Revised List of Features

This is an updated list of your FEATURES inventory (from Milestone 2). It is normal for feature lists to change during the course of a project. Some features may have been dropped. Some features may have been added. This revised features list should reflect these changes. This revised features list should identify the PRIORITY order of how the features will be developed.

Project Features:

1. Multiplayer
    a. The game will be played in sessions filled with about 20 players. The goal will be to eliminate all other players.
    b. Non-Functional Requirements:
        i.
    c. Functional Requirements:
        i. Architecture
            1. Backend Redis Server
            2. Web Server
            3. Backwards Proxy Server
            4. Session Proxy Server
    d.
2. 4x Model
    a. The 4x Model is expand, explore, exploit and exterminate. Players will need to explore the map and exploit available resources to build up armies and be the last player.
    b. Functional
        i. Unit functionality
            1. What: The main feature of 4x games is unit interactions. Units will be the driving force behind the game.
            2. Who: Any players
            3. Why: It is one of the core components of the game
            4. When: During the game
        ii. Win/Loss conditions
            1. What: Informs the players when they they have won or lost
            2. Who: Players who meet either condition.
            3. Why: Gives players a goal to obtain and keep them invested
            4. When: At the end of the player's game
    c. Non-Functional Requirements
        i. Control Units
            1. What: Players need to be able to send commands to their units to execute plans.
            2. Who: Players

3. Why: To add strategy to the game, players will need to be able to send commands to their units.
4. When: When the player wants to control units

ii. Build bases/cities

1. What: Players will be able to build buildings to control points and get more resources
2. Who: Players
3. Why: Allowing players to build buildings, players will be able to create more detailed plans and get more invested in the game.
4. When: When players want to build buildings

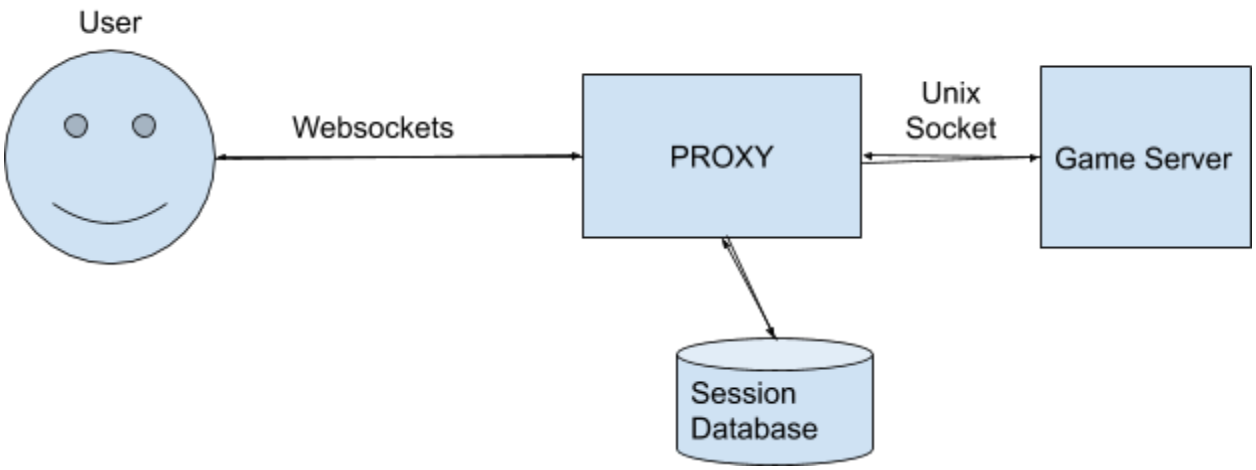
3. Leaderboard
    a. Tracks statistics of top players and displays on the Leaderboard page. Updates regularly.
    b. Functional Requirements
        i. Stores information
            1. What: Database stores player information from games
            2. Who: Players
            3. Why: Allows us to fill in leaderboard and get information to balance units
            4. When: After a game is complete
    c. Non-Functional Requirements
        i. Shows information
            1. What: Webpage that lets player see their stats compared to other players
            2. Who: Players
            3. Why: Allows players to see how good they are and compete with friends
            4. When: When players want to see their stats.
4. Terrain Generation
    a. Allows for strategy so players will have to adjust and overcome issues involved in how the land around them is made.
    b. Functional
        i. Wave function collapse
            1. Takes in seed and generates very well designed maps.
            2. Uses pre-made chunks to generate even cooler chunks.
            3.
    c. Non-Functional Requirements
        i. Good design and patterns in land that are strategic to use.
5. Base generator

