HIGH LEVEL DESIGN DOCUMENT

TEAM NAME: PIXEL PALS

TEAM MEMBERS: BRANDON GALICH
CARINE GORDILLO
SARAH SANTOS
JASON LAM
SARAH PHAN

TEAM LEADER: KAY KAYALE

DATE: NOVEMBER 22, 2023

GITHUB: GITHUB.COM/CARINEGORDILLO/CECS491

Version History

Version	Date	Changes
1	November 13, 2023	Initial submission
2	November 22, 2023	Changes to Diagram - Added Default Cross Cutting Behaviors - Added Client-Side. Server-Side - Added Front-End, Back-End - Added Microservices - Added Tech Stack - Added Specific Request and Response Added Technology Section - Tech Stack - Limitations - Advantages Added Default Behaviors of Cross Cutting Sections in Document

Version History..... 2
Introduction.....5
Purpose.....5
Relationship to Other Documents..... 5
Application Overview.....5
Constraints.....5
High Level Architecture Diagram..... 5
Introduction.....5
Diagram..... 6
Behavior of the Layer.....7
Front-End..... 7
User Interface Layer..... 7
Back-End..... 7
Manager Layer.....7
Service Layer.....7
Data Access Layer..... 7
Data Store Layer..... 7
Security..... 8
Default Security..... 8
Front-End.....8
User Interface Layer..... 8
Back-End.....8
Manager, Service, Data Access Layers..... 8
Data Store Layer..... 9
Error Handling Rules..... 9
Front-End.....9
User Interface Layer..... 9
Back-End.....9
Manager Layer.....9
Service Layer.....9
Data Access Layer..... 10
Data Store Layer..... 10
Logging Rules.....10
Default Logs..... 11
Front-End..... 11
User Interface Layer (View Related Information)..... 11

Back-End..... 11
Manager Layer (Business Related Information).....11
Service Layer (Server Related Information)..... 11
Data Access Layer (Data Related Information)..... 11
Data Store Layer (Data Store Related Information)..... 11
Technology.....12
Technology Stack..... 12
Limitation/Advantages.....12
Front-End:.....12
Back-End:.....12
Tool Chain:.....13
Services:..... 14

Introduction

Purpose

The purpose of this high-level design document is to outline the architectural and design aspects of Space Surfer. It defines the overall structure, interaction of components, and key design considerations.

Relationship to Other Documents

The high-level design document works together with the Business Requirement Document (BRD). The BRD defines the business needs and what the users would expect from Space Surfer. The high-level design document serves as a detailed plan to provide insight into a functioning system. It follows the guidelines and constraints provided by the BRD, so the architecture stays within the rules and boundaries. The high-level design acts as a bridge to help us turn the concepts into a functioning system.

Application Overview

SpaceSurfer is a tool used for enhancing productivity for users ranging from employees at a company to everyday individuals. For general users, SpaceSurfer provides seamless access to a wide range of spaces, offering a convenient way to plan and utilize public amenities. For employees, the application offers specialized features that enable them to make reservations within their office environment, ensuring a productive and organized workspace, both in person and virtually. The application will be a Single-Page Application (SPA) that consists of just one HTML page that will update the view as the user navigates through the application.

Constraints

Space Surfer will be operating within a constraint that will shape its architectural and development choices. The application will run from free to minimal expenses of AWS and other services. Considering this constraint, open-source and low-cost solutions for development and operations need to be taken into consideration.

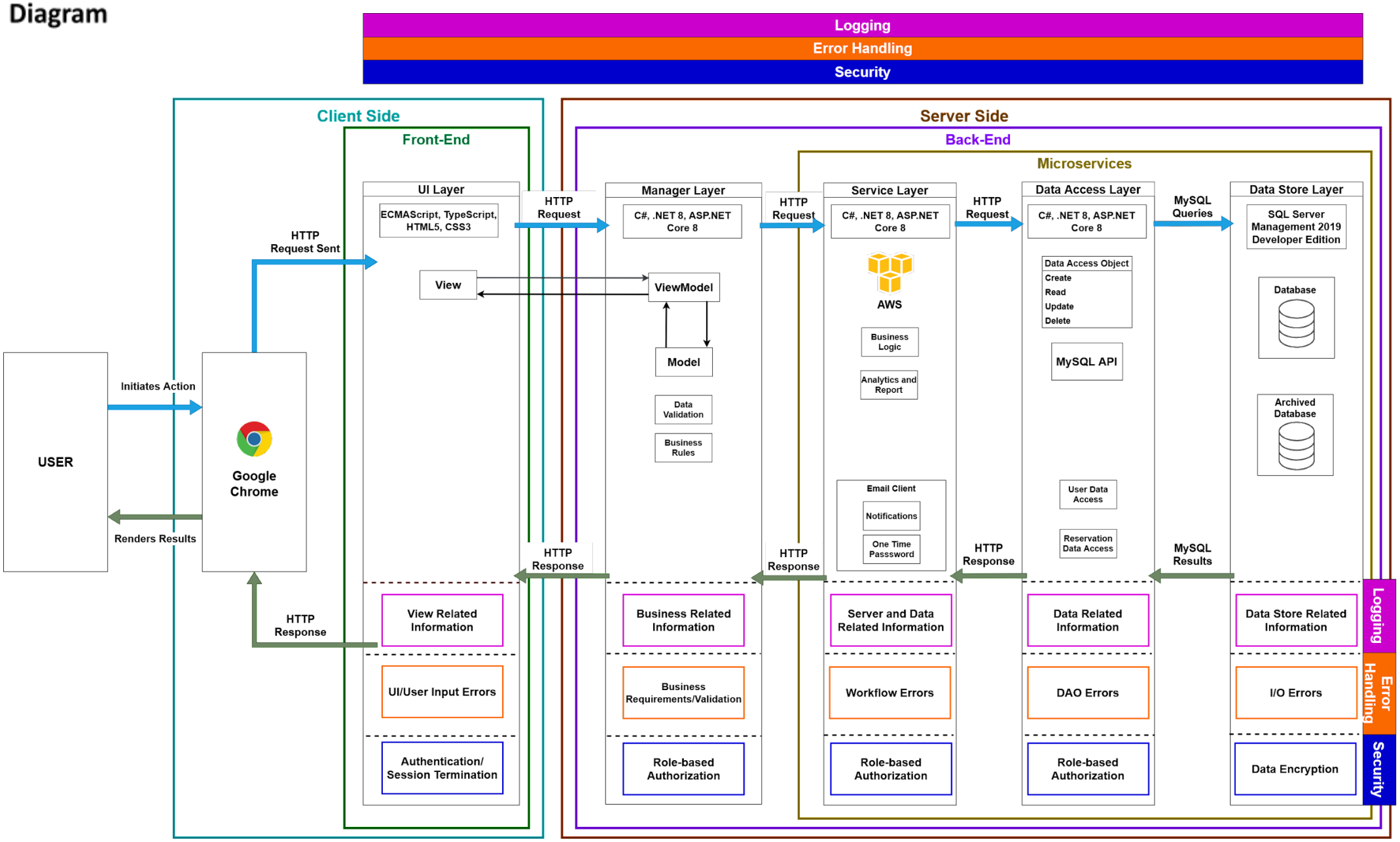
High Level Architecture Diagram

The High-Level Architecture Diagram illustrates the flow of information through Space Surfer's layers. Arrows represent the direction of information flow, showcasing the interconnected nature of the layers.

Introduction

We have divided our application into multiple different layers to show the flow of information from the front end to the back end. The layers we have listed in the flow of our application are the following: UI, Manager, Service, Data Access, and Data Store. Each of the listed layers has a behavior they are responsible for which is connected to other layers. The detailed description of each behavior is described in the “Behavior of the Layer” section.

Diagram


Logging
Error Handling
Security
Client Side
Front-End
Server Side
Back-End
Microservices
UI Layer
ECMAScript, TypeScript, HTML5, CSS3
Manager Layer
C#, .NET 8, ASP.NET Core 8
Service Layer
C#, .NET 8, ASP.NET Core 8
Data Access Layer
C#, .NET 8, ASP.NET Core 8
Data Store Layer
SQL Server Management 2019 Developer Edition
HTTP Request
HTTP Request
HTTP Request
MySQL Queries
HTTP Request Sent
View
ViewModel
Model
Data Validation
Business Rules
AWS
Business Logic
Analytics and Report
Email Client
Notifications
One Time Passsword
Data Access Object
Create
Read
Update
Delete
MySQL API
User Data Access
Reservation Data Access
Database
Archived Database
USER
Initiates Action
Renders Results
Google Chrome
HTTP Response
HTTP Response
HTTP Response
MySQL Results
HTTP Response
View Related Information
Business Related Information
Server and Data Related Information
Data Related Information
Data Store Related Information
UI/User Input Errors
Business Requirements/Validation
Workflow Errors
DAO Errors
I/O Errors
Authentication/ Session Termination
Role-based Authorization
Role-based Authorization
Role-based Authorization
Data Encryption
Logging
Error Handling
Security

Behavior of the Layer

Front-End

The front-end compromises the user interface(UI) layer. The user will interact with the application through Google Chrome browser which would send an HTTP Request to the Front-End. It is responsible for rendering the UI elements and initiating the asynchronous request to the Back-End.

