Layer Definitions under Security, Error Handling, Logging

	Security	**Error Handling**	**Logging**
UI Layer	AuthN and AuthZ will be issued to all normal users, admins and root admins. AuthN will be determined at user login and logout. AuthZ will be determined as user interacts with views	Errors that the user should know will be displayed via UI to the user. (Error: media was not uploaded) Errors that are unrecoverable or are irrelevant to the user will not be displayed to the user. (Error: Media upload was not completed in 3 seconds)	All errors in this layer will be logged (Errors that should be displayed, errors that are not displayed to user) Any interaction that user makes with UI will be logged Any backend errors that come to this layer from web framework layer will be logged
Web Framework Layer	ASP.NET Core has built in security features that are free to use. Documentation is provided on how to avoid most common and basic attacks (Sql Injection Attacks)	Set up exception handlers to handle errors. This also lets us access exceptions to accurately assess errors. Provide status code pages for errors as ASP.NET Core does not provide a status code page for HTTP error status codes, (ex. Status Code: 404; Not Found)	All errors in this layer will be logged (Exception handlers, Status code pages) Logging HTTP and HTTPS requests, especially those requests dealing with data requests, user input requests, and file requests.
Business Layer	Defining our user types as stated in our BRD. (Anonymous User, Admin User, Root Admin, Normal User) Defining what permissions each user type should have. (App permissions matrix in BRD)	All errors for all core components and feature behaviors are stated in BRD Handling errors are stated in BRD. Comprises of restricting view or functionalities, Displaying messages, and other error handling techniques stated in BRD	One main User story success logging Multiple different User story failure logging Universal components success and failure logging

Data Access	All DAOs will have granular Create, Read, Update, Delete operations. Will only Create, Read, Update, Delete data from tables that the DAO only has access to.	Implement Try Catch blocks in our DAO class to mainly catch data access errors Implement good coding practices in DAO classes. (using(a = “”){// code})	All Data access Errors will be logged Any Data Access with CRUD in the DAO will be logged Any Data from DTOs that come into this layer will be logged
Data Store	Database user with certain permissions will be able to read and write to database UserReadOnly, UserWriteOnly users will be created to read and write to the database.	SQL try catch blocks will be implemented Checking whether database I/O has error reading and writing blocks of records	Database I/O error and try catch blocks will be logged. Database I/O reading and writing to blocks of records will be logged

Tech Stack

	Pros	**Cons**	**Are we ok with Cons?**
React	Using react would allow us to manage our client side and server side applications independently and efficiently in separate teams. The component based architecture allows for better development as well as the wide variety of third party tools	Library Growth and React not being a full-feature framework make it less feasible as react does not grow their libraries at a steady pace hence forcing the user to relate to third party tools, when compared to Angular, React has less features such as tools and libraries needed to code the Model aspect in a MVVM	Due to the high popularity and easy to learn structure of React we as a team are ok with the cons of using React.

ASP.NET Core	The main pros include cross-platform support, high performance, cloud-native architecture, open-source nature, and ease of use as well as a unified story for building web UI and web APIs	The two major cons consist of limited to no support for older libraries and limited to no support for older versions.	We will be using ASP .NET Core as we believe that the Pros heavily outweigh the Cons
Places API	According to Google their API provides the most accurate, up-to-date, and comprehensive place models of the real world	Not much data has been collected for use in many Rural and Frontier areas as well as the API needing to be constantly updated.	We are ok with the Cons as our scope only includes the well explored Los Angeles area and consistently updating the API should be an automatic occurrence.
Maps Static API	According to Google their API provides the most accurate up-to-date data, easily implementable and ease of use for the user	The major con is that we will have to pay Google to use their API and will be provided with an API Key upon successfully paying them.	Currently due to having a very small introductory customer base we will have a lesser payment to be made to Google hence we are ok with the stated Con.
Quotes 500K Database	Web scraped quotes from popular quote sites. Has multiple categories for one quote.	Very large data set is collected with 500k quotes so performance issues may occur. Only limited quotes of the four most popular quote sites (goodreads, brainyquotes, famousquoteandauthors, curatedquotes)	The only feature that is using this database is motivational quotes. One of the cons is irrelevant because we can search up certain categories for what types of quotes we want. Other con is irrelevant to us since it doesn't matter where the quotes are from

List of Microservices

Microservice	User stories involved with microservice
Security	A1, A2, L1, L2 user stories, Authentication universal component
Logging	Logging universal component
Archiving	Archiving universal component
Error Handling	All user stories in BRD
User Management	R1, R2, AR1, AD1, UAD1, UPC1, UM1, UM2 user stories
LLI	LLI1, LLI2, LLI3, LLI4, LLI5, LLI6