A.O.K. Laptop System Overhaul

Course Project: IS 436 Structured Systems Analysis and Design

Deliverable 4 – “Data Modeling and Starting Design”

11/20/2019

Team Name: Meticulous Evolution Consulting

Project Sponsor: Library Services Manager, Paula Langley.

Presented By:

|  |  |  |
| --- | --- | --- |
| **NAME** | **POSITION** | **CONTACT** |
| Upen Adhikari | Quality Assurance | [adh3@umbc.edu](mailto:adh3@umbc.edu) |
| Omar Al-Hedari | Project Manager | [Omara2@umbc.edu](mailto:Omara2@umbc.edu) |
| Nima Roomi | Lead Developer/Programmer | [nimar1@umbc.edu](mailto:nimar1@umbc.edu) |
| Alex Varghese | Database Administrator | [varghes1@umbc.edu](mailto:varghes1@umbc.edu) |
| Ralu Ofoche | Systems Analyst | [rofoche1@umbc.edu](mailto:rofoche1@umbc.edu) |

**Table of contents**

**Entity Relationship Diagram 2**

**Description of Entities 3**

**Data Flow 4**

**Individual Alternative Matrix 5**

**Team Matrices 7**

**Team Choice 8**

**Architecture Matrix 9**

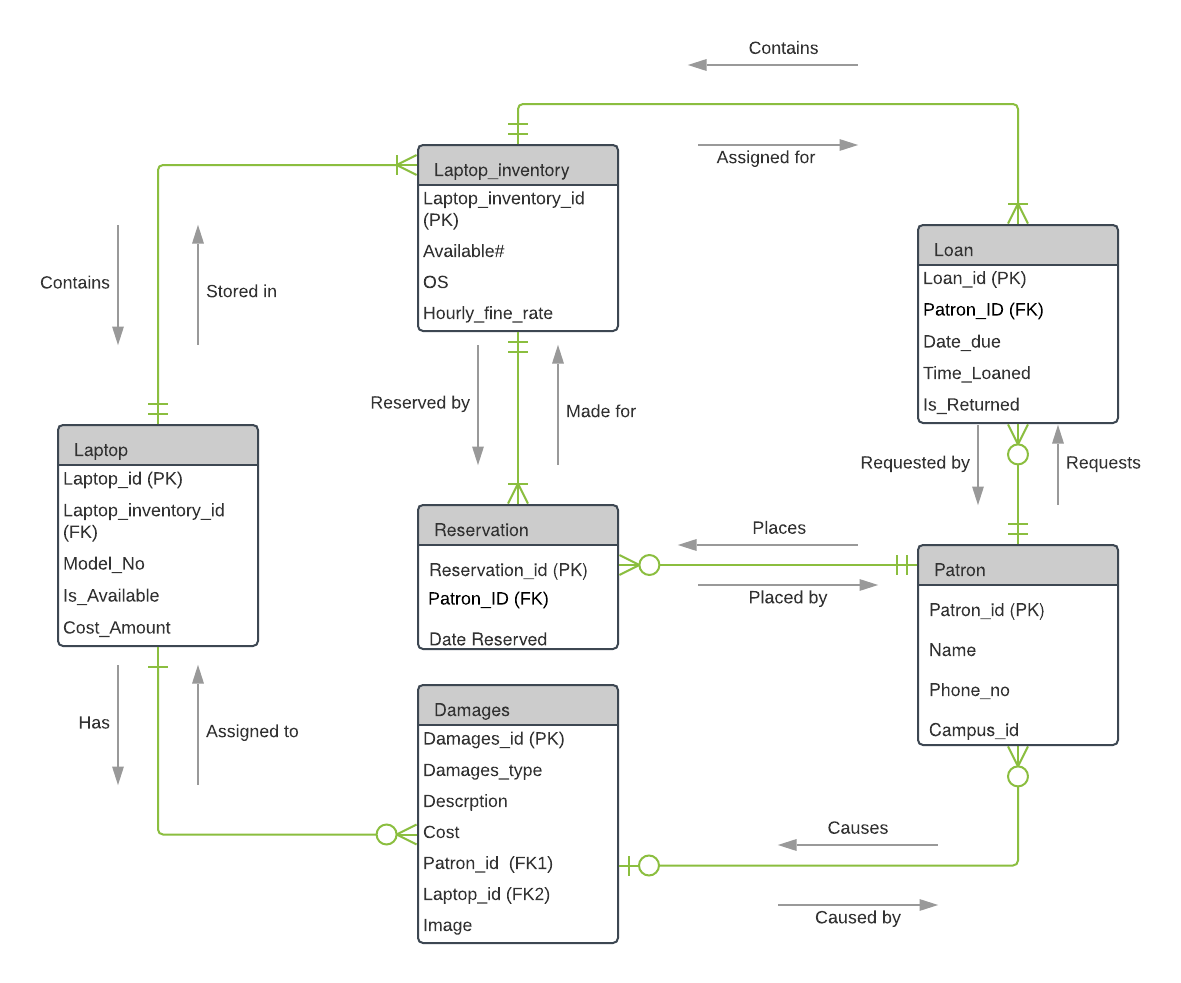
**Hardware and software specification 10**

