The A-Team

UMBC Textbook Marketplace

Tuc Nguyen - Amra Mo - Jun Kwon - Hao Wu

Amir Beshir - Sophia Shin

IS 436 - Structured Systems Analysis and Design

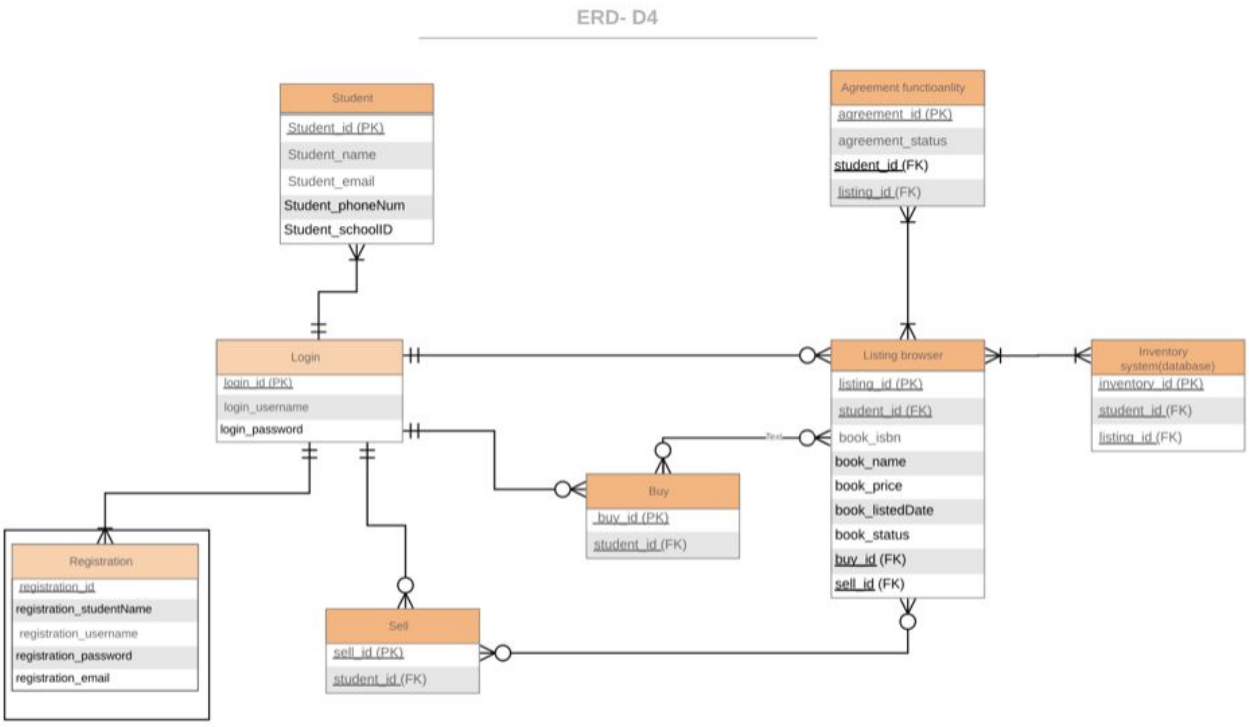
“Data Modeling and Starting Design” (D4)

(D4)

May 9, 2019

Table of Contents

Entity-Relationship diagram:



I. Student

- The student entity has a unique primary key of Student ID. Every student will have a name, email, phone number, and school ID. The student entity will have a one and only one relationship with the Login entity.

II. Login

- The login entity has a unique primary key of Login ID. Every login action has a username and password. Every login entity has one to many relationships with the Student entity. Also, every login has a one to many relationships with registration. A login can have zero to many relationships with the listing browser entity.

III. Registration

- Registration entity has a unique primary key of Registration ID. Every registration action has a student name, a username, a password, and an email. Every registration has a one and only one relationship with the login.

