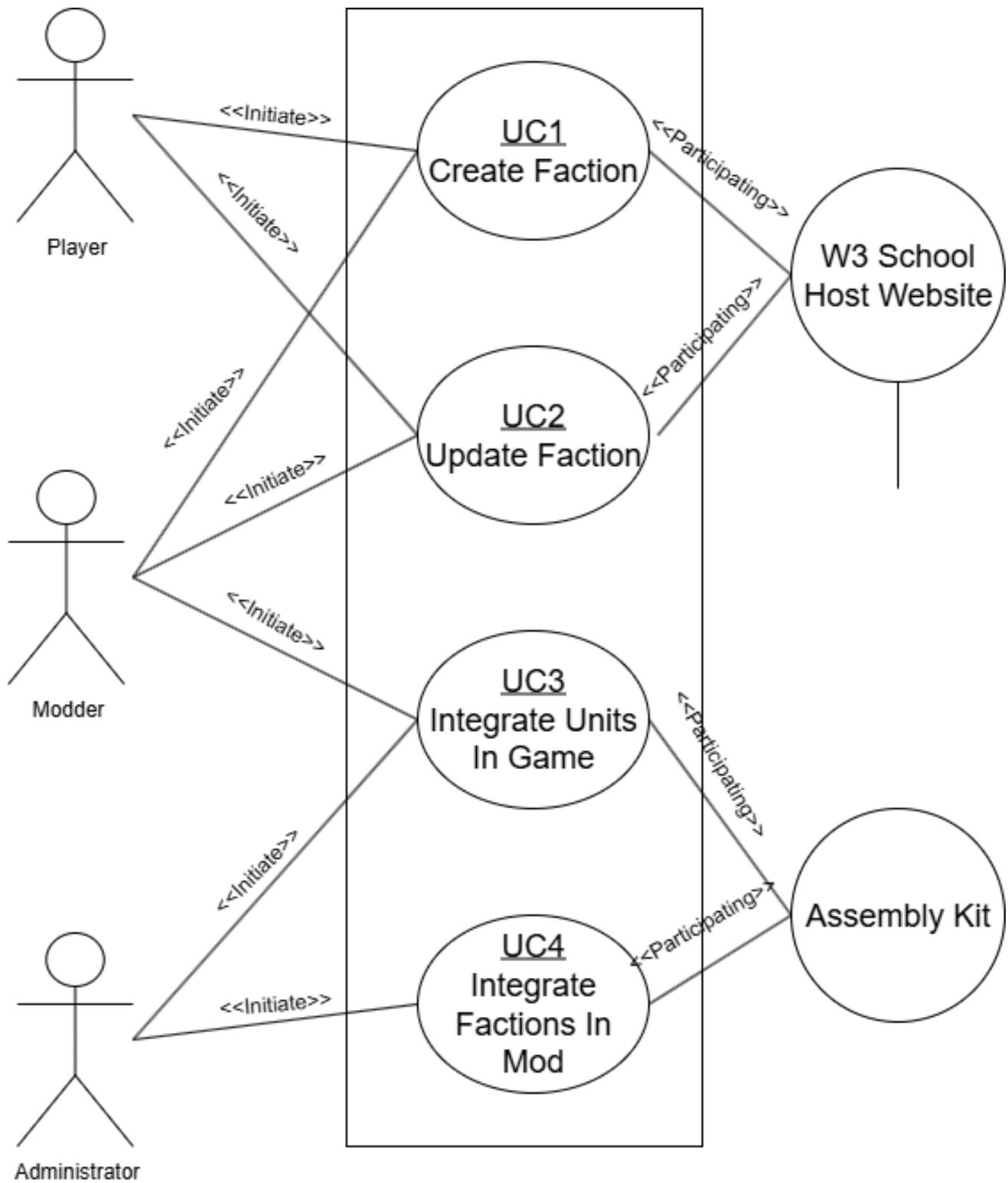
Responds to Requirements: REQ3, REQ14, REQ17, REQ18

##### **UC4: Developer Integrating Others' Units into Mod**

Description: Developer accessing others' factions to put into master mod.

Responds to Requirements: REQ3, REQ5, REQ6, REQ11, REQ13, REQ14, REQ15, REQ16, REQ17, REQ18

## ii. Use Case Diagram



### **iii. Traceability Matrix**

Requirement S	PW	UC1	UCC2	UC3	UC4
REQ1	5	X			
REQ2	5	X			
REQ3	5			X	X
REQ4	4	X			
REQ5	3	X	X		X
REQ6	3	X	X		X
REQ7	2	X			
REQ8	4	X			
REQ9	5	X			
REQ10	3	X			
REQ11	5	X	X		X
REQ12	3	X	X		
REQ13	5				X
REQ14	5			X	X
REQ15	3				X
REQ16	5				X
REQ17	3	X	X	X	X
REQ18	2	X	X	X	X
Total PW	70	47	19	15	39

