Iteration 1: Establishing Overall Website Architecture

The first iteration outlines the initial results of the design process for the ADD steps. In the first iteration, we are thinking very high-level. We are simply choosing what would be the best general reference architecture based on the drivers. We considered several, but ultimately decided that the **Rich Internet Application** was proven to be the most useful reference architecture for this application.

Step 2: Establish Iteration Goal by Selecting Drivers

Driver selection was based on what influenced the structure of the system. This helps establish the iteration goal of achieving the architectural concern of establishing an overall system structure. Drivers listed below:

Driver Table

Driver ID	Driver Content
QA-1	Security
QA-2	Availability
QA-3	Performance
QA-5	Modifiability
CON-3	The user must always be able to reliably play games that have been loaded, even when the connection is very poor
CON-8	When storing a user's info, it must be secret/encrypted and only the user that the data is referencing should have access to it.
CON-5	The website must be accessible by several web browsers (Chrome, FireFox, IE) and support use from multiple operating systems (Windows, Mac OS, Linux).
CON-6	Users must be able to open a game of their choice within three pages from any point within the website.
CON-7	The website will be up to date and running at all times for both low and high influx of players.
CON-12	Before a user can make changes to their account that are pushed to the server, the user must confirm the changes via a single pop-up window.

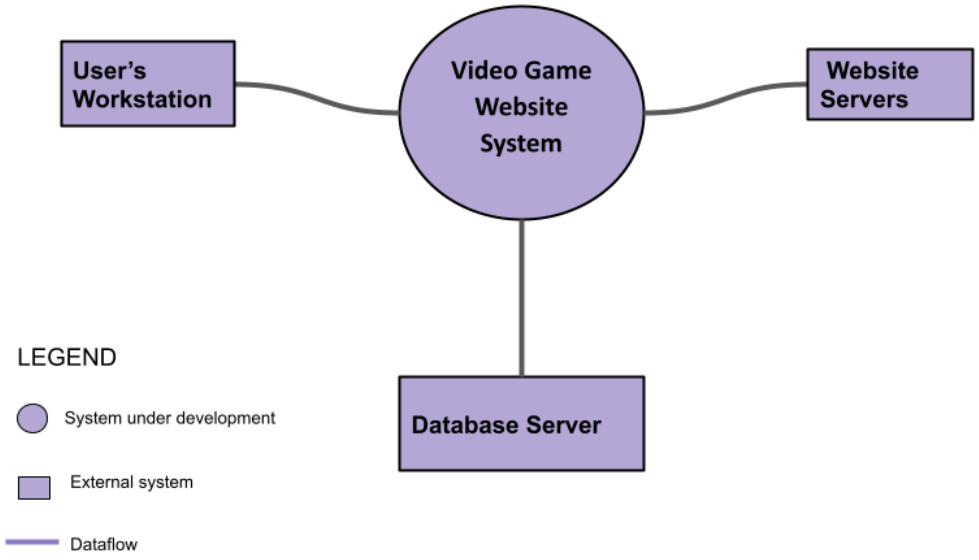

FIGURE 1: Context Diagram for the Video Game Website System

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

This is a brownfield development effort, so in this case the element to refine is the entire video game website system, which is shown in Figure 1. In this case, refinement is performed through decomposition.

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

The following table summarizes the selection of design decisions

Design Decisions and Location	Rationale
Logically Structure the client and server part of the system using the Rich Client Application reference architecture.	The Rich Internet Application (RIA) supports applications that typically run inside the browser of a user and tend to be a bit more complex than a generic web application. RIA's are very useful for web applications that need a rich user interface, which most video games require. They also allow for some processing to be done on the client-side, meaning that we can efficiently use both the client and server in order to create the smoothest user-experience possible (CON-3). This will also support QA-3 such that we can allocate certain resources to the server or client as needed. Having a rich user interface also allows us to track how a user

interacts with the website better (UC-1).

Discarded alternatives:

Alternative	**Reason for Discarding**
Web Applications	This reference architecture is mainly used for applications where most of the application resides on the server-side and a rich user interface is not needed. For a video game hosting website, client processing capabilities and the ability to support a rich user interface for various games is necessary.
Rich Client Application	Though this reference architecture supports a responsive and interactive user interface and intermittent network connection, which would support several drivers well, it does not have any web connectivity.
Service Applications	This reference architecture does not provide a user interface but rather expose services that are consumed by other applications. Therefore the use of this application is

		used by humans so the need of a user interface would be necessary. Plus it would cause more complications as well.
	Mobile application	This reference architecture is oriented toward the development of applications that are handheld devices. This alternative was discarded because there are many limitations that cause this option to not be viable nor practical, such as the screen size, and the resources that would be available.