IV. Listing Brower

- The listing browser entity has a unique primary key of listing ID. The listing browser also has foreign key student ID referencing student, foreign key buy ID from the Buy entity, and foreign key sell ID from the selling entity. It also has

variables ISBN, name, price, listed date, status(available, pending, or checked out). The listing browser has a one to only one relationship with the login entity. The listing browser can have zero to many relationships with the buy and the selling entity. Also, the listing browser can have a one to many with the agreement functionality. Finally, the listing browser can have a one to many relationships with the Inventory System (database?

V. Buy and Sell

- The buy and sell entities are both separate but follow the same relationship in the ERD. They both have primary keys to buy ID and sell ID. Both have foreign key student ID from the student table. Finally, both have a zero to many relationships with the listing browser.

VI. Agreement functionality

- The agreement functionality has a unique primary key to agreement ID. Each agreement functionality has an agreement status. Each also has two foreign keys, student ID from student and listing ID from listing browser. The agreement functionality has a one to many with the listing browser.

VII. Inventory System (database)

- Each Inventory System has a unique primary inventory ID. Also, foreign key student ID from the student and foreign key listing ID from Listing Browser. The Inventory System has a one to many relationships with the listing browser.

Alternative Matrix:

Evaluation Criteria	Relative Importance (Weight)	Alternative 1: Amazon Web Services	Score (1-5)*	Weighted Score	Alternative 2: Google Cloud	Score (1-5)*	Weighted Score	Alternative 3: Microsoft Azure	Score (1-5)*	Weighted Score
Technical Issues:										
Read/Write	10	5 TB / 5TB	5	50	5 TB / 5TB	5	50	5 TB / 5TB	4	40
Scalability	15	Easy, pay as you go	3	45	Easy, pay as you go	4	60	Easy, pay as you go	3	75
Storage Capacity	15	11 TB	4	60	11 TB	5	75	11 TB	3	45
Experience With Product	15	None	2	30	None	2	30	None	2	30
Economic Issues:										
Cost per month	30	$969.60	4	120	$588.45	5	150	$1,489.90	2	60
Organizational Issues										
Interface Difficulty	15	Easy to navigate	4	60	Easy to navigate	5	75	Easy to navigate	5	75
TOTAL	100			365			440			325

Alternative 1:

- Amazon web services were considered an good candidate for our project because it uses a secure cloud services platform, offering to compute power, database storage, content delivery, and other functionality to help businesses scale and grow. AWS is easy to navigate around and is user-friendly. The price point was the highest of the three we chose.

Alternative 2:

- Google cloud platform was considered a good candidate for our project because Google cloud is a great option for many because of its ease of use. Cloud Datastore has used NoSQL database for web and mobile applications and Cloud Storage is the object storage with integrated edge caching to store unstructured data.

Alternative 3:

- Microsoft Azure was considered a good candidate for our project because their services offer a range of uses and benefits for organizations big and small. It is especially beneficial for us since they provide the ability to scale on demand and provides simple and reliable data storage.

We have all agreed to go with using Google's cloud platform instead of its alternatives at Amazon and Microsoft, mainly due to the price and flexibility of growing and expanding our services based on the current needs of where we are. Since we are self-funded services we made it a priority to make sure its the cheapest option and since we are planning on further expanding this to other campuses and perhaps create an mobile app.

Architectural Design:

Requirements			
Operational Requirements			
System Integration Requirements	X	X	
Technical Environment Requirements	X	X	
Maintainability Requirements	X	X	
Performance Requirements			
Speed Requirements		X	
Capacity Requirements		X	
Availability/Reliability Requirements	X	X	
Security Requirements			
Access Control Requirements	X	X	
Encryption/Authentication Requirements		X	
Cultural/Political Requirements			
Multilingual Requirements		X	
Legal Requirements	X	X	

Operational Requirements

Technical Environment Requirements:

- Server-based: In order to have an always-on network connection permitting real-time database updates, the server will handle web requests sent by users visiting the website and will manage and process the data.
- Thin client-server: In order to have always-on network connection permitting real-time database updates, thin client will rely on a remote server and data and business logic is handled on the server.