User Interface Layer

The UI layer consists of the front-end of the application that displays a view to the user. The responsibility of the layer is to receive the user interactions and events from the browser and initiate HTTP requests to the Manager Layer. It utilizes ECMAScript, TypeScript, HTML5 and CSS 3. Initiating the HTTP request to the Manager layer based on the user's interactions. Additionally, it handles input validation, UI rendering, User Authentication, and session termination.

Back-End

The back-end acts as the intermediary between the UI and the data infrastructure, processing HTTP requests from the front-end. It implements cross-cutting concerns such as Logging, Error Handling, and Security to maintain system operation and security.

Manager Layer

The manager layer receives the HTTP request from the UI layer. It validates the user authentication and authorizes the user's actions. It is responsible for the business rules, user authorization and the communication between the UI layer and the Services Layer, utilizing C#, .NET 8, and ASP.NET Core 8.

Service Layer

Layer that consists of independent operations that are used in response to the user's HTTP request from the Manager layer. The operations are generalized for reuse capabilities and are responsible for operations that are stand-alone operations such as requesting passwords, utilizing C#, .NET 8, and ASP.NET Core 8. As the HTTP request is passed into the database in the Data Access and Data Store layers, the Service layer will log errors that occur within the database.

Data Access Layer

Layer that allows the user's HTTP request to reach and access the database. Logins and user roles will determine the data that are accessible from the database through authorization. It will be utilizing MySQL to send query commands to the database for creating, reading, updating, and deleting data.

Data Store Layer

Layer that stores any information that is obtained through the database and the application of both front-end and back-end. As the user's HTTP request is sent to the

database from the Data Access layer, the Data Store layer will respond back to the Data Access layer in a HTTP response utilizing SQL Server Management 2019 Developer Edition. It will perform the Create, Read, Update, and Delete(CRUD) operations in response to the HTTP request.

Security

In the Space Surfer high-level design, we take a detailed look at the security measures implemented to safeguard user data and the system. Our approach to security is like setting up multiple layers of protection. These layers are designed to work together to create a strong security system to protect data.

Starting with the User Interface (UI), we implement user input data validation to safeguard against potential issues stemming from unverified user inputs. The Manager layer introduces Front-End Business Rules, which provide an initial layer of security for the Back-End. This layer improves security between the Front-End and Back-End by protecting the system from potential unauthorized inputs and interactions, thereby enhancing security between the Front-End and Back-End. The Service layer allows reuse capabilities, allowing the layer to provide an extra scene of the layers checking for errors as well as the security of the flow of information within the application by logging errors and interactions from other layers. The Data Access layer is critical for protecting data integrity and confidentiality. It helps prevent unauthorized access to sensitive or improper information and reduces the risk of data leaks or unauthorized data transfers. The Data Store layer is the storage for all data obtained through the application. Using data encryption, the data store will have an extra layer of protection as it sends the data back through the layers to prevent leakage.

Default Security

Front-End

In the front-end portion of the application, security will consist mainly of authentication and the termination of sessions or logout.

User Interface Layer

Security will run through checks of authentication as well as the logout of the user to make sure the process is completed successfully. Prevents any leakage of data from occurring as the user logins and logouts, while also running checks on whether the user is who they say they are.

Back-End

In the back-end portion of the application, security will run through checks of authorization of the roles given to the user in multiple layers. Security will also run through checks of the database and the information that is being retrieved and accessed giving it protection from attacks and leaks.

Manager, Service, Data Access Layers

Security will run through checks on the authorization and roles given to the user. Checks to see if the user is authorized to access the information that is being requested in the HTTP request.

Data Store Layer

Security will use data encryption as the data is being sent for an extra layer of protection. Allows monitoring of outside sources to be more difficult as the information travels from the back-end to the front-end by having to go through a decryption process.

Error Handling Rules

Front-End

In the front-end portion of the application, the system will handle errors that will mainly consist of user and data input, raising an error message when the error occurs.