#### **iv. Fully-Dressed Description**

##### **UC1: User Creates Faction**

**Related Requirements:** REQ1, REQ2, REQ4, REQ5, REQ6, REQ7, REQ8, REQ9, REQ10, REQ11, REQ12, REQ17, REQ18

**Initiating Actor:** User, or Modder

**Participating Actor:** None

**Preconditions:** The user interface allows user to enter information and select options

**Postconditions:** The data is saved and accessible by developer

**Flow of Events for Main Success Scenario:**

1. User enters Faction Customization Options
2. User enters unit stats, the calculator shows the appropriate outputs in response

##### **UC2: User Updates Faction**

**Related Requirements:** REQ5, REQ6, REQ11, REQ12, REQ17, REQ18

**Initiating Actor:** User, or Modder

**Participating Actor:** None

**Preconditions:** The user's data is accessible by the user

**Postconditions:** The data is saved and accessible by developer

**Flow of Events for Main Success Scenario:**

1. User opens up their faction
2. User modifies unit stats, the calculator shows the appropriate outputs in response

### **UC3: Modder Integrates Units into Game**

**Related Requirements:** REQ3, REQ14, REQ17, REQ18

**Initiating Actor:** Modder

**Participating Actor:** None

**Preconditions:** The modder has a faction already filled out

**Postconditions:** The units are usable in-game.

**Flow of Events for Main Success Scenario:**

1. Modder opens up their faction
2. Modder accesses Backend, and copy-pastes into game files
3. Units are available without errors.

### **UC4: Developer Integrating Others' Units into Mod**

**Related Requirements:** REQ3, REQ5, REQ6, REQ11, REQ13, REQ14, REQ15, REQ16, REQ17, REQ18

**Initiating Actor:** Developer

**Participating Actor:** None

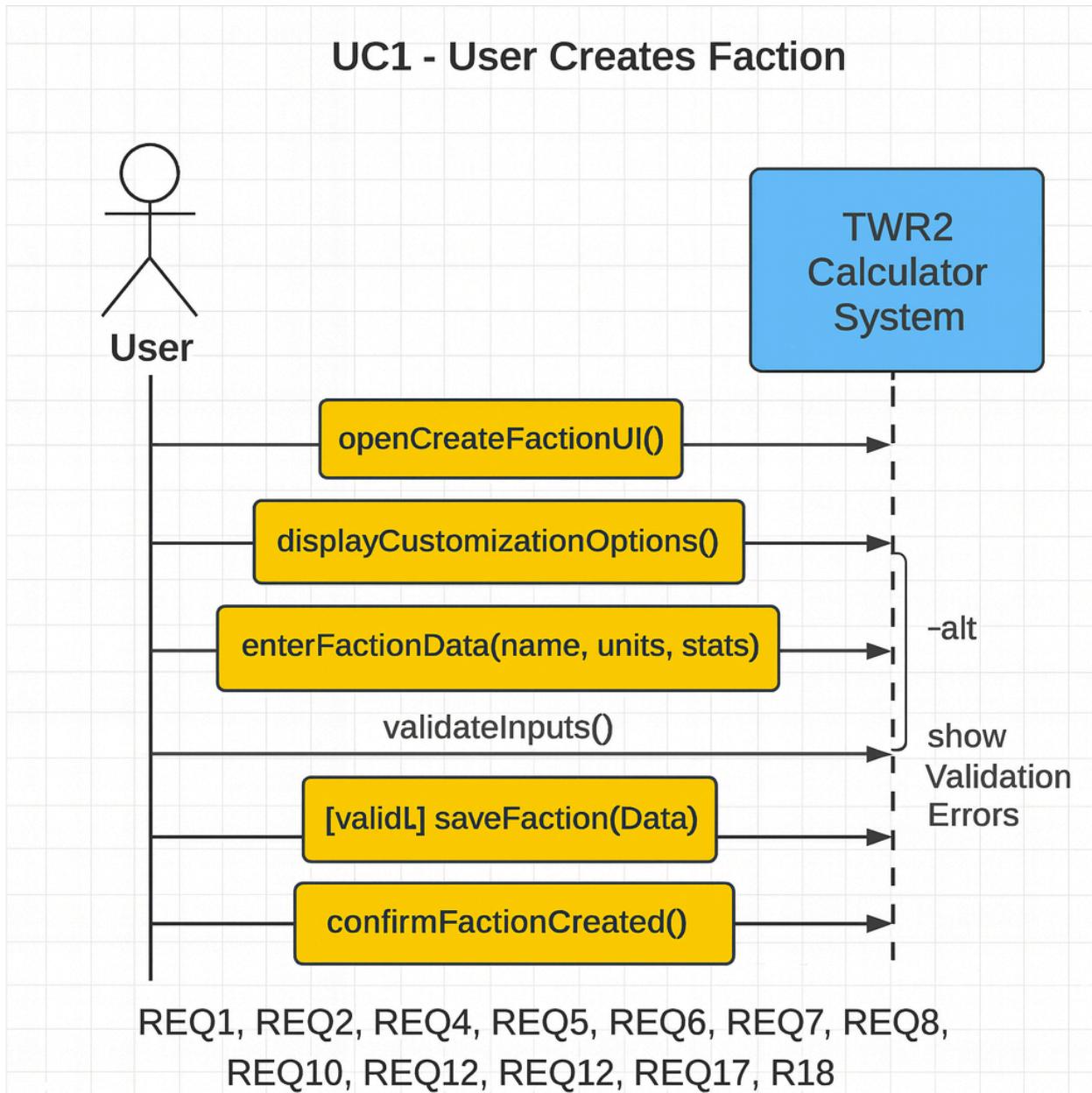
**Preconditions:** The developer has access to others' factions