Framework Decisions:

Design Decision and Location	**Rationale**
Build out the backend using Django to serve the web pages that are built using ReactJS	Django and React are both very widely used frameworks, which allows the dev team to build out the website quickly. Django also has many built-in security protections (QA-1)
Use Heroku to deploy the website	By deploying the website through a provider, we can get access to their dynamic allocation depending on the amount of users we have (CON-7). Heroku also has various built-in tools for measuring website performance that can be easily implemented (UC-7, UC-8).

Step 5: Instantiated Architectural Elements

For the most part, the RIA reference architecture suits our use well. The only difference is that we will be making use of the Isolated Storage.

Design Decision and Location	**Rationale**
When possible, store data locally in the client	The website must be playable even when connection is poor. We can save data that is needed to continue the

	game locally even when internet connectivity is poor (CON-3). Following the reference architecture, we can adapt the optional Isolated Storage as a temporary data store while internet connectivity is poor.

Step 6: Sketch Views and Record Design Decisions

The diagram in Figure 2 shows the sketch of the module view of the two reference architectures that were selected for the client and server applications. These have now been adapted according to the design decisions we have made.

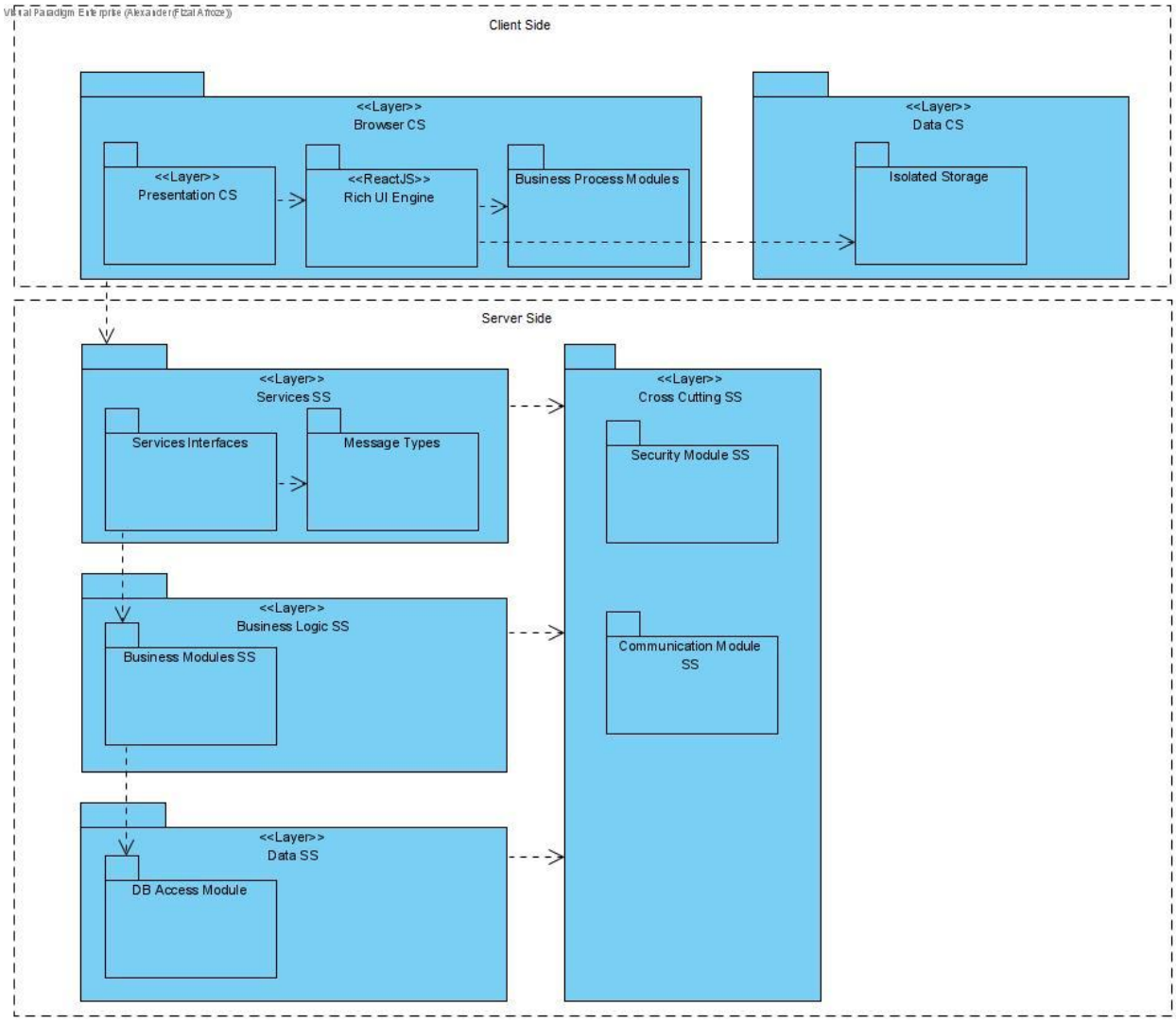

FIGURE 2: Sketch of a module view of client side and server side architecture

Each of the elements that have been selected is explained with a short description of its responsibilities. The following table summarizes the information that is captured in Figure 2:

Element	Responsibility
Browser CS	This layer contains modules for how the user interacts and manipulates the website
Presentation CS	This layer contains modules that control how the user can interact with the UI
Rich UI Engine	Contains modules that are responsible for more sophisticated UI interactions
Business Process Modules	This module contains modules that can perform business processes that can be executed on the client
Data CS	This layer is responsible for holding data on the client side
Isolated Storage	These modules are responsible for containing data that allows for users with intermittent internet to continue playing
Services SS	This layer contains modules that can expose services for the client use
Services Interface	This module contains services that are exposed for client use