User Interface Layer

- UI/User Input Error: Errors that occur when a user inputs a data type that is not supported by the specified type
    - Ex: Displays an error message when the user inputs an invalid data type. System will raise an exception and a “Invalid Input.” message will display

Back-End

In the back-end portion of the application, the system will handle errors that occur within or at the database level such as CRUD operations (Create, Read, Update, Delete) consisting of the Data Access and Data Store layers along with errors within the validation of data, connections to services, and business requirements in the Manager and Service layers. The system will return empty or null values and error messages when the errors occur within the layers.

Manager Layer

- Input Validation Error: Errors that occur when the system allows the user’s HTTP request to be validated, but are not passing in another check such as authorized access
    - Ex: Displays an error message when the inputted data has been validated and is not within their given role of data access. System will raise an exception and a “Access Denied.” message will display
- Business Requirements Error: Errors that occur when the system allows the user’s HTTP request to be validated, but does not pass business requirements resulting in a failed outcome
    - Ex: Inputted data that has been validated and does not follow business rules will display an error message. System will raise an exception and a “Please Come Back Later.” message will display

Service Layer

- Workflow Error: Errors that occur through the traveling of data of the user’s HTTP request

- Ex: Displays an error message if the connection to the server is not available or offline System will raise an expectation and a “Failed to process request. Please Try Again Later.” message will display

Data Access Layer

- DAO Error: Errors that occur when the Data Access layer uses DAOs (Data Access Object) to access the database in the Data Store layer, but fails to interact with the database.
    - Ex: Will return an error if the connection to the database is not available or offline. Business layers (Manager + Service) will determine the message to be displayed as the response is being sent back through the layers to the user such as the system responding with a “Please Come Back Later.” message

Data Store Layer

- I/O Error: Errors that occur within the Data Store when the Data Access layer requests a retrieval of data. An input expects an output in which an output is created, but the database returns null, empty, or errors.
    - Null Ex: Will return null if one value is expected and no information is found within the database on a specific given facility or company reservation spot
    - Error Ex: Will return an error if there are currently no facilities or companies within the database. Business layers (Manager + Service) will determine the message to be displayed as the response is being sent back through the layers to the user such as the system responding with a “Please Come Back Later.” message
    - Empty Ex: Will return an empty array if multiple values of reservation spots of facilities or companies are expected and the database is empty

Logging Rules

Logs will be stored within a database. Database information is easily accessible and can be retrieved quickly. Having logs stored in a database allows the data to be more secure and consistent. The specific type of database that is used to store logs is a relational database. We will be using SQL Server 2019 Developer Edition that offers security, performance optimization, and scalability for a database. Compared to non-relational databases, relational databases allow for ease of access due to the structure and format of rows within the database at the cost of possibly being slower than non-relational databases.

Logs will be structured within a log interface within the Service layer. Multiple different levels of logs will have the log interface as a dependency. The logs will then query to the database storing logs depending on their level and relevant information such as timestamp, category, and description.

Default Logs

Front-End

In the front-end portion, logs will mainly consist of anything within the UI and the view of the user.

User Interface Layer (View Related Information)

View: Will log information that deals with what the user is able to see and interactions made by the user in the UI such as input being submitted when input is needed

Back-End

In the back-end portion, logs will consist of information that occurs within the Data Access, Data Store, Manager, and Service layers. This includes business requirements, data validation, data accessing and retrieval, success and failure outcomes, and errors.

Manager Layer (Business Related Information)

Business: Will log information that revolves around the business requirements as input is being sent through checks on data validation such as whether a input passes or fails a business requirement

Service Layer (Server Related Information)

Server: Will log information that revolves around the flow of the system. This includes but is not limited to, connections to the server, data travel between layers, and query executions

Data Access Layer (Data Related Information)

Data: Will log information that deals with data being sent or interacting with the database. This includes connections to the database and accessing and retrieving data within the database

Data Store Layer (Data Store Related Information)

Data Store: Will log information that is stored within the database. This includes information of interactions by the user and application, Front-End and Back-End, as well as one layer to another

Technology

Technology Stack

Front-End	ECMAScript(ES6, ES13), TypeScript(v4+), HTML5, CSS 3
Back-End	ASP.NET Core 8, C# 10+, IIS 10+, .NET 8.0, SQL Server 2019 Developer Edition
Tool Chain	Visual Studio 2022 Community Edition, Visual Studio Code 1.x, Visual Studio for Mac 17.x, MySQL Workbench
Services	AWS

Limitation/Advantages

Front-End:

Advantages:

The ECMAScript (ES6, ES13) standards provide improved code readability, maintenance, and accessibility to modern language features. Additionally, they guarantee compatibility with future browser versions. TypeScript enhances code quality, reduces runtime errors, and introduces a more structured approach to coding, essential for maintaining a stable system. HTML5 allows the integration of multimedia elements, making it more accessible across all browsers and platforms, and introduces offline web application features. CSS3 facilitates styling, layout techniques, animations, transitions, and responsive design, improving both user experience and visual appeal.

