Iteration 3: Addressing Quality Attribute Scenario Driver (QA-6)

This section provides the results preformed in each step of the ADD method in the third iteration of the design process. Expanding the decisions made in the primary and secondary iteration the third iteration focus on the quality attribute of usability.

Step 2: Establish Iteration Goal by Selecting Drivers

For this iteration the architect focuses on the quality scenario 6 QA-6: The user desire functions that make searching for a game more informative. The functions must operate as intended, and if not, an error is sent to the maintenance technician.

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

For the usability scenario, the elements that were decided by the architect to refine were the:

- Web application system
- Database server
- API used for scraping game information

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

Design concepts in this iteration are listed below:

Design Decisions and Location	**Rationale and Assumptions**
Implement the python script to scrap information from the database to create views	By using python and linking it to the MYSQL workbench database the system can create views based on the tuples provided in the tables the script can also be used to reference multiple tables from the database to create a multi table view with different columns from other tables.
Using Django web framework tables from the database can be converted into classes	These classes will aid in the construction of different variants of the primary web page in the web application.
Introduce elements from the message queue to allow trap technology	Traps received from the web servers are placed in a message queue and then retrieved in the application. The traps are processed and delivered to ensure that the functions of the system built of python and Django are properly functioning

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

The instantiation decisions are explained in the table below:

Design Decisions and Location	Rationale
Deploy a message queue	Deploying a message queue will ensure that no traps are lost in the case of failure in compiling the system’s various pages this is essential as in the case of failure caused by the traps not being saved, it can cause information not to be presented in the destination it is intended to be within
Use python and Django to compile the known views featured in the database server	The implementation of Django and python allows the system to be able to combine tables with simpler code than within the database server by referencing foreign keys within the tables.
Using the API scrap information and transfer it into the database to automatically fill tuples	The are two APIs featured in the Game_Knight system find and add information automatically into the MYSQL workbench database. This information can be further developed into pages featured in the web application.

The results of the design decisions are further inspected in the next step of the iteration process.

Step 6: Sketch Views and Record Design Decisions

Figure 1.9 shows a refined deployment diagram that includes decisions listed in the instantiation above

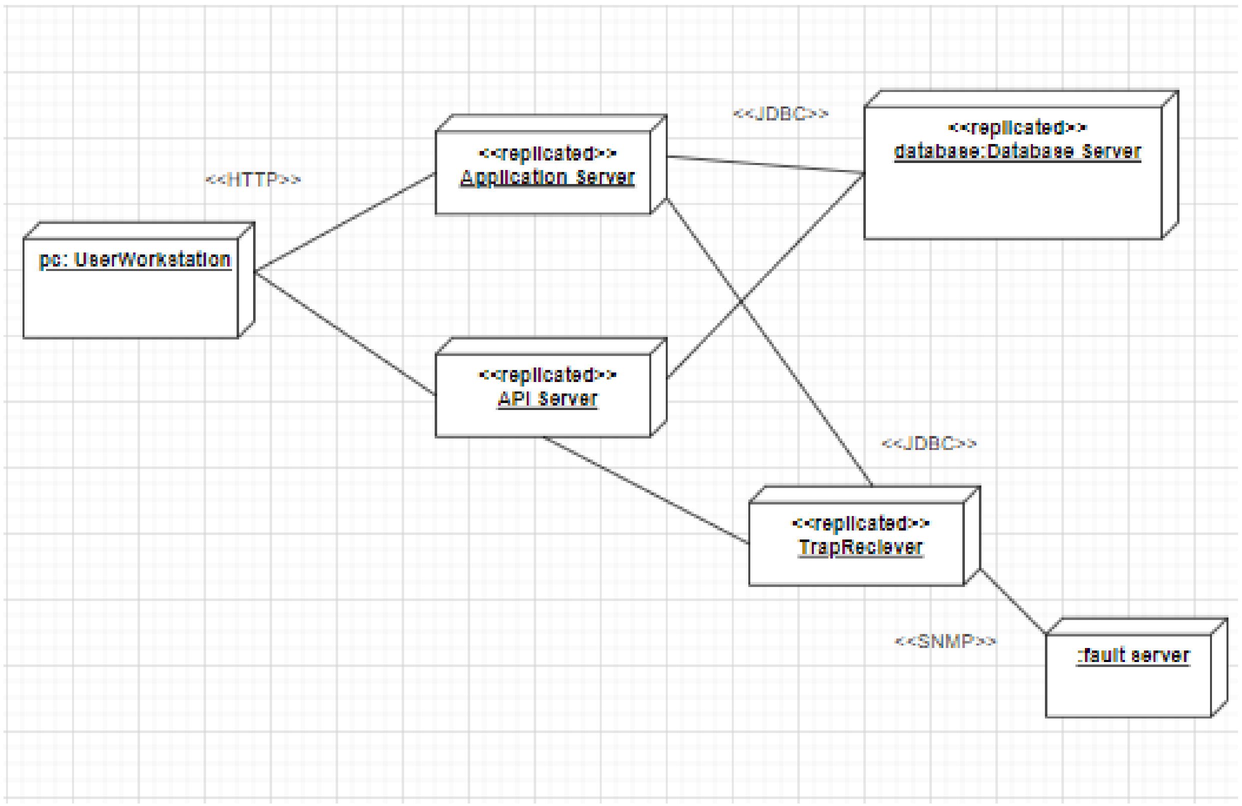

FIGURE 1.9 Finalised Deployment Diagram

The tables below explains the responsibilities for the elements added to the updated version of the deployment diagram

Element	Responsibility
API server	The API server was left out of the first iteration's deployment diagram as it was an incomplete the API server is important to the system as it adds information scrapped online to the tables within the database it is used to create views of tables with sampled information.
TrapReceiver	The TrapReceiver gets traps from the system and converts it into events that can be used by the technician to improve the system.

