Software Technical Requirements Document

AI-based Tool for Resources Allocation

Imen Souissi

Grand Canyon University: SWE-540

October 19, 2022

Document Version v1.0, v2.0

October 19, 2022

**AUTHORS**

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Department** |
| Imen Souissi | Developer | Development Department |

**DOCUMENT HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Document Revision Description** | **Document Author** |
| August 24, 2022 | 0.1 | Initial Document Creation | Myself |
| September 24, 2022 | 1.0 | Review document, modify policy section reflecting organizational legal - contractual service agreements for the protection of user information including ISO, PCI DSS | Myself |

**APPROVALS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Approval Date** | **Approved Version** | **Approver Role** | **Approver** |
| September 28, 2022 | 1.0 | Mentor / Adviser | Professor, Mohamed Mneimneh |
| October 19, 2022 | 1.0 | Mentor/Adviser | Professor Ali Wahid |

# Introduction

This project entitled “AI-based Tool for Resources Allocation” may help organization to efficiently allocate its resources using an AI-based technology.

This project was designed to address these issues and consists of two phases. The first phase utilizes an Artificial intelligence technology to implement the core of the project. The second phase will be a testing phase while feeding the model with appropriate data.

## **Purpose of the document**

Technical Specification Document includes the most relevant aspects of a product from its development to its correct use. In order to create a technical requirements document, a technical writer combines complex information and simplifies it into an engaging informative and easy to understand document.

Brief, a technical requirement document compiles the entire product development work flow Besides, it provides explanation of the product functionality, use, characteristics and purpose.

## **Project Scope**

AI-based tool for resources allocation will develop and implement artificial intelligence-based technology for all kind of software organizations with the objective of helping them to successfully optimize their resources allocation - assigning and managing assets in a manner that supports an organization's strategic planning goals - resources can be both human (employees) and electronic (software, computers, other devices).

Briefly, the purpose of this project is to develop an artificial intelligence-based tool that allows software organizations to effectively allocate resources. The goal is to mitigate risks associated to identifying right priorities.

In Scope Features:

* Implement and set up an artificial intelligence-based technology to assign and manage resources (human and electronic).
* Create a SaaS (Software as a Service) based product.

Out of Scope Features:

* This will not include email campaign or any kind of marketing.
* This will not include internal communication automation.

## 

## **Related documents**

|  |  |  |
| --- | --- | --- |
| **Component** | **Name (with link to the document)** | **Description** |
| Functional Specifications document | GitHub Link: <https://github.com/Imen-Souissi/functional_spec> | A functional Specification Document |

## **Terms/Acronyms and Definitions**

|  |  |  |
| --- | --- | --- |
| **Term/Acronym** | **Definition** | **Description** |
| AIBTRA | Stands for AI-based Tool for Resources Allocation. | The name of the application |
| AIBTRA users | They are the team of engineers that need to allocate some specific resources to start a project or perform certain task. | The users of the application. |
| A proposal | It is a request of hardware material - such as servers, switchers, etc., - or software tool – such as license, ticket, etc. - this proposal needs to be reviewed by certain department and then either gets approved - in this case a labview is created - or denied. | Allocation process |
| A LabView | It is the output of an approved proposal - requesting certain hardware to work on certain case by a team of engineers. | Allocation response |
| Lab Team | The group of technicians that work in the laboratory. | System users |
| VM | The virtual machine | Virtual machine |
| VMC | View Model Controller paradigm. | The application architecture pattern |
| Zend Framework2 | The framework used to build the application | A framework |

## **Risks and Assumptions**

* The product is designed to run on a cross- platform but it might have technical issues when running on Linux.
* The product follows an MVC paradigm which has to be friendly with the Machine Learning PHP libraries.
* The product has high priority within the company, therefore the utilization of resources required will be constrained by this.
* The product has to be maintainable for 15 years.
* The product has to be able to serve thousands of companies.
* The product has to be available 24/7.
* The product has to run at a short response latency.

1. **Architecture**

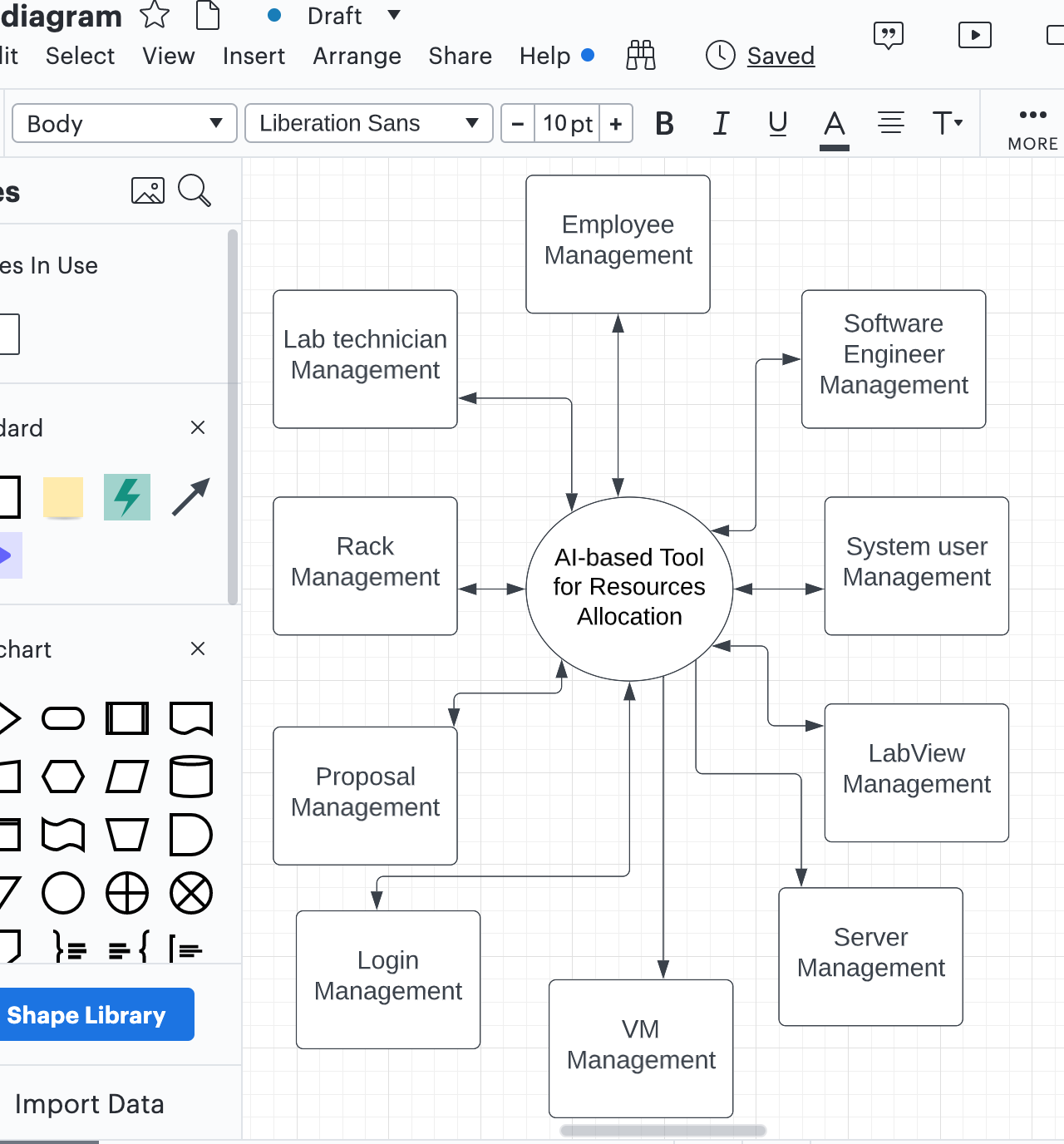
It is an N-tier architecture which divides the application into logical layers and physical tiers. Layers are a way to separate responsibilities and manage dependencies. Each layer has a specific responsibility. A higher layer can use services in a lower layer, but not the other way around.

Tiers are physically separated, running on separate machines. A tier can call to another tier directly or use asynchronous messaging (message queue). Although each layer might be hosted in its own tier. Several layers might be hosted on the same tier. Physically separating the tiers improves scalability and resiliency, but also adds latency from the additional network communication.

This is a traditional three-tier application which has a presentation tier, a middle tier, and a database tier.

