HomeView

High Level Design

Uniting streaming services on one site

October 6, 2021

Unite
Christian Lam
Daniel Monge
Eric Truong (Team Leader)
Erina Lara
Michael Lamera

Table of Contents

1. Introduction

1.1 Overview

This document describes the architecture used in the development of the HomeView web application. The document will also be referring to some of the features listed in the Business Requirements Document (BRD).

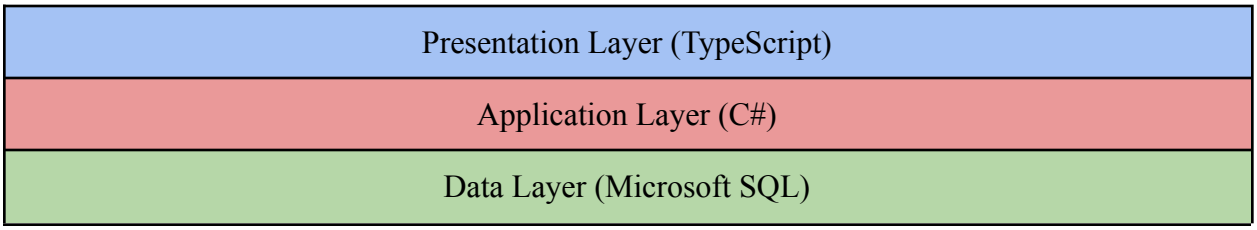
1.2 Application Overview

HomeView is a web application that makes it easier to browse U.S. streaming services. To make it a personalized experience, HomeView will also provide features such as customizable playlists, blacklisting ability, reviews, information on actors, and a news section. HomeView will be a single-page application where each page will house a certain feature.

1.3 Constraints

The budget for this web application is limited to $0 which means that all software used to create this product will have to be free.

2. Architecture



2.1 Presentation Layer

This layer houses our user interface (UI) which allows the user to communicate with the software. The user would be able to send data such as, search queries, or receive data like movies or TV shows. Our interface is written with TypeScript and shown through the supported web browser. This layer is commonly known as the front-end.

2.2 Application Layer

The application layer is where all the business logic is used to process any kind of data collected from the presentation layer. This layer is also in charge of coordinating the data flow to and from the presentation and data layer.

2.3 Data Layer

The data layer is where all of our databases required for the necessary features will be stored. All data that has been processed by the application layer will be sent here, and likewise, data needed to be displayed will be retrieved from this layer. The databases will be managed using Microsoft SQL Server 2019 Developer Edition. This layer along with the application layer is commonly known as the back-end.

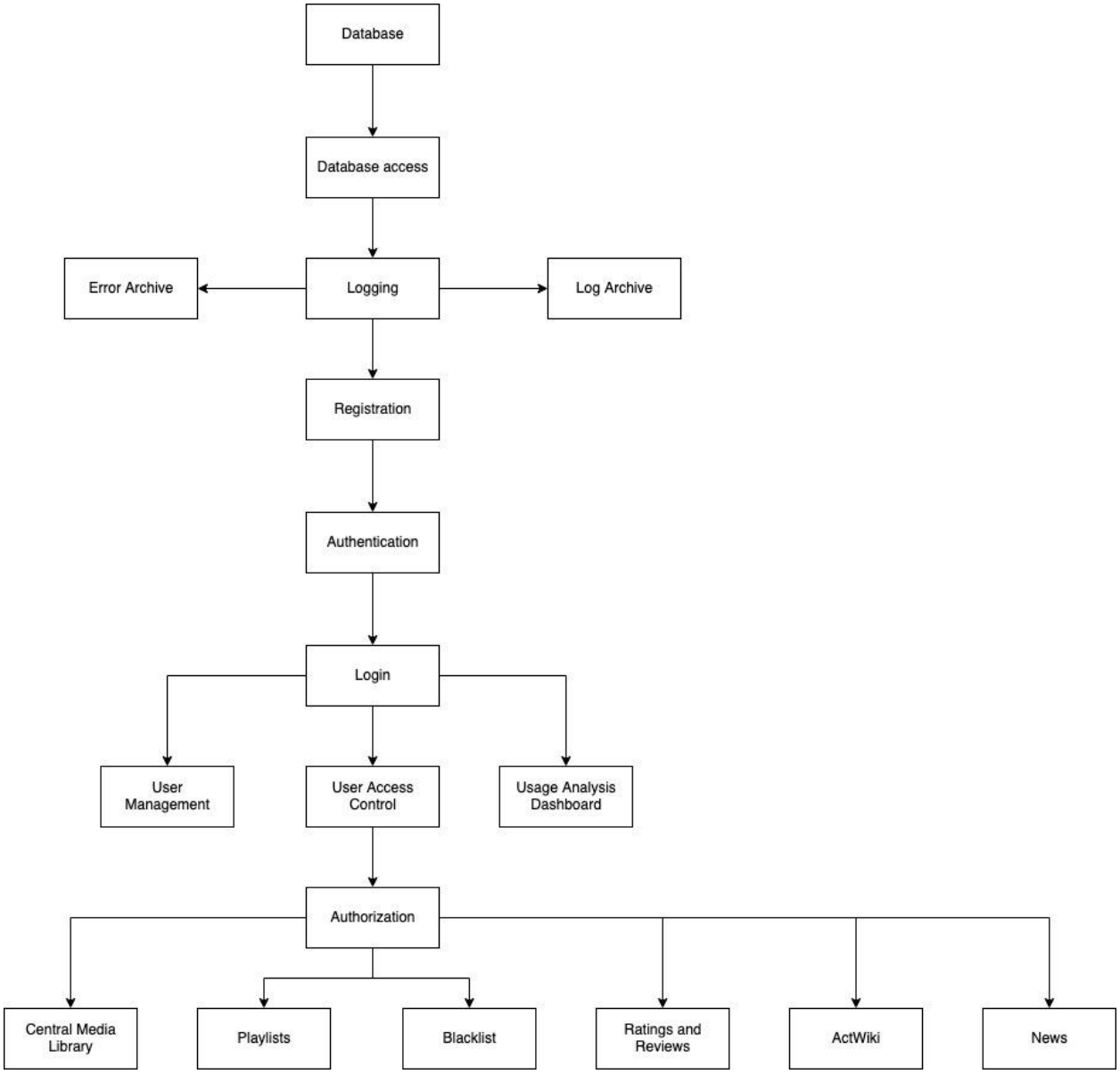
2.4 Benefits of 3 tier architecture

The main benefit of the 3 tier architecture was that all layers were logically and physically separated. By having the tiers separated, each tier can be upgraded individually without affecting any of the other tiers. One of the benefits that comes with this architecture is faster development. Since all tiers are physically separated, each tier can be worked on by different teams at the same time, effectively reducing the amount of time needed to work on the product. Another benefit is improved scalability, which means that each tier can grow in regards to performance without

affecting any of the other tiers. The tiered architecture also provides improved reliability to the tiers, preventing any impact in the tiers if one of them has an outage. Lastly, the tiered architecture has improved security because the presentation and the data layers do not communicate with each other. The business layer acts as the middleman for the two layers and can function as an internal firewall, preventing malicious exploits if designed correctly.