The UML sequence diagram shown in figure 1.10 represents how the TrapReceiver introduced in this iteration operates within the system and creates events to be further understood to how the system functions. This sequence diagram is associated with UC-2 (detect function operability), which is associated with the quality attributes QA-3 (performance) and QA-5 (testability).

The purpose of the sequence diagram is to represent the communication that occurs between the TrapReceiver and other lifelines.

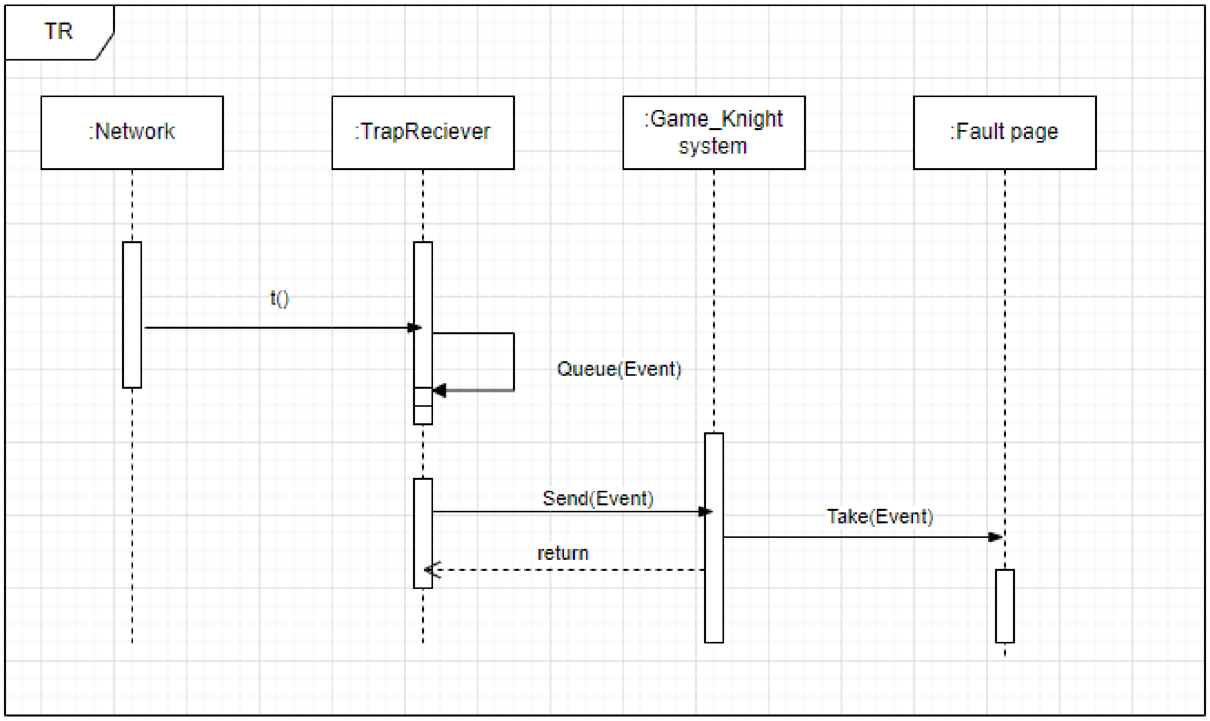

FIGURE 1.10 TrapReceiver interaction diagram

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

In the third iteration, important design decisions have been created to address QA-6 usability, which also impacts QA-2 Modifiability, QA-3 Performance, the table below expands on the different drivers and the decisions involved in this iteration. Design decisions completely addressed in the previous iteration are not referred in the table below.

Not Addressed	**Partially Addressed**	**Completely Addressed**	**Design Decisions Made During the Iteration**
	QA-1		The Availability of the system was decided to be at all hours depending on if the other servers are functioning the way they are intended to
	QA-2		The modifiability of the system is dependant on the API and Django application systems if the

	information is present in the Django classes the script can make tuples.
QA-3	The performance of the system is ensured if the TrapReceiver functions and 100% of the traps are processed, even in the situation of failure.
QA-4	As the security of the website is not necessary it will no longer be addressed
QA-5	The TrapReceiver ensures the testability of the system
QA-6	The usability quality attribute is improved with the improvements of Modifiability and Performance quality attributes
CON-2	No Relevant decisions made
CON-3	No Relevant decisions made
CON-4	Use of the save tactic to manage data logging will aid in this constraint as it is required in the greater implementation.
CRN-1	
CRN-4	No Relevant decisions made