**Integration with UMBC App 11**

**Kanban Board(Task Distribution) 11**

**Detailed Project Work Plan 12**

**Entity Relationship Diagram:**

****

**Entities:**

Patron:

In this ER diagram, the patron is the entity that requests a laptop by making a reservation. The patron has to have registration details to begin the process of the loan. This entity holds patron\_id, patron’s name, registration\_id and the phone number.

Damages:

This entity holds damages information about the laptops. It includes damages description, cost, images, and information about the patron who damages the laptop.

Reservation:

This entity holds the reservation details of the laptop including patron\_id, reservation date and time.

Laptop:

This entity holds the details of the laptop. So, each laptop will be identified using the unique laptop\_id. It also holds the model no, laptop cost and the availability of the laptop.

Loan:

This entity holds the loaning information of the patron including loaning date and return date. The loaning\_id will be the primary key to identify the loaning details of the customer and patron\_id will be the foreign key to identify who had applied for a loan.

Laptop Inventory:

This entity holds an inventory of the available laptop, OS, and hourly\_fine\_rate.

Relationships:

Requests/requested By: This relationship is between the patron and the loan. The cardinality between patron and the loan is one many. The patron may request many loans or no loans at all. In requested by relationship, the loan is requested by the patron. The cardinality between patron and the loan is many to a mandatory one.

Causes/caused by:

This represents the relationships between patron and damages. The cardinality between patron and damages is many to an optional one. This means that the patron may cause at least one damages or no damages. But, the damage is caused by the patron.

Has/assigned to:

This is the relationship between damages and patron. The cardinality between laptop and damages will be one to many as a laptop can have many damages or no damages. Similarly, the damages will be assigned to at least one laptop.

Stored in/contains:

This is the relationship between laptop and laptop\_inventory. The cardinality between these two entities has mandatory to many relationships as laptop\_inventory stores one laptop or many laptops. Similarly, laptop inventory contains at least one laptop.

Contains/assigned for:

This is the relationship between laptop\_inventory and the loan. The cardinality between laptop\_inventory and loan is mandatory to many as laptop inventory will have to have or assigned for many loans or at least a loan.

Reserved by/made for:

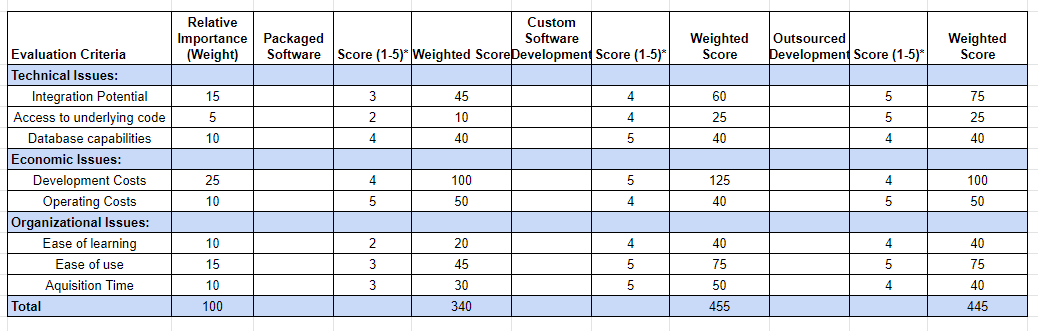
This is the relationship between the laptop\_inventory and the reservation. The cardinality between these two entities will be mandatory to many as the reservation is made for laptop\_inventory.