a. Kind of like most city builders the land is zoned for certain buildings and structures that will aid in resource gathering and unit building.
b. Functionality
    i. Users will zone the land in the game and communicate that with the servers.
    ii. The server will describe a timeline for the buildings to complete the build and communicate that back to the users.
    iii. Wave Function collapse will work with on top of terrain to build the bases
c. Non-Functional Requirements
    i. The users see a satisfying building that generates in layers and conjoins with surrounding paths on the ground.

6. Server side statistics
    a. Since it would be a shame to let the players tell the server if they won the battle, instead the server before interaction or battle will calculate the statistics and prepare ahead. On top of this the server is also scheduling events in the game and announcing them as they occur.
    b. Functionality
        i. State storage
        ii. State backup database incase game server crash
        iii. Statistics engine calculating probabilities and actions.
    c. Non-Functional Requirements
        i. Users will be alerted to enemies and can watch the battles
        ii. Battles will be realistics in states aka the foot troops should have died from the tank.

Architecture Diagram

This deliverable is a picture or diagram that shows each architectural component of your application. The diagram should identify how your application's front-end, integration layer, and backend processes will be hosted. This diagram should identify the flow of data from one layer to another. This diagram should identify the protocols being used to/from each component layer.

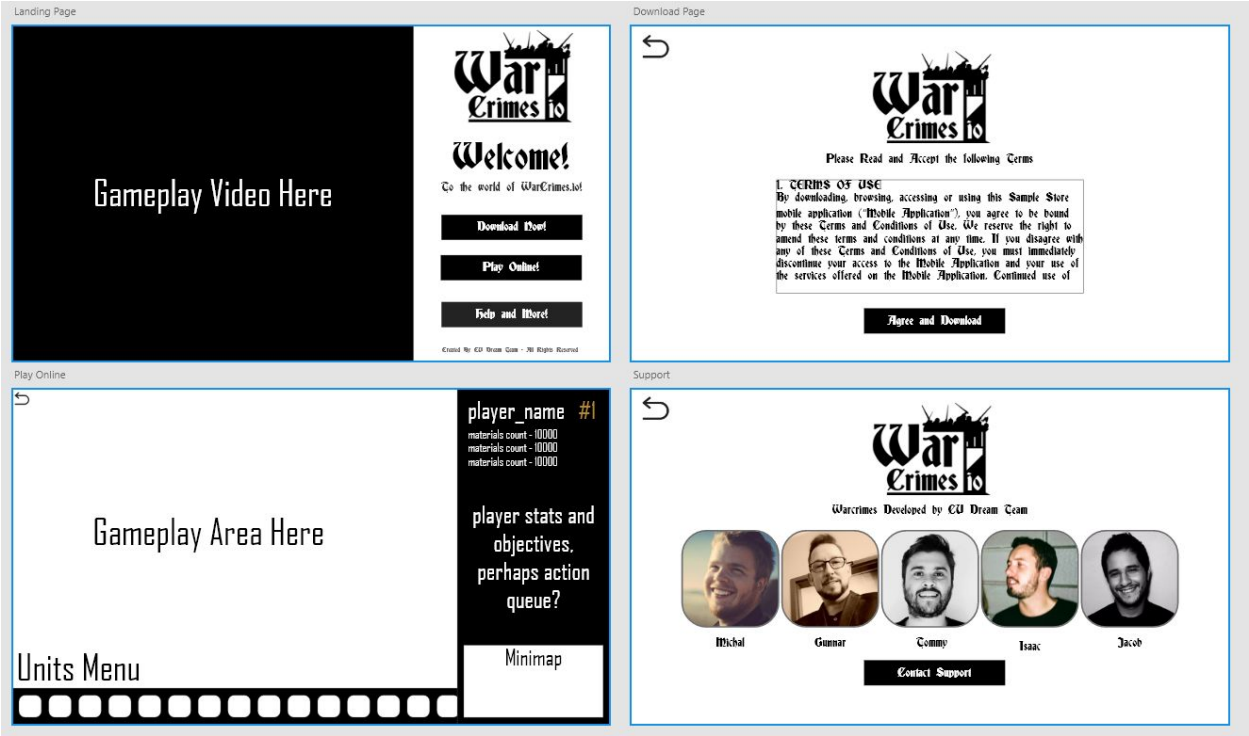


Front End design

This deliverable is a series of diagrams that show the basic design of your application's front end. Typically, this design is most easily presented in terms of a WIREFRAME. This may be hand drawn, or created using a web page wireframe drawing tool. The front end design should identify each major feature of the application's front end

Website and UI designed using Adobe XD

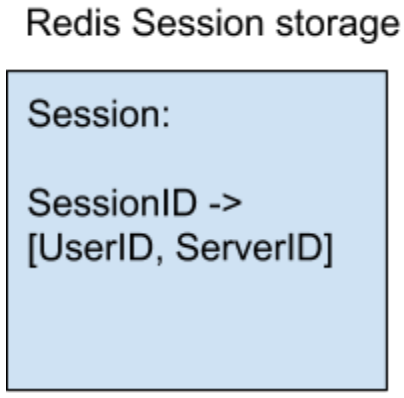
Prototype on this link:
https://xd.adobe.com/view/3700bb2e-e5dd-448f-630f-575082a0542b-4131/?fullscreen



Web Service Design
If your application is using Web Services via APIs, this deliverable should list the Web Service being used along with a description of the Web Service's API including the data being passed to and received from the API.

We are currently not using any web services.

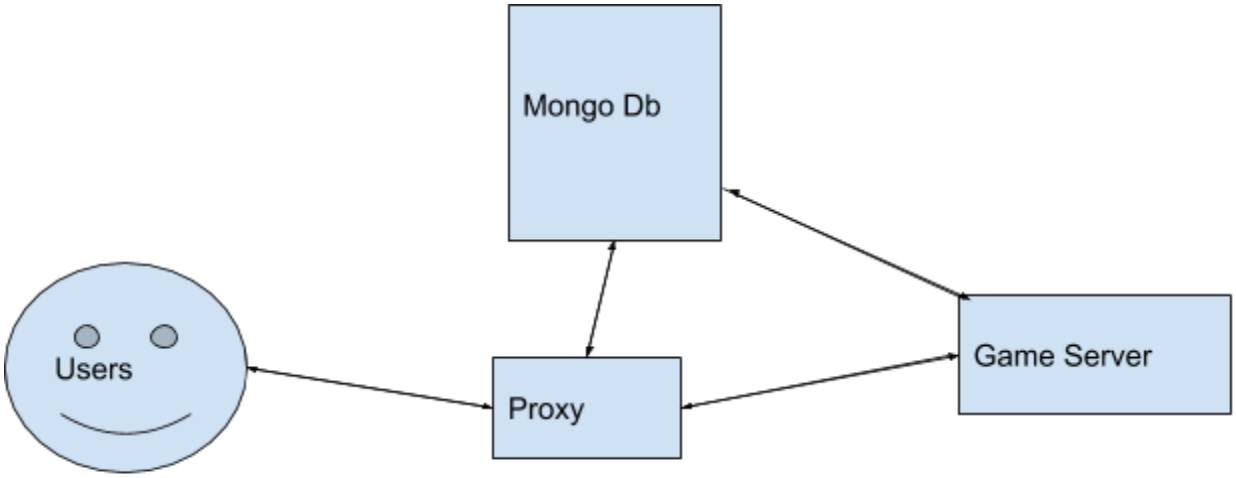
Database design
This deliverable provides a summary design of your application's database. The design document should identify each type of data being stored in your database. This may be documented in terms of a schema definition, showing data entities ("files") and attributes ("fields") This may be documented via an Entity Relationship Diagram showing database tables and columns. The document should identify the specific DBMS technology being used to store your application data (PostgreSQL, MySQL, Firebase, etc.)
We are using Redis for our session storage database, and we are using Mongo as a leaderboard database.



Mongo Database:

Data Stored - Top 50 player names and their scores stored in order from greatest to least

Architecture Communication:


Mongo Db
Users
Proxy
Game Server