**Postconditions:** The factions are usable in the mod

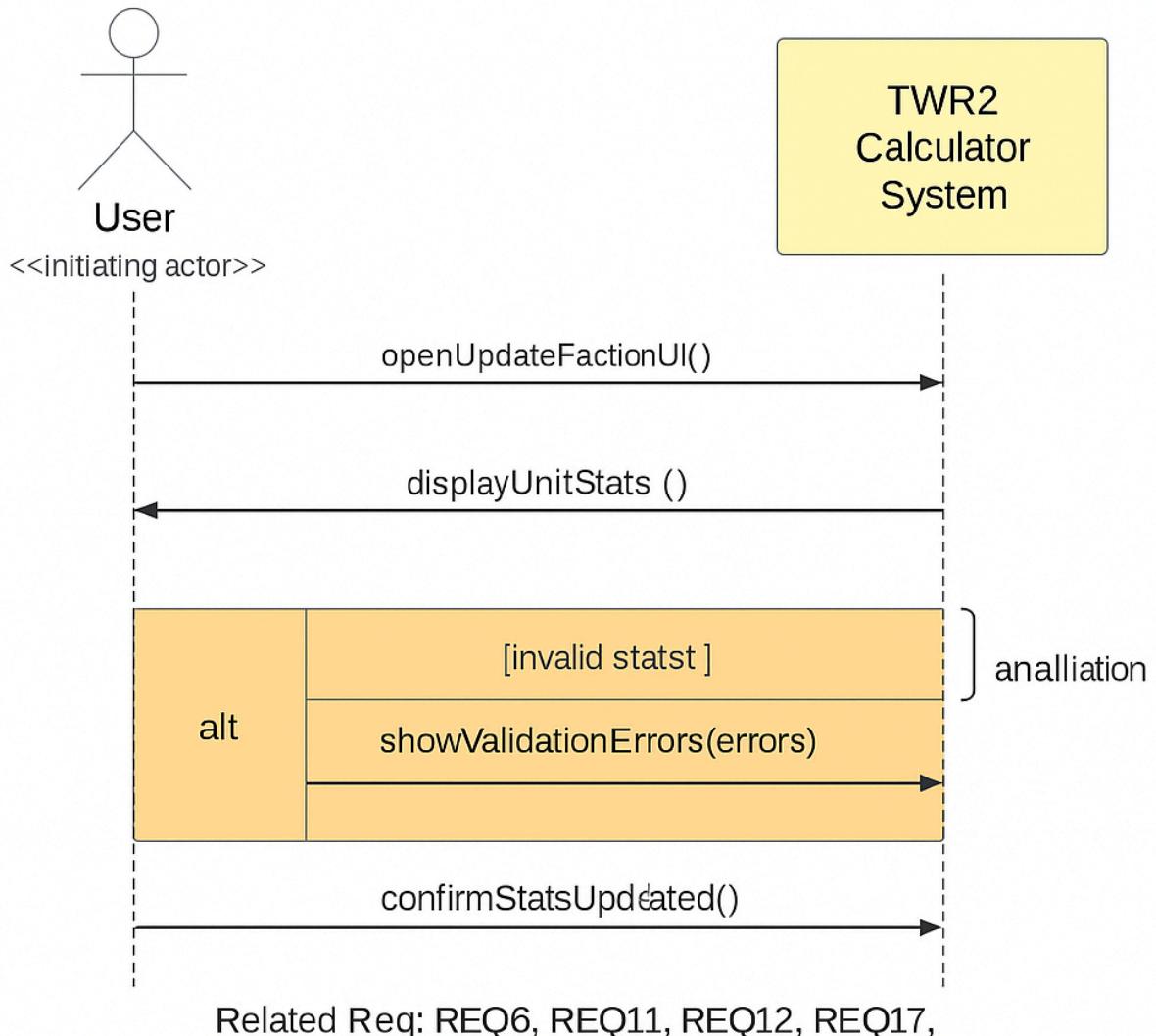
**Flow of Events for Main Success Scenario:**

4. Developer opens others' faction
5. Developer accesses Backend, and copy-pastes into game files
6. Factions are available without errors.

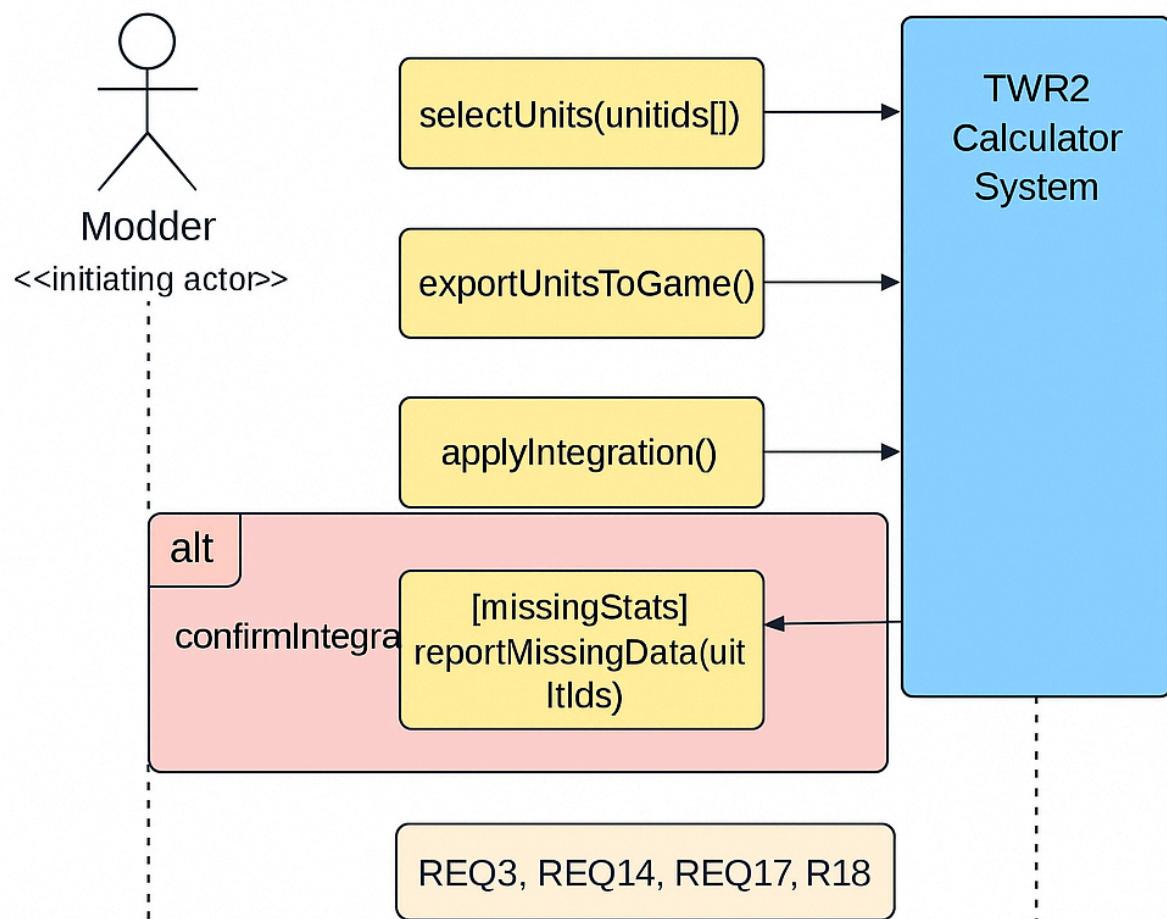
### 3d. System Sequence Diagrams



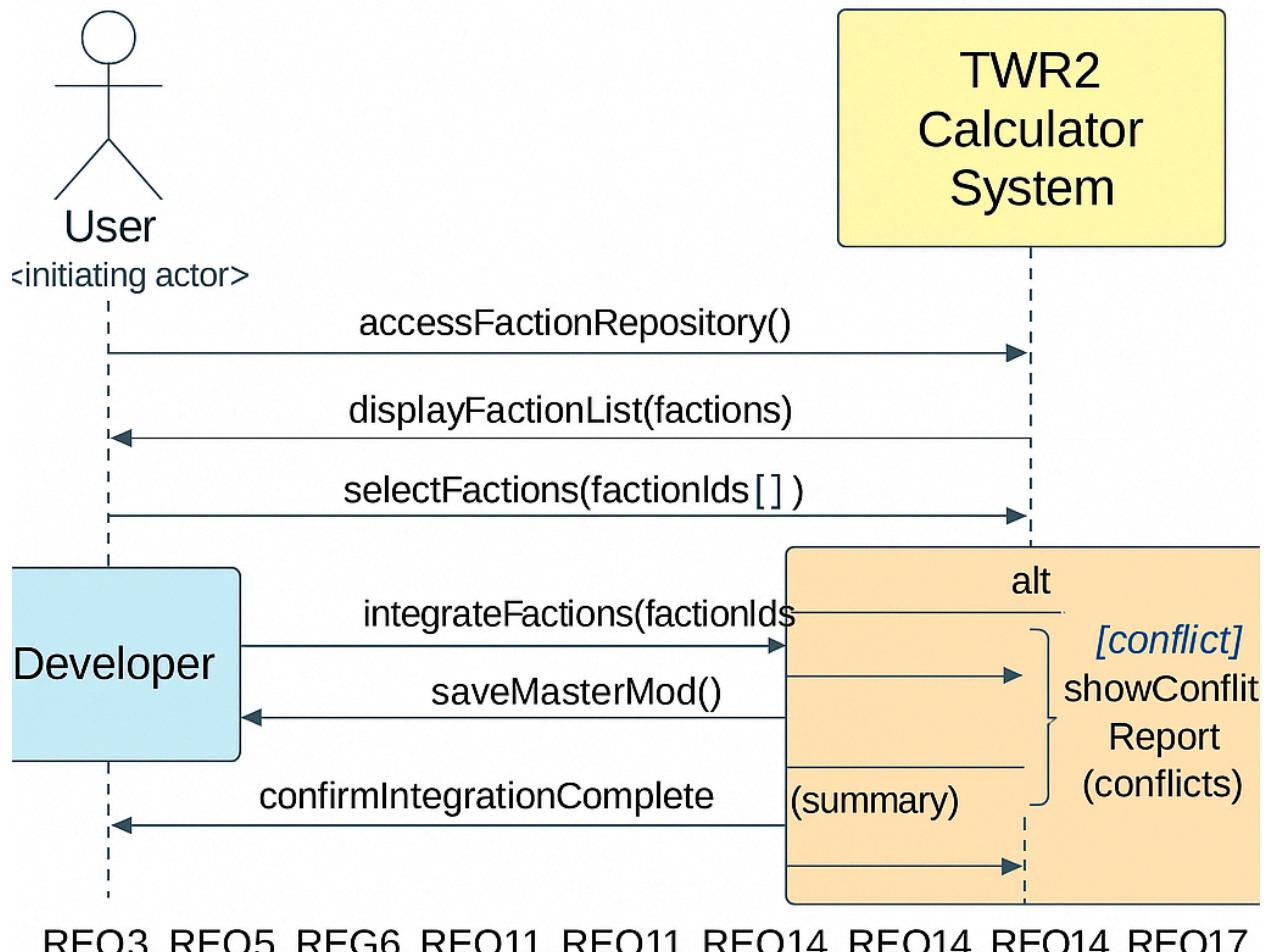
## UC2: User Updates Faction



## UC3: Modder Integrates Units in Game



## UC4 - Developer Integrates Factions in Mod



## **4. User Interface Specification**

### **4a) Preliminary Design**

1. Open the Faction Builder
  - a. User selects “Create New Faction” from the navigation bar
  - b. System displays the Faction Builder form, including input fields for unit attributes and dropdown menus for culture and unit type
2. Enter Unit Information
  - a. The user then fills in the chosen fields Name, Man Count, Charge Bonus, Morale, Melee Attack, Melee Defense, Armour, HP, Accuracy, Ammo etc.
  - b. Drop down menus are displayed for Melee Weapon, Missile Weapon, Shield, Mount, attributes, and abilities.
  - c. The system validates input in real-time and displays price and points usage.
3. Add Unit to Faction
  - a. The user presses the “Add Unit” Button
  - b. The system displays the unit in a table below the form, showing a summary of entered values
4. Submit Faction
  - a. Once the user has added all desired units, they press the “Submit Faction” button.
  - b. System saves the faction data, validates all fields, and exports the custom faction
  - c. A confirmation message is displayed: “Faction submitted successfully.”

**Figure: The Mod Calculator (Desktop)**

The image shows a screenshot of the 'The Mod Calculator' application, specifically the 'Unit Calculator' section. The interface is clean and modern, featuring a dark header bar at the top with the title 'The Mod Calculator' and a subtitle 'Unit Calculator'. Below the