System Integration Requirements:

-Server-based: The server will handle data storage, data access and presentation logic, then the system will be able to read and write to the database.

-Thin client-server: Presentation logic will handle on the thin client and data access logic will handle on the server so that the system will be able to read and write to the database.

Maintainability Requirements:

- Server-based: The server will handle business and data access logic in order to manage and maintain the data.
- Thin client-server: Thin client will rely on a remote server and the server will handle data storage and data access logic to maintain the data.

Performance Requirements:

Speed Requirements:

- Thin client-server: In order to maximize the performance of the application, the server has to handle data storage and data access logic because it handles all the database queries.

Capacity Requirements:

- Thin client-server: To be able to support simultaneous users at all other times, the thin client has to handle presentation logic (the user interface) and the server will be handled data storage and data access logic to distribute user traffic.

Availability/Reliability Requirements:

- Server-based: Since the system will be available for use 24 hours per day and 365 days per year, the server will run all the processes the application requires in order to work properly.
- Thin client-server: Since the system will be available for the user 24 hours per day and 365 days per year, the server will run all the processes the application requires in order to work properly and the thin client will be able to communicate with a server.

Security Requirements:

Access Control:

- Server-based: The server will handle data access logic because of developers should be authorized to access the system code to provide new features and fix bugs.
- Thin client-server: Thin client will handle presentation logic and the server will handle data access logic because student users will have permissions to view and edit their own book listing and account information and system administrators will be allowed permissions to maintain the system.

Encryption/Authentication Requirements:

- Thin client-server: Thin client will handle presentation logic and the server will handle data access logic because data will be encrypted from the user's computer to the web site to provide security.

Cultural/Political Requirements:

Multilingual Requirements:

- Thin client-server: Since the system will be able to distinguish between the US currency and currency from other nations, a thin client will handle presentation logic which is the acceptance of the user's commands.

Legal Requirements:

- Server-based: In order to protect student's personal information, the server will handle data access logic to process required to access data

- Thin client-server: In order to protect student's personal information, the server will handle data access logic to process required to access data and the thin client will make requests to the remote server to get some information.

The Decision of System Architecture

Our decision of system architecture for this project is a two-tiered client/server architecture. We believe that a two-tiered client/server architecture fits on our project because the user interface is stored at the client side and the database is stored on the server side. This architecture can improve the performance of the system and help with scalability.

Hardware and Software Specification

	Standard Client	Standard Server
Operating System	Microsoft Windows	Linux
	Apple MacOS	Microsoft Windows
		Apple MacOS
Software	Chrome	Chrome
	Firefox	Firefox
	Safari	Safari
Hardware	None	None
Network	None	None

Project Work Plan:

ID	Task Name	Days (Duration)	Start	Finish	February		March			April				May	
					Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11
5	Create questionaire	21	March 7	March 15											
6	Conduct interview	6	March 10	March 16											
7	Report observations and findings	7	March 12	March 18											
8	Write Functional Req.	1	March 9	March 10											
9	Write Non-Funciontal Req.	1	March 10	March 11											
10	Finalize Requirements Definition	3	March 20	March 25											
11	D3	15	April 4	April 18											
12	Develop Process Model	8	April 7	April 15											
13	Re-edit Requirements definiton	4	April 15	April 18											
14	Re-edit Use Case Documentation	4	April 15	April 18											
15	D4														
16	Develop ERD diagram	2	April 18	April 20											
17	Develop Alternative Matrix	5	April 21	April 27											
18	Provide Narrative	3	April 25	April 28											
19	Develop architecture into a matrix	5	April 29	May 4											
20	Make decision of the system	5	May 2	May 7											
21	D5														
22	Develop interface prototype (3)	1	April 18	April 19											
23	Discuss interface standards	1	April 19	April 19											
24	Program Design	15	April 24	May 5											
25	System Implementation	1	May 5	May 5											
26	Discuss team SDLC process	1	May 7	May 7											

Kanban Board: