SUPER HEROES IN TRAINING

My Academic Pyramid

Design Document

CECS 491A Sec 05

December 13, 2018

Team Leader: Krystal Leon, 013986607

Arturo Peña Contreras, 010914811

Luis Julian, 007472593

Hyunwoo Kim, 014392909

Victor Kim, 012016990

Trong Nguyen, 016208983

Revision History

Date	Version	Description
11/1/18	1.0	First draft.
12/5/18	1.1	Post Sprint 3 Revision.

Table of Contents

1. Introduction

This document is made for My Academic Pyramid web application implementation in the future.

1.1 Purpose of the document

This document informs the reader about My Academic Pyramid web application architecture. It illustrates the decisions that were made to maximize the performance of the web application with high level architecture diagrams. It's designed for the developer or a person who has equivalent knowledge.

1.3 Relationship to Other Documents

This document follows business rules and functional/non- functional requirements in the Business Requirements Documents (BRD).

2. Application Overview

My Academic Pyramid is a social media web application where students can communicate with their peers and ask questions on a discussion board. The application will also provide an online tutoring service, allowing students to converse with tutors and seek assistance online without the need to travel to campus. My Academic Pyramid is a single page application (SPA) which is a web application that has only one html page and dynamically changes the contents in the single page by user interaction.

2.1 Constraints

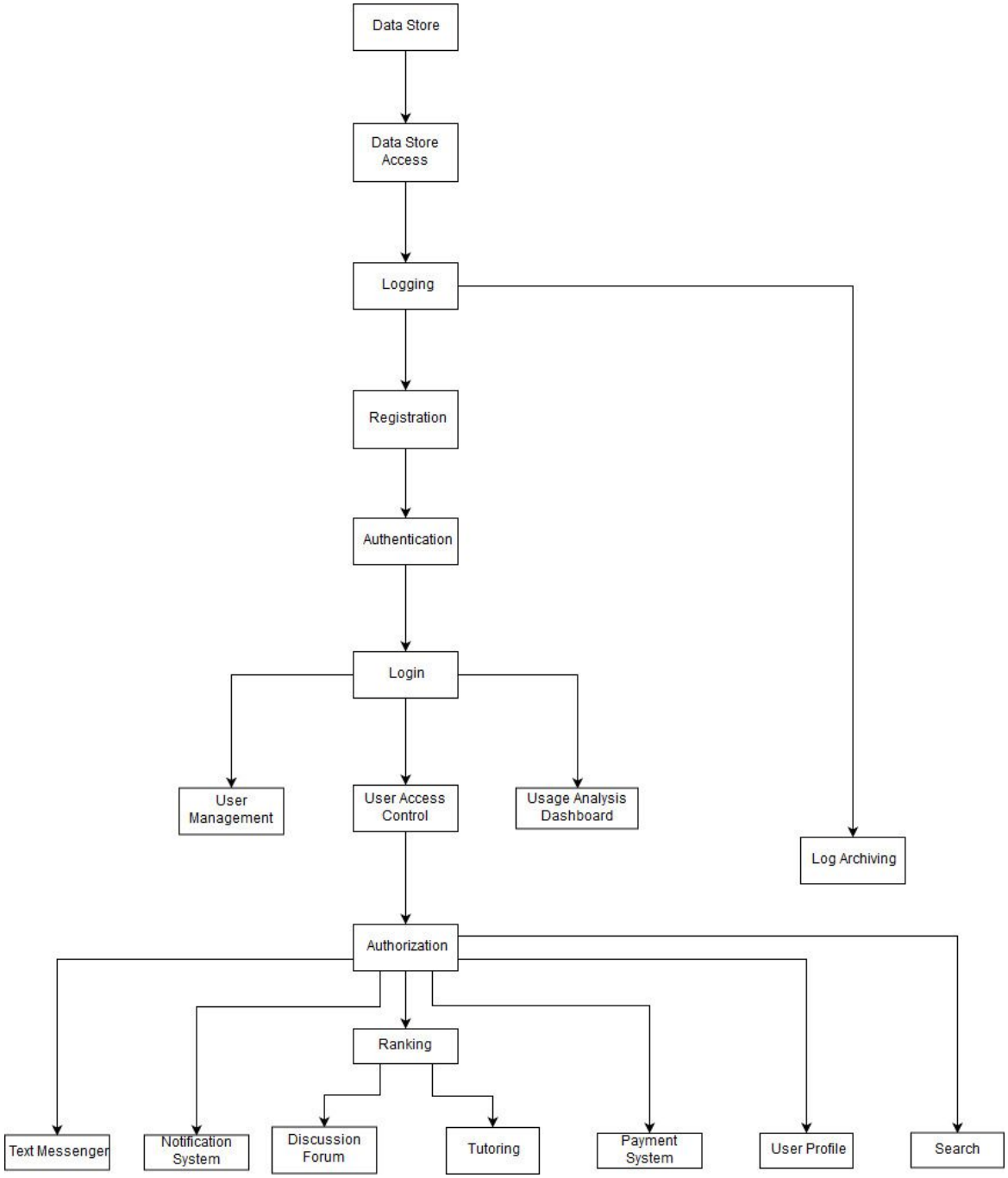
The budget is limited to $0. Therefore, the certain parts of the architecture were restricted during making decisions. If there is not a product or technology available as a completely free product, the architecture will be modified to follow the constraints on the budget.