Places/placed by:

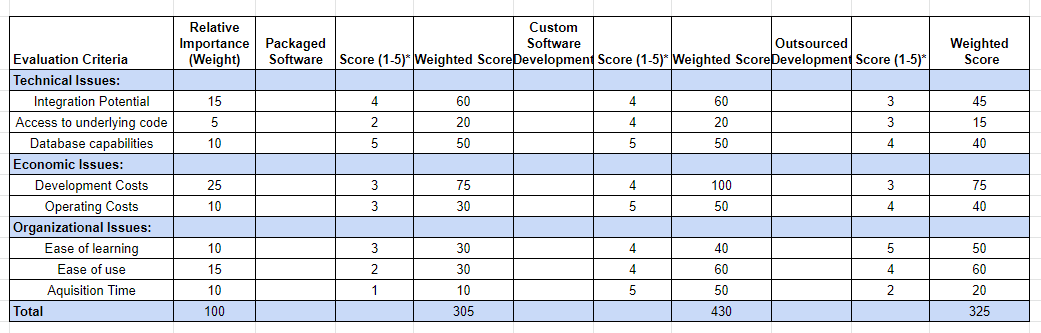
This is the relationship between patron and reservation. The cardinality between reservation and patron is mandatory to many. The reservation is placed by the patron and the patron places the reservation of the laptop.

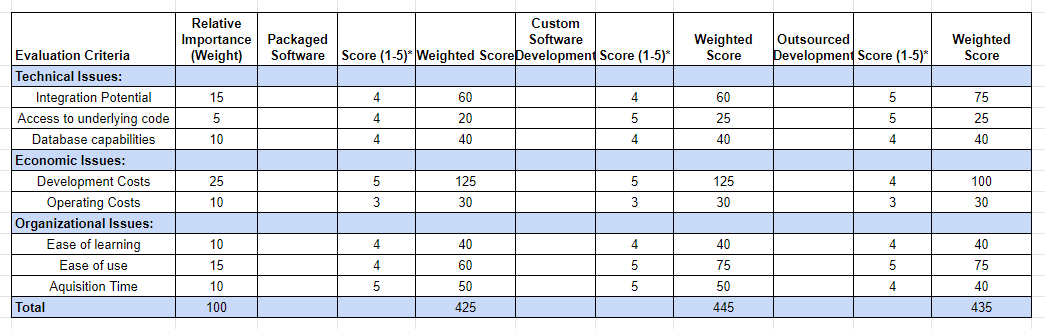
**Individual Alternative Matrices**

Nima:

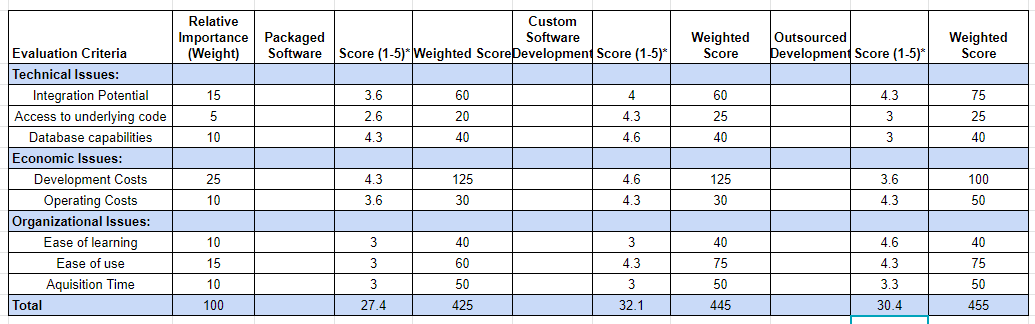


Upen:



Alex:

**Team Matrix**



**Alternative 1: Packaged Software**

By purchasing and implementing prepackaged software it will result in similar performance, and flexibility with lower investment. With prepackaged software, the A.O.K. library will not need to reinvent the wheel with a custom solution. There are thousands of libraries with similar systems which have tried and tested products which could be used at UMBC. This method will resolve significant cost savings, time savings, and more reliable software support. This solution can be customized to an extent to fit UMBC unique issues. However, the solution will not be a perfect fit, compromises will have to be made.