header is a navigation bar with five tabs: 'Basic Stats' (selected), 'Combat Stats', 'Equipment', 'Attributes', and 'Results'. The main content area is titled 'Basic Unit Information' and contains three input fields: 'Unit Name' (placeholder: 'e.g., Roman Hastati'), 'Man Count' (placeholder: 'e.g., 120'), and 'Culture' (a dropdown menu labeled 'Select Culture'). Below these fields is a 'Unit Type' dropdown menu labeled 'Select Unit Type'. At the bottom of this section are three buttons: 'Add Unit' (dark brown), 'Reset All Fields' (dark brown), and 'Submit Faction' (green). Below this section, there are two empty input fields labeled 'Price' and 'Points Usage'.

## **4b) User Effort Estimation**

UC1: User Creates a faction with all the customization options and stats for their units

1. Select “Basic Stats” → 1 click (navigation)
2. Enter Unit Name (x15 units) → ~ 10 keystrokes(clerical)
3. Enter Man Count (x15 units) → ~3 keystrokes (clerical)
4. Select “Combat Stats” (x15 units) → 1 click( navigation)
5. Enter Melee Attack, Melee Defense, Charge Bonus, Morale, Armour, HP, Accuracy(x15)  
→ 3 clicks(clerical) + 6 keystrokes(clerical)
6. Select Equipment (x15) → 1 click (navigation)
7. Select Primary Weapon, Shield Type, Missile Weapon, Ammunition, Accuracy,  
Range(x15) → 3 clicks (clerical) + 6 keystrokes (clerical)
8. Select Attributes → 1 click (navigation)
9. Select Mount, Artillery/Animal/Chariot(x15) → 2 clicks (navigation) + 2 clicks (clerical)
10. Enter Num of Artillery/Animal/Chariot(x15) → 2 keystrokes (clerical)
11. Select “Add unit” button to add the unit(x15) → 1 click (navigational)
12. Select “Submit Faction” → 1 click navigational

Totals:

Keystrokes and Clicks = approx. 856

Navigation vs Clerical : ~ 9 % navigational vs 91.1% Clerical

UC2: User Updates Faction

1. Select Unit to Update → 1 click( navigation)
2. Change Unit Name → 10 keystrokes (clerical)
3. Adjust 3 stat fields (e.g. melee attack, Charge Bonus, HP) → 6 keystrokes (clerical)
4. Click “Save Changes” -> 1 click (navigation)

Totals:

16 keystrokes, 2 clicks

11% navigation vs. 88.85% clerical

### UC3: Modder Integrates Units into Game

1. Click “Export Faction to Sheet” → 1 click (navigation)
2. Choose export location → 2 clicks (navigation) + 10 keystrokes (filename, clerical)
3. Open PACS file reader → 2 clicks (navigation)
4. Navigate through folders (~3–4 levels deep) → 4 clicks (navigation)
5. Open the target mod file → 1 click (navigation)
6. Switch to exported sheet → 1 click (navigation)
7. Copy values-> 2 keystrokes, keyboard shortcuts(clerical)
8. Switch back to PACS file → 1 click (navigation)
9. Paste values → ~2 keyboard shortcuts (clerical)
10. Save changes → 1 click (navigation)

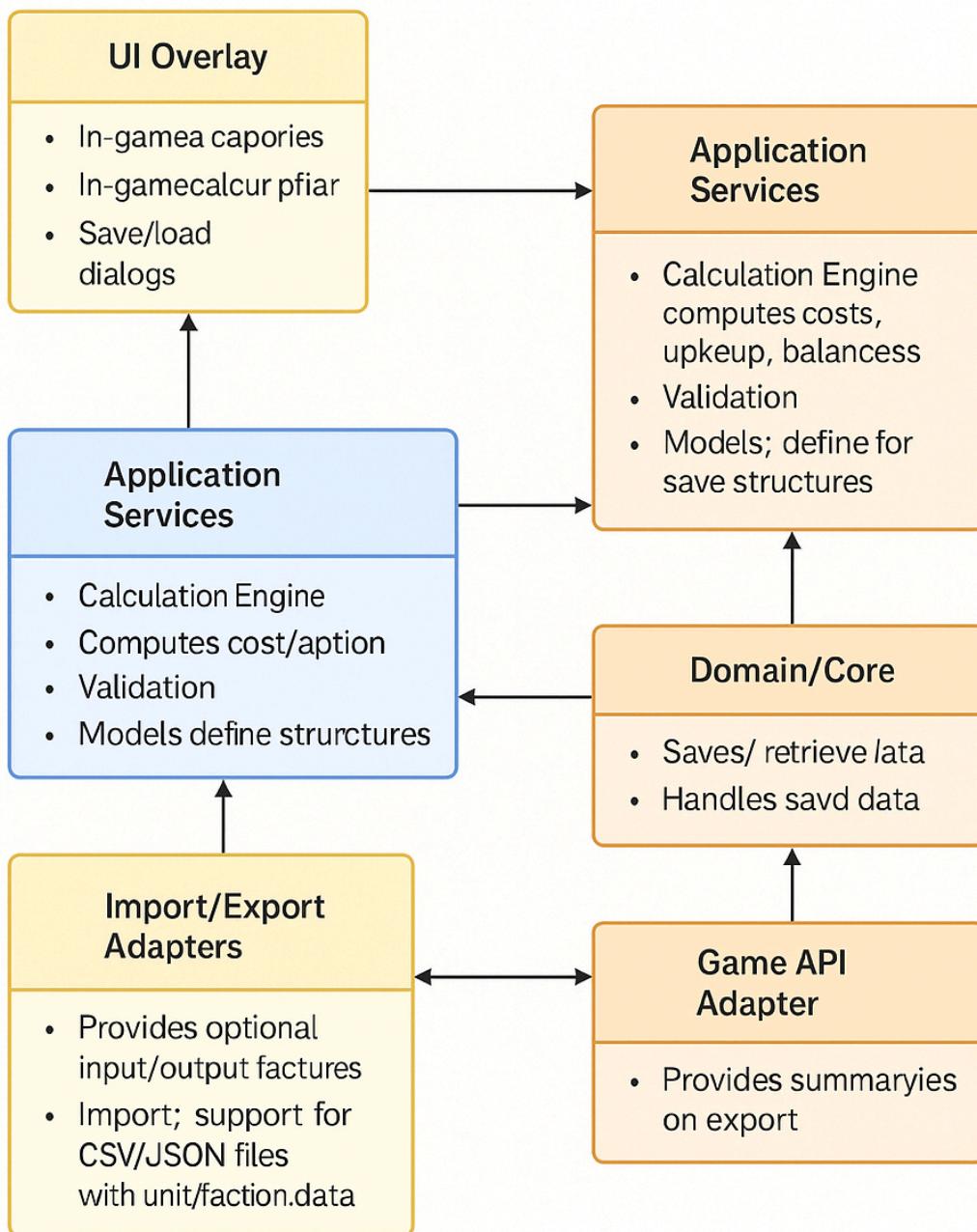
Totals 27 keyboards + clicks

48% Navigational

52% Clerical

## 5. System Architecture and System Design

### 5a. Identifying Subsystems



## **5b. Architecture Styles**

We will develop The Calculator Mod as a monolithic web application. This design choice simplifies development and deployment by housing the user-facing website and all backend logic in a single, cohesive codebase. Users will interact directly with this website, which will be the central hub for creating and modifying custom factions. The application will leverage an event-driven architecture to dynamically update unit prices based on user actions.

The Calculator Mod will exist as a single, unified website where users can create their own custom factions. All components of the application, including the user interface (frontend), the core business logic (backend), and the database, will be tightly integrated into this single deployment. This architecture provides a straightforward user experience; a user simply navigates to the website and starts building their faction. There is no need for separate applications or complex setup processes. The application will serve as the sole platform for users to define their desired faction, including selecting units and customizing their stats.

The core functionality of The Calculator Mod will rely on an event-driven system. When a user makes a significant change, such as selecting a new faction or adding/removing a unit, an event will be triggered. This event will initiate a process that automatically calculates and updates the unit prices in real-time. For example, if a user changes their units to be more like Roman units, rather than Greek-style units, the system will adjust the price of all units to reflect the new cultural and gameplay attributes. The developer's workflow will be just as streamlined. After a user finishes creating their custom faction, the developer can access their data within the same monolithic application. This allows them to easily retrieve the completed faction data and integrate it directly into the game. This approach eliminates the need for manual communication or data transfer between the user and the developer, as all relevant information is stored and accessible in one place.

### **5c. Mapping Subsystems**

- **Faction Builder Subsystem:** This subsystem runs inside the user's browser. It provides the interface for entering unit stats (man count, attack, defence, etc), uploading banners. It also sends the user input to the server for validation, balance calculations, and storage
- **Database Subsystem:** Manages storage of user-created factions, unit stats, and customization options. Ensures persistence and supports scaling to 100+ factions. Hosted on the server, accessed via API calls from the Faction Builder
- **Balance Calculator Subsystem:** Processes unit data to ensure fair gameplay. Runs on the server for consistency and speed. Returns results to the browser in under 2 seconds.
- **Export and Integration System:** Generates output in Sheets/CSV format for modders. Files are downloaded by the user's browser. The exported file is then manually copied into the Total War: Rome II PACS mod structure by the modder (outside the system).
- **Communication Infrastructure:** Provides secure, reliable exchange between browser(client) and web server. Handles API calls, data submission, and error reporting

## **5d. Connectors and Network Protocols**

The TWR2 Calculator Mod is designed to run locally on a single machine alongside Total War: Rome II. Because of this, the system does not depend on external networks or internet-based services. Instead, it relies on internal connectors and file-based protocols to manage communication between subsystems.

### **In-process calls.**

The UI Overlay interacts with the Application Services through direct function calls. These calls then flow into the Domain/Core logic, ensuring that calculations for costs, upkeep, and comparisons are performed quickly and without noticeable delay for the player.

### **Event notifications.**

To keep the interface responsive, certain actions within the Application Services trigger lightweight events, such as when totals are computed or a faction build is saved. These event signals allow the UI to update dynamically without requiring the player to restart processes.

### **File connectors.**

Persistence is managed using JSON files for storing factions, units, and builds. When integrating these into the game, the Game API Adapter writes the data into pack or database files, which are the standard formats recognized by Rome II. This file-based connector approach allows the mod to maintain compatibility with existing game infrastructure.

### **Future extension.**

Although not required for core functionality, the design could support online sharing of saved builds through HTTPS endpoints. This would provide players the option to upload and download builds while still allowing the mod to function entirely offline if network access is unavailable.

## **5e. Global Control Flow**

- **Execution Orderness:** The system operates using an **event-driven architecture**. This means the program waits for specific events, such as user actions or system triggers, to occur. When an event happens, it activates a specialized handler to process the request. These handlers are designed to perform a specific task and return the results, allowing events to be processed in any order and at any time they happen.
- **Time Dependency:** The Calculator Mod is not a real-time system, meaning it doesn't have strict timing requirements for its core operations. Instead, it processes user events as they happen, and those events can occur at any time. The system's logic isn't dependent on a clock or a scheduler. It simply responds to user actions in an **on-demand fashion**.  
For example, when a user changes their chosen faction or modifies a unit's stats, the system immediately processes that event and recalculates the unit's cost. This ensures that the user always sees updated information as they build their custom faction, without any need to wait for a specific time or trigger.