3. Dependency Diagram

This section illustrates the dependency diagram of the web application.

3.1 Introduction

The dependency diagram illustrates how each different features of our application relies on each other. For example, feature A relies on feature B. When feature B is not available, feature A will not work as specified in the business rules. According to the dependency diagram, all of our web application features rely on Login and Registration. So that along with data access and data store will be among the first features we implement.


Data Store
Data Store Access
Logging
Registration
Authentication
Login
User Management
User Access Control
Usage Analysis Dashboard
Log Archiving
Authorization
Ranking
Text Messenger
Notification System
Discussion Forum
Tutoring
Payment System
User Profile
Search

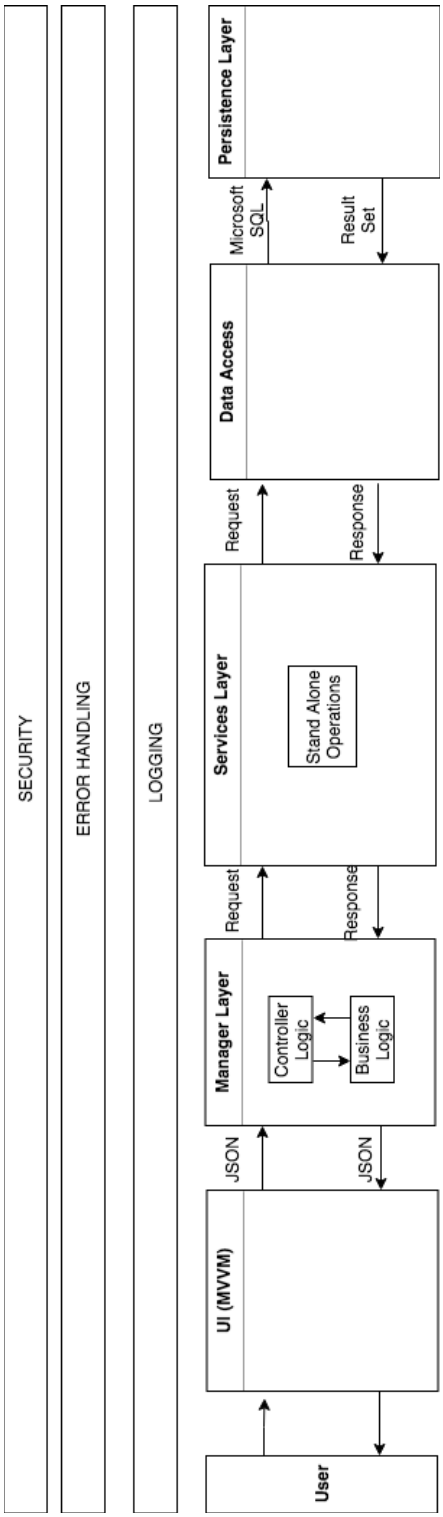
4. High Level Flow Diagram

This section explains the flow of the data with high level architecture.

4.1 Introduction

In order to see the flow of the data, we have divided our web application by different layers and each of them handle the data flows in our application. Our web application consists of six different layers which are UI, Router, Data Handler, Service, Data Access, and database. Each of these layers takes a request and returns a various of response depends on the given request. A layer communicates directly with its surrounded layers only to lower the complexity level of our web application. The behavior of each layer will be explained in details. In addition to those layers, there are layers that affect multiple layers directly. These are security, error handling, and logging.

4.2 Diagram


SECURITY
ERROR HANDLING
LOGGING
User
UI (MVVM)
JSON
JSON
Manager Layer
Controller Logic
Business Logic
Request
Response
Services Layer
Stand Alone Operations
Request
Response
Data Access
Microsoft SQL
Result Set
Persistence Layer

4.3 Behavior of the Layer

4.3.1 User Interface Layer (UI)

The front-end of our web application that displays the contents to the user. This layer directly interacts with the user and accesses the server to fulfill the valid user requests. It forms a request and sends it to the server and renders any response that is received from the server.

4.3.2 Manager Layer

The first layer that interacts with the user's request. It takes a request from the UI layer and determines whether it's a valid request. It makes sure the request meets the business requirements and may modify it such that it can be sent to the service layer. It takes the response from that layer and returns JSON representation of it to the client.

4.3.3 Service Layer

This consists of stand alone services that meet the request of the user. It takes input from the manager layer and processes it some way before returning results. It may user the lower levels to store and access data if necessary. Stand alone services should be as general as possible to make them usable in other applications.

4.3.4 Data Access Layer

Accepts any read and write commands and sends them to the database to carry out.

4.3.5 Persistence Layer

The persistence layer that stores information of the application. It stores and modifies data. It also carries out any query that it is given from the data access layer.

5. Security

Considering security in high level design is critical to protect the confidentiality of our users, we follow a defense-in-depth strategy. We utilize security at many places of different layers, a lot of it overlapping, to protect data even when one of the layers fails. As such we will be able to meet the client's requirements on security.

Different forms of security occurs in all the layers of our application. Authentication will be invoked in the manager layer to validate the session. In each layer afterwards and the UI, authorization will be done to make sure the user has access to a feature and the session is still valid. The data will be validated in the UI, the manager, the data access layer, and database to prevent any code injection that may make our program do any unauthorized actions. The sensitive information in database will be encrypted to mitigate damage in the unlikely event that data is leaked. Similarly to other layers, authorization of the user making the request will be carried out before database access is allowed.

As with all applications, invoking security will always lower the performance of the applications. Many authorization checks require multiple calls to the database which will lower the speed with which requests are handled. Data validation at multiple layers will also require using a greater share of CPU cycles on instructions not related to handling requests and slows the

rate with which requests are handled. After considering these issues and discussing it with the client we decided to accept the performance loss to have a greater degree of security.

6. Error Handling Rules

Database:

- When a search to a database comes up empty, the data access layer will return an empty array if multiple records can be returned. If the search is expected to return only one record, then it will return null when the search finds nothing. The logic in the business layer will take that result and decide whether to raise an exception.

Business Logic:

- When the user tries to access a feature they do not have access to, the server will send a response with an error status code of 403. The UI will use that to build and display an error message.
- When the user attempts to connect to the server when it is offline, the server will respond with 503 status code.
- When the program running on the server makes an error, it will return a response with an appropriate 5XX status code.

UI:

- When the user tries to input invalid data, the application will raise an exception as a result of invalid behavior.

UI & Business Logic:

- When the user tries to carry out an action that goes against application specific business rules, an exception will be raised. Custom exceptions, appropriate to a situation, will be raised to produce a descriptive error message to user and administrators.

7. Logging Rules

There are two solutions to store logs : Storing in a flat file and storing in a database. A text file is simple and unstructured, so we have to use a tool to find what we want to retrieve. Log data can be split into multiple files and those file may be located in different locations. Also, it takes using a same search query in every file lowers the performance of our web application.

Therefore, we have decided to use a database to store logs, because it quickly retrieves data and eliminates issues with a text file. We store log messages under a structure that allows us to write rich query to retrieve log data quickly. We can combine multiple search conditions and attributes to have a deep insight about the events. We may use a NoSQL database which is designed to store documents and allow rich querying as well as indexing. NoSQL database is less schema, so it allows any data being stored in the database without a prior key. A relational database requires many tables to represents an entity. In contrast, NoSQL database can represent an entity by using a single rich data structure. For archiving, we can archive old log data in non-frequent storages such as in HDDs.

In order to have a structured logging system, we have decided to create a logging class. Other classes will inherit the logging class to execute inherited functions to store log data. These functions will act as queries to the NoSQL database to store the data.