To keep in mind, packaged software most of the time never fit perfectly with most library systems, especially with the A.O.K. system. The organizational processes of the packaged software would have to adapt to the software used on the A.O.K. library systems. Getting the software packaged, the reliance on the vendor to do maintenance and future enhancements will eventually get in the way by depending tremendously on the vendor. This itself would be problematic for scheduling for both the library administration and the vendor causing serious inconvenience for both. Additionally, due to the dependency on the packaged software, there would not be any technical skills nor in-house functional skills for solving any adjustments if needed. The software does not meet all the requirements needed to have an efficient Laptop Loaning system.

**Alternative 2: Custom Software Development:**

Using custom software development allows for a more complete and robust software solution for the A.O.K. library. Custom development will allow the software to be custom tailored and optimized for the unique use cases. The A.O.K. library uses a legacy software called ALEPH for item and patron authentication and management. ALEPH has no accessible api and will need a custom solution to interface with it. This will involve the utiliationzation of a dedicated server with ALEPH and custom software to create an api endpoint for internal use. This option will allow UMBC to build and retain complete understanding of the system and keep technical knowledge in-house. Additionally, by developing a custom software it will allow the team to maintain consistency between the different technologies. However, this route has several drawbacks. The custom solution will take significant time, energy, and resources. It will significantly more expensive, has a higher risk of failure, and will require additional developers.

**Alternative 3: Outsourced Development:**

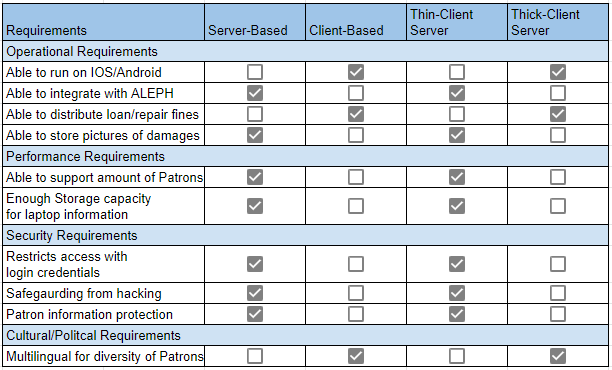
By outsourcing the development of the software we will be able to hire higher specialized expertise that we may not have in house. By having higher expertise will save tremendous amount of time and money for the library for budgetary purposes. With expertise to assist to setting the software to the A.O.K. system the lower they are risks in implementing it. By outsourcing the development, there is a lack of in-house expertise of any sort due to relying on experts to come in to implement the system. By hiring higher expertise, the dependency is significant just as packaged software that adds to numerous issues that the A.O.K. library has before and after either options limiting future options on any further enhancement of the system. By relying on experts to set up the system software, the chances of potential loss of confidential information is very likely to occur. Performance of the software implemented onto the A.O.K system is determined based on the terms and conditions of the contract set from the beginning of the software implementation onto the system.

**Explanation of Team Choice**

After weighing the averages of everyone's choices, we decided to pick alternatives 2 which is custom software development. We picked this because it scored the highest of the three options we had. We believe that custom software development would also be helpful for us for a personalized solution as it acts as a full proof. We also believe that it will help us to resolve integration issues because general software commonly causes integration issues. The UMBC library uses ALEPH for patron authentication which has no accessible API and custom solution would offer as an endpoint for interface solution with ALEPH.