## **5f. Hardware Requirements**

- **Screen Display:** The Faction Builder subsystem will run on multiple devices such as laptops, desktops, and tablets. It must support modern web browsers at varying resolutions. Minimum supported resolution is 1280 x 720 pixels, but the system is optimized for 1920 x 1080 pixels or higher. A color display is required to view banners, icons, and faction customization options properly.
- **Communication Network:** The system requires a stable internet connection for real-time communication between the client (browser) and the server. A minimum 5 Mbps connection is recommended to ensure smooth interaction with the Balance Calculator, database access, and file export. Intermittent connectivity may result in delays or failed submissions.
- **Database Server:** The database subsystem will run on a server hosting PostgreSQL( or equivalent relational database). Minimum requirements include 2 CPU cores, 4 GB RAM, and 20 GB disk storage to support up to 100 user-created factions. Scaling beyond this capacity may require additional memory and disk space.
- **Application Server:** The server hosting the Balance Calculator and Export Subsystem must support [Node.js](#)( or equivalent backend runtime). Minimum requirements include 2 CPU cores and 4 GB RAM to ensure calculations are processed within 2 seconds.
- **User Machine:** Users must have a device capable of running modern web browsers. Minimum requirements are a dual-core processor, 4 GB RAM, and 500 MB of available disk space for the downloaded export file.
- **External Modding Environment:** to integrate exported factions into the ROME II game, modders will need a PACS file reader/editor installed on their local machine, along with sufficient disk space for Rome II game files.

## Project Management

## **Plan of Work:**

Through our Calculator Mod, we will give gamers access to a faction builder integrated into Total War: Rome II using widely accessible online technologies and hosting from W3Schools spaces and GitHub. A web application is created, linked to Google Sheets for computations, and scripts are written to include each faction into a common master mod.

**Languages:** HTML, CSS, JavaScript(frontend), Python(Backend)

**Platforms/Tools:** GitHub, W3Schools Spaces(hosting), Google Apps Script

**Integrations:** Google Sheets API, Total War: Rome II Assembly Kit, GitHub repository for version tracking.

## **Success Criteria:**

- The online builder, which is hosted on W3Schools Spaces, allows players to establish factions, and the results are saved in Google Sheets.
  - When creating a faction, the balance rules are automatically applied.
  - The master mod seamlessly incorporates data from Sheets.
  - All community factions are constantly updated in the common master mod.

## Road Map:

## **Product Ownership & Responsibility Breakdown**

### **Product Ownership**

The Calculator Mod is a shared team product, guided by distributed ownership. Each member champions one area, while decisions are reached collaboratively.

#### **Uong SovanDara (Coordinator & Web Lead)**

- **So far:** Drafted the problem statement and goals, coordinated report submissions, and created initial UI layout in W3Schools Spaces.
- **Current:** Leading the development of the front-end Faction Builder subsystem, ensuring WCAG 2.1 accessibility compliance, and maintaining project documentation.
- **Future:** Implement navigation flows across subsystems, finalize UI testing on multiple hardware platforms, prepare demonstration materials, and oversee integration of user feedback into the final release.

#### **Oziel Martinez (Data Integration & Testing Lead)**

- **So far:** Researched Google Sheets API, built schema for unit costs and balancing formulas, and created validation requirements.
- **Current:** Developing the Balance Calculator subsystem and implementing error-handling scripts. Running preliminary database stress tests with faction datasets.
- **Future:** Conduct large-scale performance testing (100+ factions), automate regression testing for balancing rules, prepare testing documentation, and validate hardware requirements for server scaling.

#### **Thol Ucca Kool (Systems Analyst & Debugging Lead)**

- **So far:** Contributed to subsystem identification and architecture mapping, supported debugging in early prototype builds, and refined use case flows.
- **Current:** Monitoring subsystem interactions (UI ↔ Sheets ↔ backend ↔ export), creating detailed system sequence diagrams, and coordinating early integration testing.
- **Future:** Lead debugging during final integration, optimize backend performance, create user troubleshooting documentation, and ensure smooth communication across connectors and network protocols.

#### Alexander Redinger (Backend & Mod Integration Lead)

- **So far:** Provided expertise on Rome II Assembly Kit, initiated backend pipelines for export and integration, and validated balance formulas against in-game mechanics.
- **Current:** Expanding the Export and Integration subsystem to support multiple concurrent uploads, refining the backend data flow, and ensuring compatibility with the PACS file structure.
- **Future:** Build advanced balancing algorithms to prevent exploits, manage compatibility testing across Rome II versions, implement version control protocols for mod releases, and develop training materials for community moderators to maintain the shared master mod.

## **References:**

1. GitHub, version control and project planning <https://docs.github.com/en>
2. Creative Assembly. *Total War: Rome II – Assembly Kit (TWR2)*. Available via Total War Wiki. [https://wiki.totalwar.com/w/Assembly\\_Kit\\_%28TWR2%29.html](https://wiki.totalwar.com/w/Assembly_Kit_%28TWR2%29.html)
3. W3Schools. *HTML, CSS, and JavaScript Tutorials*. <https://www.w3schools.com>
4. Google. *Google Docs Editors (Docs, Sheets, Slides)*. <https://www.google.com/docs/about>