# 

# 

# 

# 

# 

# 

# CSC 431

# 

# **Time for Gym**

# 

# 

# **System Architecture Specification (SAS)**

**Team 10**

| Ross Stewart | Developer |
| --- | --- |
| James Johnson | Developer |

# 

# 

# 

# 

# 

# **Version History**

| **Version** | **Date** | **Author(s)** | **Change Comments** |
| --- | --- | --- | --- |
| **1.0** | 4/4/23 | Ross Stewart,  James Johnson | First Draft |
| **1.0.1** | 4/6/23 | Ross Stewart,  James Johnson | Revisions for extended deadline |
| **1.1** | 5/5/23 | Ross Stewart,  James Johnson | Final Draft |
|  |  |  |  |

# 

# 

# 

# 

# 

# 

# 

# 

# **Table of Contents**

1. System Analysis 4

1.1 System Overview 4

1.2 System Diagram 4

1.3 Actor Identification 5

1.4 Design Rationale 5

1.4.1 Architectural Style 5

1.4.2 Design Pattern(s) 5

1.4.3 Framework 5

2. Functional Design 8

2.1 Sequence Diagram 8

3. Structural Design 10

# 

# 

# 

# 

# 

# 

# **Table of Figures**

1.2 System Diagram 4

2. Functional design 8

2.1 Sequence Diagram 8

3. Structural Design 10

3.1 Class Diagram 10

### 

### 

### 

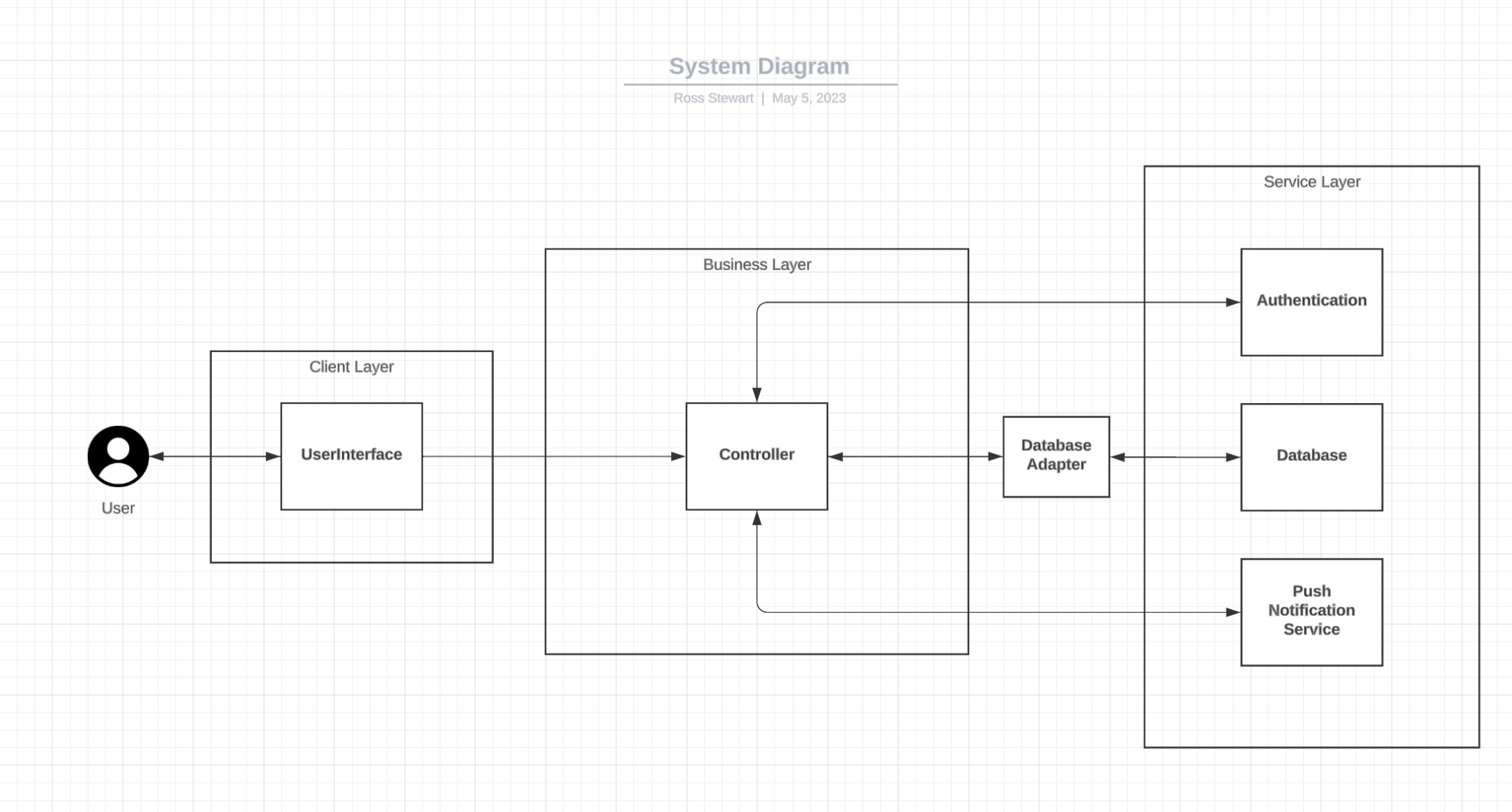
### 

### **1.** **System Analysis**

#### **1.1** **System Overview**

The system will be made up of three parts, The User Interface, The Database, and a controller. The User interface will allow the user to view the entire application, for example they will be able to view the different exercises and gym wait times. The database will allow us to store different exercises and the videos for them. The controller will help guide our search function and allow us to be able to show the correct information on the UI.

#### **1.2** **System Diagram**



The diagram above shows the three-tier architecture of the system. The user only interacts with UserInterface. The UserInterface interacts with the business layer. The business layer interacts with the third-party software and databases.

#### 

#### **1.3** **Actor Identification**

* User: Will be allowed to view exercises and favorite them. Will also be allowed to use the wait time functionality as well as view different exercises.
* System Administrator: Will be able to make changes to the system as needed and update different functions and features.

#### **1.4** **Design Rationale**

##### **1.4.1** **Architectural Style**

We are using a three tier architecture for our Application, because we need a good client-based user interface connected to the server to retrieve our data and other information without cluttering the UI. This will allow for use of both AWS and the React Framework for an uninterrupted user experience.

The Business logic tier will be able to allow us to determine the correct information to present to the user based on what they select. The service tier allows for us to “talk” to our databases and provide security, this would be done with AWS.

##### **1.4.2** **Design Pattern(s)**

We will use the strategy design pattern which will allow us to manipulate data, predictions, and analysis all within our back end. To achieve these goals we must use different algorithms interchangeably.

##### **1.4.3** **Framework**

We plan on using React Native for our framework because it is well known and documented, and allows us flexibility. It will also allow us to develop the app for the different platforms we plan to release it on with ease. React is also easy to combine with APIs and AWS which is another big part of making this app work.

### 

### **2.** **Functional Design**

### **2.1** **Sequence Diagram**

This diagram depicts the sequence of actions a user could take when using the application.

### **3.** **Structural Design**

### **3.1** **Class Diagram**

The following is a UML Class Diagram that represents the core functionality of our system.

### 

The user interacts with the UserInterface class, which interacts with the Controller class. The controller class handles the database interaction and well as the user authentication.