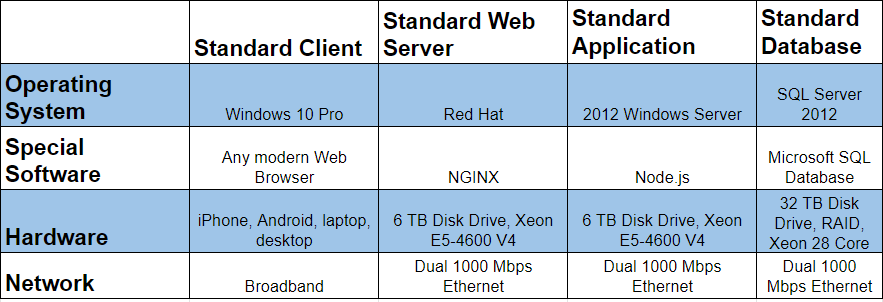
However, the cost of custom development will be a little higher as compared to packaged software and outsourced development. But it will allow UMBC to build and retain a full understanding of the system and it can suit easily to the changing requirements. Additionally, the custom software development will allow the team to maintain consistency between different techniques and in-house development would be safer as the chances of intrusion can be minimized. As custom development offers reliability, compatibility, flexibility, adaptability, uniqueness and considering its size and needs, we prefer custom development over other alternatives.

**Architecture Matrix**

****

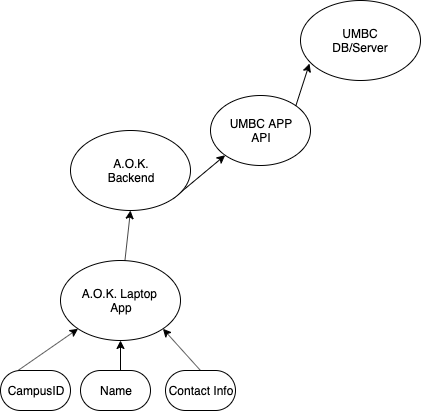
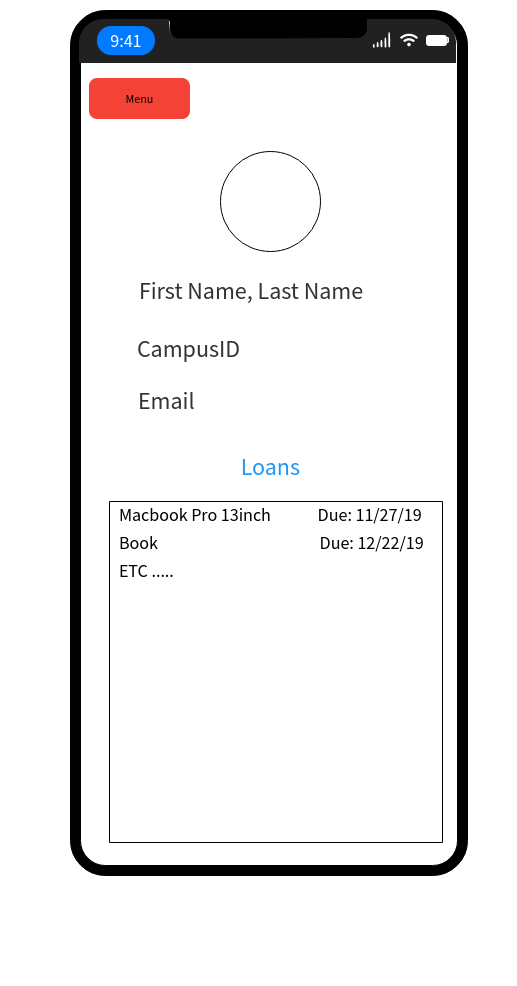
For our system, we decided to use a Server based architecture after reviewing our requirements. We chose server-based architecture because most of our requirements have to do with storing information, integration, or access logic. For example, as seen on our architecture matrix, one of our requirements so to integrate our system with the old system (ALEPH). Another one of our requirements that server-based architecture servers covers is storing pictures of damages. So based on all of the requirements we need to cover, we went with a server-based architecture.

