Activity: Architectural Design Process – Iteration 1

Breakout room 11; November 12, 2021

Members Names	Student ID
Raphaiel Halim	100700318
William Robinson	100751756
Liam Rea	100743012
Phillip Jasonwiki	100751888

Step 1: Review Inputs

Category	Details
Design Purpose	To produce a sufficiently detailed design to implement
Primary Functional Requirements	For the use cases in figure 1, the primary ones chosen are:

UC-1: Customizable battles	It is one of the core components of the game; being able to choose what battle it is.
UC-2: Inventory	It is one of the core components; allows player to choose what items to use
UC-5: Combat mechanics	It is one of the core components of the game; without it battles cannot occur properly

Category	Details
Quality Attribute Scenarios	

Scenario ID	Importance to Customer	Implementation Difficulty
QA-1	Medium	High
QA-2	High	Medium
QA-3	High	High
QA-4	Medium	High
QA-5	High	High
QA-6	Low	Low
QA-7	Medium	Medium

From the list, only QA-2, QA-3, QA-5 are selected as the drivers

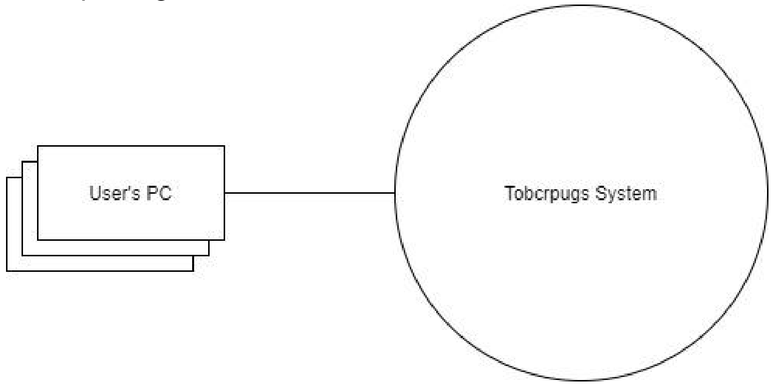
Category	Details
Constraints	All of the constraints discussed in figure 2 are included as drivers.
Architectural Concerns	

ID	Concern

CRN-1	Inter-connection between classes may become messy if careful steps are not taken
CRN-2	Delegation of work to each member
CRN-3	As classes are created, clear function descriptions are needed to ensure that they work when put together

Step 2: Establish Iteration Goal by Selecting Drivers

Drivers:
QA-2: Performance,
QA-3: Performance,
QA-5: Performance,
CON-2: Player can only hold specific amount of items (10-20 slots based on level)
CON-1: Max of 10 opponents created at a time
CRN-3: As classes are created, clear function descriptions are needed to ensure that they work when put together



Step 3: Choose One or More Elements of the System to Refine

Element to Refine: Tobcrpugs System

Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers

Design Concepts: Rich Client Application

We need a rich client application as most of our QA concerns are performance based. We want the user to interact with the system in a highly interactive manner. This program also needs no connection to the internet to function, so a server-client style application is not needed. Given the simplistic nature of the program, getting it to run on old hardware should be relatively simple as it only requires a few calculations and not much work in the way of graphics.

Discarded Alternatives:

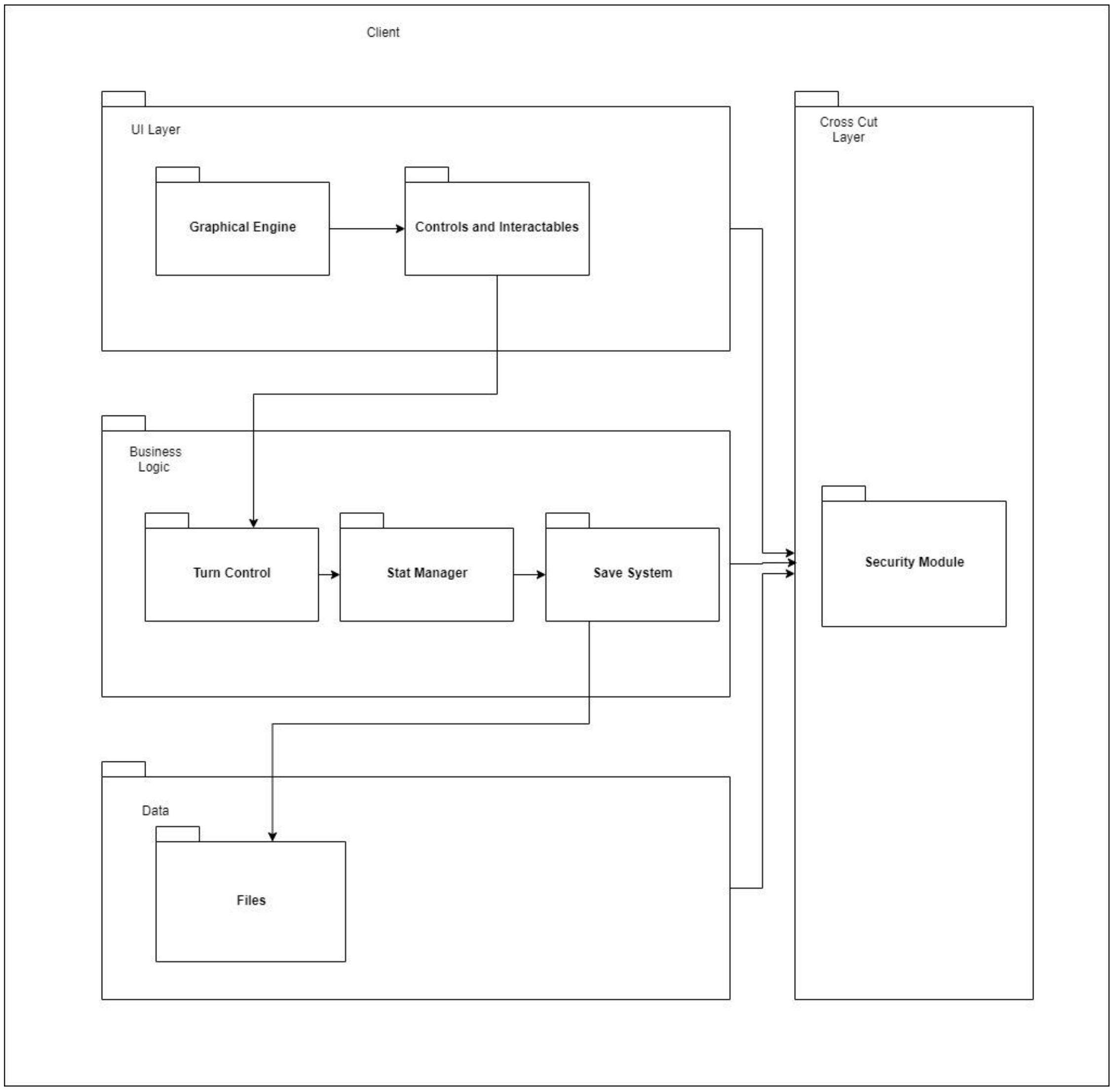
Alternative	**Reason for discarding**
Mobile Applications	Type of device was not considered for accessing the system
Web Applications	This reference architecture focuses on the application being accessed through a website. We prefer the application to be developed and run locally without the need of an internet connection.
Rich Internet application (RIA)	Group's lack of knowledge/experience in working with and developing Rich Internet Applications.