8. Network Architecture Diagram

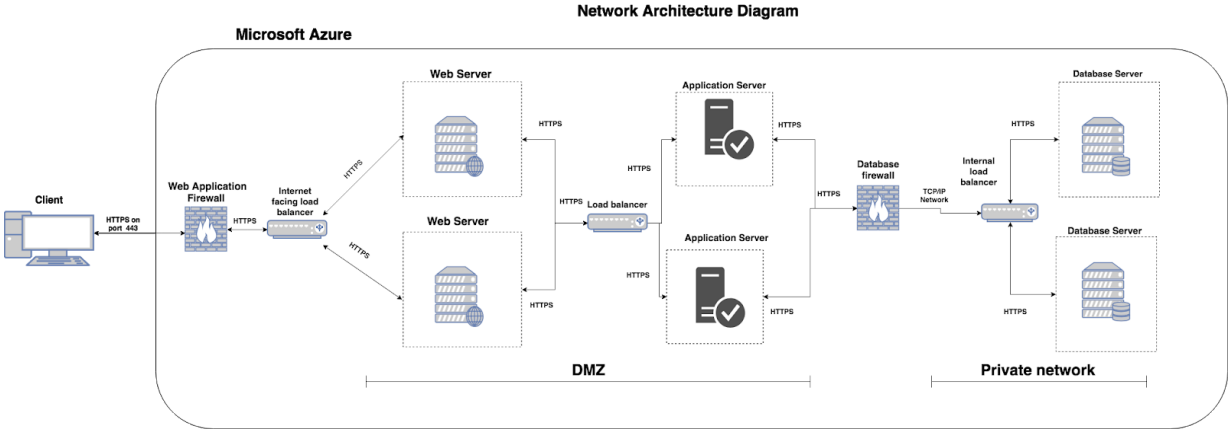
8.1 Introduction

In this network architecture design, we provide a high level network hardware architecture diagram that we have designed to maximize the performance of our web application. It explains an overall process of traffic flows in the networking system. This diagram consists of the following network components : Firewall, load balancer, web server, application server, and database.

8.2 Design

8.2.1 An ideal design

This design maintains the availability of our system and allows clients to connect to our application without downtime. An ideal network architecture consists of two web servers, two application servers, two SQL database, two firewalls, and two load balancer.



Client: User uses the web browser to access our internet application. All requests and responses will use HTTPS protocol on secured port 443.

Web application firewall: This front-end firewall allows external traffic to go into the "Demilitarized Zone" only. This network zone is isolated from our private network containing the databases and only allow HTTP or HTTPS.

Database firewall: This firewall provides an additional layer of security. If the external traffic wants to pass the DMZ to reach the internal back-end server, it must go through the second firewall. The database firewall has a different set of firewall rules which can help limit the access to the private network

Internet facing load balancer: Helps route traffic from the client to our two web servers. The load balancer keeps track of the status of the servers. In case one web server goes down or has bottleneck issue, the load balancer can redirect the traffic to a better web server.

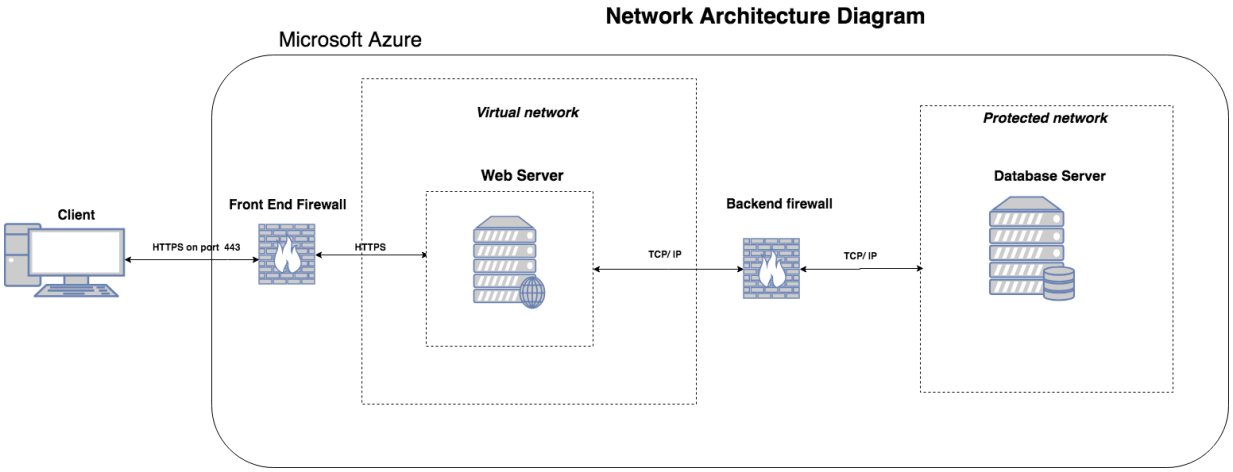
Internal load balancer: Acts like an internet facing load balancer, but instead routing external traffic, it routes internal traffic which has already passed the first load balancer. The traffic will be distributed into two databases.

Web servers: Satisfy the requests from the client by accept HTTP or HTTPS. The web server gathers resources and delivers the web page to client side. The web servers are exposed to a public network for being accessed by clients.

Databases: Our collection of data for the web application.

8.2.2 The realistic design

It is very costly if we follow the ideal design. In reality, due to budget constraint, we only get one web server, one application server, and one database. Therefore, we have to remove additional web servers, application servers and databases from the diagram. However, Azure Student plan offers a feature call Virtual Network. This feature allows us to isolate network layers in order to improve the security, so we are going to factor it into our design.



To enhance security, we put web servers in an isolated network (virtual network). By doing that, we can protect our back-end resources, application server and database server from attackers in unsecured network. The virtual network creates boundaries to allow only traffic which is defined in the user's configuration. Our database is located in a protected network and its access is handled by the second firewall. This approach provides an extra layer of security in our web application.