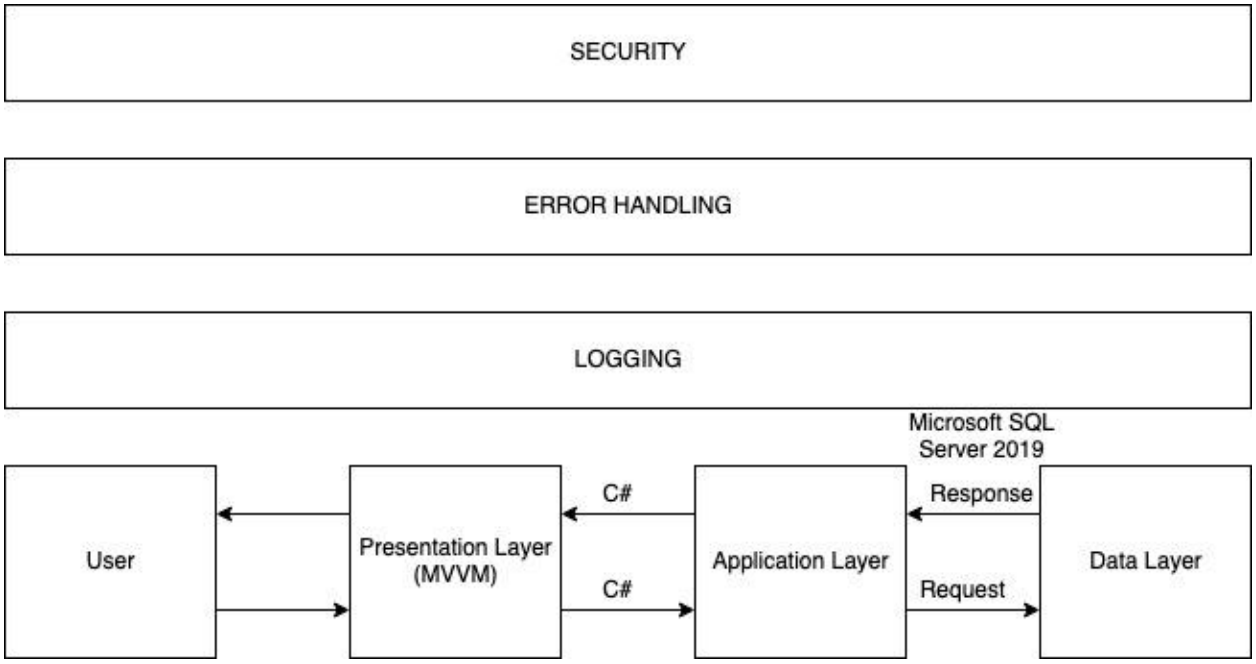
3. Dependency Diagram

The dependency diagram highlights which feature relies on which in our application. Since most of our data will be stored in databases, they will be the first feature to be implemented as all other features will need to utilize them.



4. High Level Flow Diagram

The high level flow diagram shows how data flows inside our application, and which language is used to communicate between the layers. The layers will take in input or a request and will return the resulting data. The diagram also shows that some layers, like security, error handling, and logging; will be affecting multiple layers.



5. Security

To help secure our application, we will be utilizing encryption on user's passwords, applying NIST standards, and preparing against OWASP's top 10 security threats. In order to access our application, the user must first be authorized. The user must register an account with us by supplying their email address and creating a strong password for their account. They will then have to verify their email address, and they will have access to our application. Our application will also authenticate users by having them log in with their username and password when they arrive on our website, and if they chose not to remember their credentials.

6. Error Handling Rules

All errors possible in the application must have a message whenever it happens. For example, whenever the user tries to log in or tries to access data that is not in our system, the user will be shown a pop message detailing that an error has occurred in our system. In addition, the site must be responsive at all times and not crash when a user is active.

7. Logging Rules

We will be keeping logs of everything related to user input such as log in attempts, successful logins, title clicks, and search queries. These logs will help debug if any application errors occur, and it will help analyze user's preferences. All logs can be downloaded and viewed through the User Analysis Dashboard.

8. Network Architecture Diagram

The network architecture diagram shows the hardware that we will be using to host our web application. The web host will be responsible for allowing our product to be accessible using the internet by uploading our files into the cloud. With that, the user is then able to access our website on the internet, and be able to input data. Our web server which stores our business logic, will be able to take the user's input and process it to be sent to the SQL server.