Build the game in Java
The framework allows for building a client rich application. With it being OOP based, it allows us to use many of the design patterns we are taught from the course.

Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

Design Decision	**Rationale**
All data will be stored locally	Given that we cannot guarantee a stable internet connection to the player, save data will be stored locally. This enables offline play and quick access times to the data that way players can play immediately.
Project will be open source	An open source project provides many advantages. Including; smaller budget, community support, easier innovation and better design. Since we are a small team it would make sense to keep this project open source for others to work on in the future.

Step 6: Sketch Views and Record Design Decisions



Element	Responsibility
UI Layer	Contains the different modules responsible for rendering the user interface and receiving the user inputs
Graphical Engine	The responsibility of the engine is to display what is currently going on in the business logic layer as a graphical representation
Turn Control	Sweeps through the state of battle to determine what the next move is
Stat manager	Calculates buffs and debuffs and holds int for time applied
Security Module	Verifies that a user’s moves are legal within the stat system and are not being edited to always do a fixed amount of damage.

Business Logic	Layer contains the modules that perform the turn control, stat manager and save system operations locally on the client’s machine
Save System	The system that handles the reading from and writing to file. Allows for loading of a saved game and the ability to save the current game
Cross Cutting Layer	A layer that permeates the entire program and can be accessed by any layer. Aspects of this layer are present in all other layers
Data	Layer responsible for communications with the machine and local save files

Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
	UC-1		Basic understanding with reason of importance to the system was created.
	UC-2		Basic understanding with reason of importance to the system was created.
	UC-5		Basic understanding with reason of importance to the system was created.
	QA-2		Importance and difficulty of quality scenario defined. Selected to be one of the main important drivers of the system.
	QA-3		Importance and difficulty of quality scenario defined. Selected to be one of the main important drivers of the system.
	QA-5		Importance and difficulty of quality scenario defined. Selected to be one of the main important drivers of the system.
CON-1			No relevant information decided or even discussed in iteration
CON-2			No relevant information decided or even discussed in iteration
		CRN-3	An understanding and description was created outlining the concern. Rationale was stated.

Appendix

Use Cases

Use Case	Description
UC-1: Customizable Battles	On the main screen there should be an option titled “custom battle” this will let the user select an existing character and customize the opponents are playing against OR generate a random character at a random level and play with them
UC-2: Inventory	The inventory screen should enable the user to “drop”, “inspect”, and “equip” items.
UC-3: Turn Based combat	At the start of every turn, turn order will be calculated based on speed and other variables (debuffs and buffs)
UC-4: Random Encounters	Based on the main party's level, enemies will be generated to be sufficiently difficult. Enabling a constant progression and easier creation of dungeons
UC-5: Combat mechanics	Combat should not be static, instead every action is based on the stats and status effects of the character that uses them. Damage should be calculated based on the stats and status effects of the character receiving the attack or spell
UC-6: Saves and Autosaves	The user should be able to save their game at any point, this will be stored in a text file that can be later loaded to continue the game. Also after certain key story events a file will be automatically saved
UC-7: Leveling System	Integrated in the character component, a leveling system should exist. Upon hitting a certain threshold the character gains stats and is able to distribute them as seen fit.

FIGURE 1: Use cases

Constraints

ID	Constraint
CON-1	Max of 10 opponents created at a time
CON-2	Player can only hold specific amount of items (10-20 slots based on level)
CON-3	Max of 3 actions could be performed per turn
CON-4	Minimum of 1 opponent generated with level scaling to player
CON-5	Minimum of 0 damage can be dealt to avoid calculation issues
CON-6	Max save file directory up to 1GB total allocated
CON-7	Max of 5 different character save files before data overwritten
CON-8	Max of 1 autosave files are generated
CON-9	A max player level of 30, after which no more levelling up for the player
CON-10	Minimum player level of 1

FIGURE 2: constraints