Hardware & Software Specification



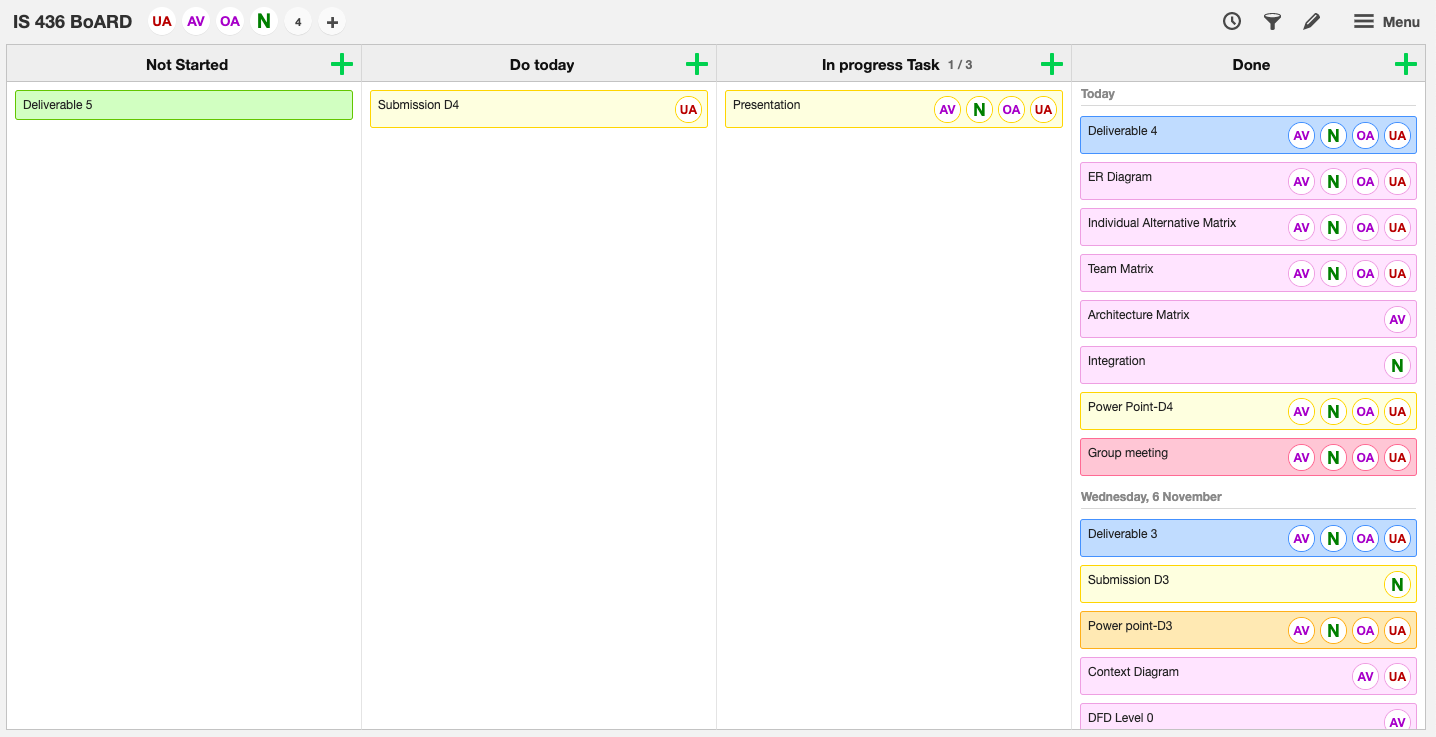
For our system, we used the criteria above for our hardware and software specification. Since ALEPH can only be run and managed on Windows. The software cannot be run on Linux. We be running Microsoft SQL Database as our DBMS. The database server will be running SQL Server 2012 by microsoft and will have 32TB of storage with RAID, and a Xeon 28 core cpu. This system will also have dual 1000 Mbps ethernet connection. The application will be running on Microsoft server 2012 with Node.js. The system will have 6TB of storage, a Xeon E5-4600 V4, and dual 1000Mbps Ethernet. The web server will be running Red Hat Linux with NGINX for the web server. This system will have 6TB of storage with a Xeon E54600V4 and dual 1Gb ethernet connections. The clients accessing this system will be varied and since the application will be web based accessibility will be fairly uniform and simple.

Integration with UMBC App



The UMBC has all the credentials needed for renting out the laptops. The system will use a (SSO) Single Sign On system to allow the student to see their status in the libraries system. It will show how many laptops are available for rent, this will include the different times of availability(4 hours and 3 days). It will show the status of not only the laptops that are rented out but any books that are checked out as well as the date to return them. It will display the picture of the laptop when it was checked out. The application will give reminders of due dates of the laptop and book returns. The integration into the UMBC App mitigates many of the security concerns because the UMBC logon authentication process is incredibly secure. We will not have to rely on a separate authentication process for the library renting process.

Kanban Board: Task Distribution

****

Detailed Project Work Plan