Message Types	This module contains services that are specific to managing the types of messages that a client and server exchange
Business Logic SS	This layer contains modules responsible to implement business operations and other business logic
Business Modules	This module is responsible for implementing business operations and other business logic
Data SS	This layer contains modules responsible for data storage and retrieval for other layers
DB Access Module	This module is responsible for communication between the Server and the Database
Cross Cutting SS	This layer contains modules that range in functionality from security to various communications that are needed between layers
Security Module	This module is needed for implementing various security measures such as encryption, logging, etc.
Communication Module	This module is responsible for various communications between various layers

Figure 3 shows the initial deployment diagram and where elements from figure 2 are supposed to be within the deployment diagram. Also, the table below lists the responsibility of each component for the initial deployment diagram in figure 3.

Element	Responsibility
User workstation	The users devices that can host the client side logic of the system.
Application Server	The server that can host the server side logic of the web application.
Database Server	The server that contains and hosts the relational database of the system

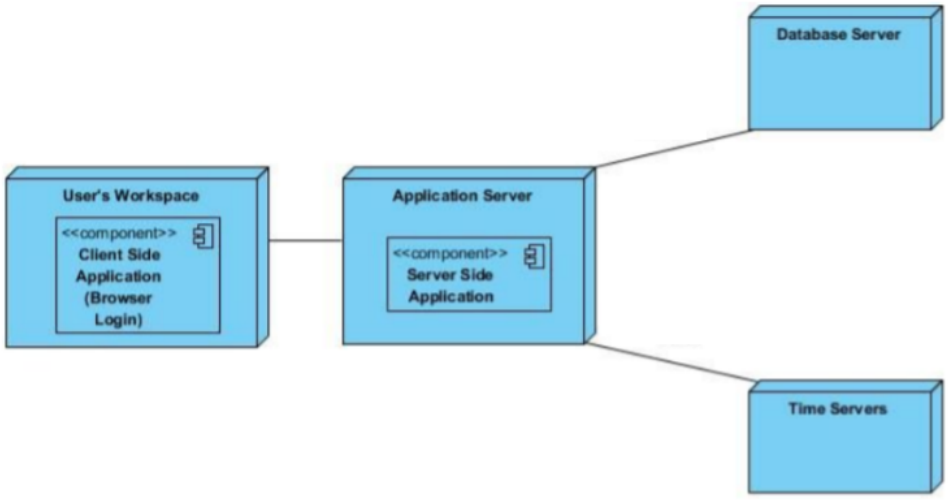

FIGURE 3: Deployment diagram for the video game website system

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 Iteration

	QA-1		The security module implemented in the selected reference architecture that supports this functionality
QA-2			No relevant decisions made
QA-3			No relevant decisions made
QA-5			No relevant decisions made
	CON-3		The Data SS layer is implemented in the selected reference architecture that supports this functionality. More information about how the data is saved must be decided
	CON-8		The security module implemented in the selected reference architecture that supports this functionality.
		CON-5	Using Django along with ReactJS should allow for this website to run on all popular browsers
CON-7			No relevant decisions made
CON-12			No relevant decisions made
		CRN-1	Using Django will allow for the older web pages to be served alongside the newer web pages with minimal new interfacing needed. In addition to this the reference architecture was selected.
		CRN-2	Django and React have been selected and should encapsulate all uses within the Package Diagram