Limitations:

Updates to ECMAScript's language features may cause compatibility issues with older browsers. For developers unfamiliar with static typing, TypeScript's learning curve may be challenging, but it ultimately improves code quality and maintainability. Achieving consistency in browser implementations for HTML5 may affect the rendering of certain features. While widely accepted, some older browsers might not support all CSS3 features, necessitating fallbacks or alternative styling approaches.

Back-End:

Advantages:

Using ASP.NET Core 8 supports cross-platform development, enabling applications to run on various operating systems. It includes support for MVVM (Model-View-ViewModel) architecture for creating online apps and web APIs. C# 10+ is a strongly typed language that promotes clean code and maintainability. IIS 10+ provides a server for hosting ASP.NET applications, allowing the creation of system and application administrator accounts for

granular-level access. .NET 8.0 enables cross-platform development, offering libraries and frameworks to improve productivity and code reuse. SQL Server 2019 Developer Edition allows the use of a relational database management system, providing security, performance optimization, and scalability for a data store.

Limitations:

Migration from older ASP.NET versions to ASP.NET Core 8 may require careful considerations and testing due to potential incompatibilities. For developers unfamiliar with object-oriented programming, there may be a learning curve when working with C# 10+. Challenges may arise concerning platform-specific features or dependencies. IIS 10+ is primarily intended for Windows-based operating systems, limiting its use on non-Windows platforms. Due to the extensive features of .NET 8.0, novices might find the learning curve difficult. Some .NET applications may have higher memory consumption than more lightweight frameworks. The use of SQL Server 2019 Developer Edition is intended for testing and development, not for production settings, which require a production edition license.

Tool Chain:

Advantages:

Visual Studio 2022 Community Edition, as the primary IDE for .NET development, supports cross-platform development, allowing developers to build applications for various platforms. It provides tools for building, testing, and deploying .NET applications, including ASP.NET web applications. With support for the latest language features and standards, including C# 10 and .NET 8, Visual Studio 2022 enables developers to make full use of the capabilities of multiple programming languages. Visual Studio Code 1.x, another cross-platform IDE, comes with built-in support for version control systems like Git, allowing developers to work directly within the editor. Visual Studio for Mac 17.x, designed for MacOS developers, is similar to Visual Studio Code 1.x. MySQL Workbench, used for accessing the database, is an open-source database tool, free to use and easy to adopt. It is supported across multiple platforms, giving developers flexibility in a diverse development environment. It also includes security features, such as user authentication, access controls, and encryption, to secure sensitive data and fulfill compliance requirements.

Limitations:

Developers using Visual Studio 2022 Community Edition may lack several advanced features found in the Enterprise edition, such as some testing tools and collaboration capabilities. Upgrading may be necessary for developers requiring these functionalities. Visual Studio Code 1.x may require more configuration than other code editors, potentially posing a drawback for

those who prefer a more simplified setup. Features available for Visual Studio Code for Windows may not be present in Visual Studio for Mac 17.x. Developers working on resource-intensive projects and/or running several extensions may need to consider hardware specifications, as Visual Studio IDEs consume a significant amount of system resources, leading to performance issues. While MySQL Workbench is used for easy setup and management of the database, it can result in performance bottlenecks, strict schemas, and a lack of GUI tools. Queries with multiple joins can slow the application, increase query times, and lead to storage issues.

Services:

Advantages:

With AWS, we can choose the web application platform, database, operating system, programming language, and other services that are required. It loads necessary software and services for the application in a virtual environment provided by AWS. AWS offers a pay-as-you-go pricing model, allowing users to pay only for the resources they consume, rather than investing in expensive servers. With its secure infrastructure and tools for identity management, encryption, and compliance, AWS places a high priority on security.

Limitations:

For new users, a great variety of AWS services may result in a challenging learning curve. The complexity of these services might require advanced training and expertise. Applications hosted on AWS are directly impacted by any downtime on the platform. Usage-based fees may apply according to the amount of resources consumed, which can affect costs.