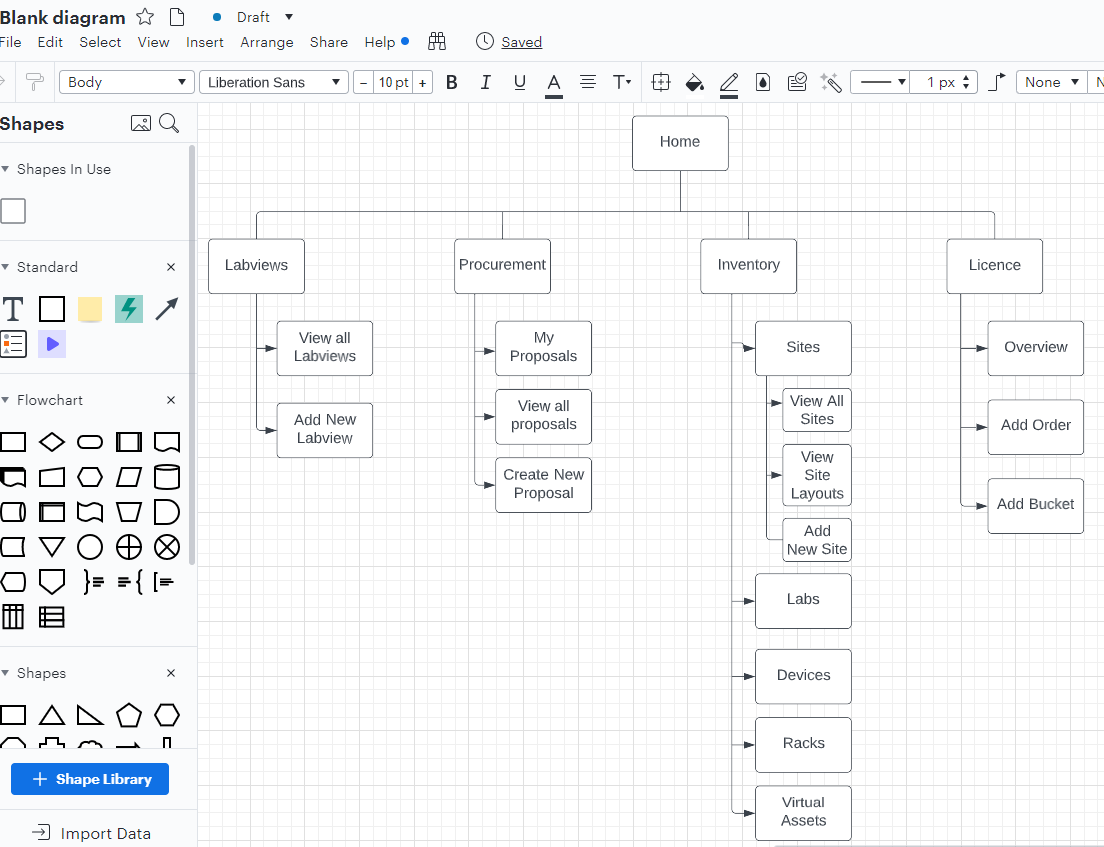
## **System Architecture**

**Data flow Diagram: Using Lucid chart Tool**



**Sitemap: Using Lucid chart Tool**

According to the article “Why You Should Build A Sitemap Before Designing Your Site” (n.d.) A sitemap can be an effective planning tool for both designers and non-designers alike. It’s a centralized planning tool that can help organize and clarify the content that needs to be on your site, as well as help you eliminate unnecessary pages. And a sitemap, because it’s basically just an outline or flow-chart of the content your site needs, can be created by anyone, regardless of their design skills. Read on for more reasons why a sitemap should be the starting point for your new website design.



## **Application Architecture**

AIBTRA application follows an MVC architecture pattern – MVC pattern is commonly used for developing user interfaces that divide the related program logic into three interconnected elements the model, the view and the controller – besides, AIBTRA application is an SSR application - Server-side rendering (SSR) is an application's ability to convert HTML files on the server into a fully rendered HTML page for the client. Usually, SSR is recommended for apps in which you have to pre-render frequently updated data from external sources.

## **Hardware Requirements**

For a basic user that will use the system like it is intended to be, the minimum requirement for a computer to be able to run AIBTRA application would be simply the normal required tools to connect to internet and an updated browser. Let us note that credentials are required to log in into the application.

Generally, we recommend 16GB RAM to be able to login into the application, download spreadsheets and make other type of operations.

The user needs to connect to the local network of the organization via a VPN - Virtual Private Network – tool when he is not in the building and can’t get access to the organization network. In this case we are using GlobalProtect - provides a unique mobile security solution by integrating traditionally distinct technologies, to manage the device, protect the device and control the data.

As for server, we are using Zend server - It includes a comprehensive set of application performance monitoring features that help to give users a complete view of how the application and server are performing during development or production.

## **Software Requirements**

We are using an open-source PHP framework, it is pure object-oriented and built around the MVC design pattern. Zend framework contains collection of PHP packages which can be used to develop web applications and services.

Besides, Zend Framework 2 uses 100% object-oriented code and utilizes most of the new features of PHP 5.3, namely namespaces, late static binding, lambda functions and closures.

Zend framework follows an SSR approach.

Similarly, as for server, we are using Zend server - It includes a comprehensive set of application performance monitoring features that help to give users a complete view of how the application and server are performing during development or production.

## **Software**

The application and software designs must be done with PHP, HTML, jQuery, JavaScript and MySQL.

## **Documentation / Comments**

When writing the code, a documentation module should be present under the root of the application. Besides, commenting out important notes (such as hard coding parts, note for future developer, etc.)

## **Error Checking / Handling**

- The application must handle exceptions and errors to prevent application crashes.

- The application must:

- Use exception handling code (try/catch blocks) appropriately.

- All data going into a database table should be checked to confirm the data is in the appropriate format. If it is not, try and convert it. If error or exception still occurs, flag the user of the error.

## **Software Versions and Storage**

## For software versions and storage, I used Git.

## **Data Requirements**

### This application follows an MVC – Model View Controller – paradigm, The model is the layer that keeps data for the application. The layer is responsible for storing and retrieving data from the database, and it is also responsible for validating the data. For example, if you want to create a new user, you will have to create a new user model and save it to the database.

## **File Structure:**

## The file structure follows the Zend framework 2 file structure, so the framework can update the files after any changes or creation of new modules. To follow the framework file structure, you have to go back to the Zend framework 2 documentation page and create the same given structure in order to add a new module.

## **Extract Functionality**

In our case, this feature doesn’t exist yet, I’m trying to work on it by extracting data from the Salesforce platform – precisely extracting new cases that the team used to do it manually - and get it into the application by creating data pipelines.

## **Export Functionality**

Exporting data in electronic format when it’s needed by some other system or organization. The data is either needed in real time or can be processed as a batch job. In this case our system allows the user to export data as Excel files.

## **Schema Objects**

We are using MySQL database. The application is running xtradb 5.7. Similarly, we are using PDO\_MySQL – PDO acronym for data objects, it is a data access layer which uses a unified API – which is a driver that implements the PHP Data Objects (PDO) interface to enable access from PHP to MySQL databases. PDO\_MySQL uses emulated prepares by default.

## **Data Mapping**

The data mapping is used at the model level, when creating a model file called nameOfModule.php to map our variables to the data base existing variables, this is using PHP PDO driver which is a lightweight, consistent framework for accessing databases in PHP.

## **Security**

Employees shall be forced to change their password the next time they log in if they have not changed it within the length of time established as “password expiration duration”

Password shall never be viewable at the point of entry or at any other time.

The access permission for system data may only be changed by the system’s data administrator.

## **Communication Interfaces**

This application doesn’t require communication with another ever/system however the user has to enter the information related to new cases from salesforce interface.

1. **System Configurations**

To build this application I have started with creating a “Zend framework 2” project, this framework like all the other application frameworks supplies tools and ready configurations that gives shape and structure to the overall application, besides it requires a specific directories/files hierarchy for the three main components of this MVC application which are Model, View and Controller, the “Zend framework 2” documentation provides all the required elements. On the top of that it provides a copy/paste ready configurations for the module.php, module.config.php, autoloader files, etc. Almost all the required files for a new module are provided by the framework which makes the configuration process a bit smoothy.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Button, Link, Icon Label** | **OnClick Event** | **Other Event** | **Visible** | **Enabled Vs Disabled** | **Navigate To** | **Validation** | **Dependencies** |
| Login | Authentication that has to verify the type of credentials the user has in order to log in into the application: admin credentials, other | OnMouseHover display the required fields to sign in | Yes always | Enable the entry of the first field otherwise it is disabled | Home page of the user account | Verify if the credentials are valid, username and password are confirm with database | If User logs in so the new user button should be disabled |
| Create new proposal | A form should be completed | Display the required fields of the form | Visible only when the user logs in | Enable the entry of all the fields, disable the submit button until all required fields are filled out | A form | Verify if all the fields have been filled out | This functionality should be an admin privilege |

1. **Integration Requirements**

:

Integration requirements indicate the integration of organizations, processes, data or systems. The following is the most important features of our system integration requirements:

* + Application integration is done through SOAP and REST services.
  + Large capacity for data integration and storage through a data lake or cloud-based data warehouse
  + Integration will be able to support the present and coming data velocities whether batch or streaming data
  + Event-based integration over clock-driven integrations
  + A document-centric data integration strategy
  + Hybrid integration, including for both cloud-to-cloud and cloud-to-ground scenarios.
  + Integration must be accessible via REST/SOAP APIs
  + Instead of location, new integrations must be focused on connectivity for speed and agility
  + Elastic integration able to operate in real-time and fluctuating event-based demands
  + SaaS-based approach wherein integration is delivered as a service to anyone who needs access

1. **Performance**

:

Performance requirements indicate how quickly a process or service works, particularly when volumes are at their peak. In order to assess the performance of our system the following must be clearly specified:

|  |  |
| --- | --- |
| • Response Time | 0.1 second is about the limit for having the user feel that the system is reacting instantaneously, meaning that no special feedback is necessary except to display the result. |
| • Workload | The backup should be completed overnight otherwise it may seriously disrupt the performance experienced by the users the next day.  A DB system may easily handle 10,000 read transaction per hour but only 3,000 update transactions per hour. |
| • Scalability | We expect to have more accurately defined workloads which will require more hardware. |
| • Platform Considerations | We are using the same platform - operating system and software utilities – that the company provided for the old system: same processor and resource requirements for the old system, so we are going to be able to assess the amount of free space and performance. |

1. **Software Quality**

:

According to Chappell (n.d.) But what exactly is software quality? It’s not an easy question to answer, since the concept means different things to different people. One useful way to think about the topic is to divide software quality into three aspects: functional quality, structural quality, and process quality. Doing this helps us see the big picture, and it also helps clarify the trade-offs that need to be made among competing goals.

The three aspects of software quality are functional quality, structural quality, and process quality.

|  |  |  |
| --- | --- | --- |
| **Functional Quality** | **Software Quality** | **Process Quality** |
| Meeting the specified requirements: the product meets the specified requirements | Code testability: The code is organized in a way that makes testing easy. | Meeting delivery dates: Our application has met the deadlines according to Jira timeframe. |
| Creating software that has few defects: This software works properly; minor defects are in process of reviewing then fixing. Including duplicated notifications, some hardcoded parts of the code itself. | Code maintainability: The code has been refactored into separate logic units such as module, each module has its own MVC layers, so testing is very easy. | Meeting budgets: Our application does not have a budget as it is a university project. |
| Good enough performance: the overall performance of the system is very acceptable | Code understandability: The code implements a Zend framework which gives a very clean and clear structure of all the modules. The modules’ titles reflect the main function of the module itself. | A repeatable development process that reliably delivers quality software: It is a university project that has been developed by a single developer, so no stressing factors or anything. The process is running very smoothly. |
| Ease of learning and ease of use: the Zend framework and PHP programming language made it very easy to learn and use the application through a very detailed documentations and simple user interface. | Code efficiency: The code has been efficiently written. |  |
|  | Code security: The software is protected enough to be attacked or to receive any kind of common attacks. |  |

1. **Open Issues:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **::**  **Issue ID** | **Issue** | **Raised By** | **Raised On** | **Solution/ Decision** | **Resolved By** | **Resolved On** | **Status** |
| 1 | Duplication of a report that has to be auto - generated when creating a new labview | Previous Developer | 2021 | Spot the code then try to apply the Separation of Interest Principle | Myself | 2022 | Processing |
| 2 | The auto generated labviews should be archived not appear in the list of the created labviews. | Lab Team | 2021 | Implement new feature that can support this functionality | Myself | 2022 | Processing |

# 

# Appendix

**A proposal:** It is a request of hardware material such as servers, switchers, etc,. this proposal needs to be reviewed by certain department and then either gets approved - in this case a labview is created or denied

**A LabView:** It is the output of an approved proposal - requesting certain hardware to work on certain case by a team of engineers

**Lab Team:** The group of technicians that works in the laboratory.

**References:**

Chappell, D. (n.d.). *The three aspects of software quality: Functional, structural, and process.* http://www.davidchappell.com/writing/white\_papers/The\_Three\_Aspects\_of\_Software\_Quality\_v1.0-Chappell.pdf

Why you should build a sitemap before designing your site. (n.d.). https://neilpatel.com/blog/build-a-sitemap/#