Bilkent University

# Senior Design Project

*Project short-name: GymFeat - AI Training Coach*

Low Level Design Report

Talha Burak Çuhadar, Mustafa Çağrı Güngör, Ayşe Ezgi Yavuz, Gonca Yılmaz, Ravan Aliyev

Supervisor: Halil Altay Güvenir

Innovation Expert: Emin Okutan

Feb 8, 2021

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

# 1. Introduction

Given the importance of healthy life, sports itself play a prominent role to develop a healthy body. Not only it helps people to develop the muscle mass, but it also improves the health of one's mind [1]. According to the researches about how sport affects the health of mind, it is shown that doing exercises "positively impacts the level of serotonin, a chemical that helps regulate mental health", reduces the level of stress, improves mood, and distracts people from negative thoughts [2, 3]. In addition to these impacts of sport on mind, it also boosts the immune system [4]. Therefore, we should always prioritize doing exercises in our lives.

In normal circumstances, some people would prefer to go to the gym, and some people would prefer to do exercises at home and there used to be a trade-off between them. However, since the beginning of the pandemic, people are seeking new means to stay active inside the house, because many parts of our lives remain restricted. At this moment "maintaining an exercise routine at home can seem more like a 'should' than a 'want to'" says Shannon Collins, an Integrative Manual Physical Therapist [4]. We, as a team, would like to encourage people to be more active at home.

Nevertheless, when people work out at home without the guidance of an expert, there is a risk of getting injured. Imagine yourself, trying to preserve your healthy body and mind, and while doing so, getting injured. Our project idea started out with the question, how we can prevent people suffering from exercise-related injuries, i.e. sprains, muscle strains, tendinitis, and so on. According to an article from Harvard Health Publishing, people should choose their workout carefully, learn the proper technique in order to prevent injuries and drink water to stay hydrated [5]. Hence, we would like to address these issues and provide a mobile application, which will prepare a training program for you, check your body movement to make sure that you are applying each movement correctly, and remind you of drinking water during the training.

## 1.1 Object design trade-offs

### 1.1.1 Understandability vs. Complexity

Our application is designed with the aim of providing users an opportunity to complete their training with only one application. In order to achieve this purpose, we included multiple functionalities during the design process including the gym program and AI trainer However, it might be problematic for users to learn the use of all of the functionalities at the beginning. In order to find the balance between understandability and complexity, we limited the number of functionalities.

### 1.1.2 Portability vs. Development Time

In order to ensure that GymFeat is easily portable and available on multiple platforms, we decided to use the Node.js and Ionic frameworks which enable us to deploy the resulting product into multiple platforms including the web and mobile. We decided to continue with these frameworks considering the trade off between the importance of portability and the longer development time.

### 1.1.2 Pose Estimation Accuracy vs. Memory

GymFeat is an app that estimates the pose of the users to analyze their training to give them appropriate feedback. For pose estimation, we decided to use the Resnet50 model which requires memory more than 100 MB because it estimates the pose with higher accuracy.

### 1.1.3 Reliability vs Rapid Development

We are planning to start with a rapid and agile development strategy to have a minimum viable product in our hands that might not be reliable enough to satisfy our project requirements. Then, we will gradually work on this product to increase its reliability and other functionalities. To sum up, rapid development is favored in the first stage of the project, and reliability is considered in the later stages.

## 1.2 Interface documentation guidelines

This report follows the generic guidelines for class interfaces. The class names are written as "ClassName". As for variables, they are written in the standard form "variableName" and method names are shown as "methodName()". As for constructors, the naming is done as "ClassName()" and they are shown under "Methods". In the class interface, first the class names are written, then the properties attributed to that specific class, and finally the methods. The class interface is shown as the following:

| class ClassName |
|---|
| The description of the class is written here. |
| Properties |

| |
|---|
| Properties are listed here.<br><br>private int variableName1<br><br>public String variableName2 |
| Methods |
| Methods are listed here.<br><br>public methodName1(): the functionality of the method 1<br><br>private methodName2(): the functionality of the method 2 |

## 1.3 Engineering standards (e.g., UML and IEEE)

In this report, the IEEE standards will be used for citations and references, since IEEE standards are widely used for reports in the field of computer science [6]. For the diagrams, UML guidelines are followed [7]. Because UML is used as a general-purpose modeling language often in the software engineering field [8].

## 1.4 Definitions, acronyms, and abbreviations

- Back-end: The back end of a website consists of a server, an application, and a database [9].
- Class: “A generalized description of a project” [10]
- Client: A computer (Host), which is capable of using a particular service or receiving new information from servers, or in other words service providers [11].
- Controller: The controller of Model and View, it accepts input from the user and also updates the necessary components [12].
- Method: Indicates the behavior of the objects associated with a class [13].
- Model: Data used by program, i.e. database, file, or a simple object [12].
- Property: The attributes associated with a class [14].
- Server: A remote computer, which provides information, or services to the Client [11].
- UI: It is an abbreviation for User Interface, that is where the human-computer interaction and communication occurs [15].

- View: The means of displaying an object within the application, basically anything the user can see [12].

# 2. Packages

## 2.1 Client

The client consists of 3 subsystems respectively Controller subsystem, View subsystem and Data subsystem. Due to the architecture of our application, the controller subsystem consists of the controllers that are without views. Whereas, the view classes will be mentioned as managers. The view classes will hold their controllers within them. The controller subsystem will be responsible for the operations done to data, sending data to the server and when responses are taken for requests, the Controller collects it. View subsystem will be responsible for interface operations and again, managing the communication between the view itself and the corresponding server element. Displaying pages or taken data on the screen will be done by the view subsystem. Data subsystem handles local storage for the output photos of users.

### 2.1.1 Controller

The controller subsystem is responsible for handling the events that are received from the UI components that will be mentioned in the View.

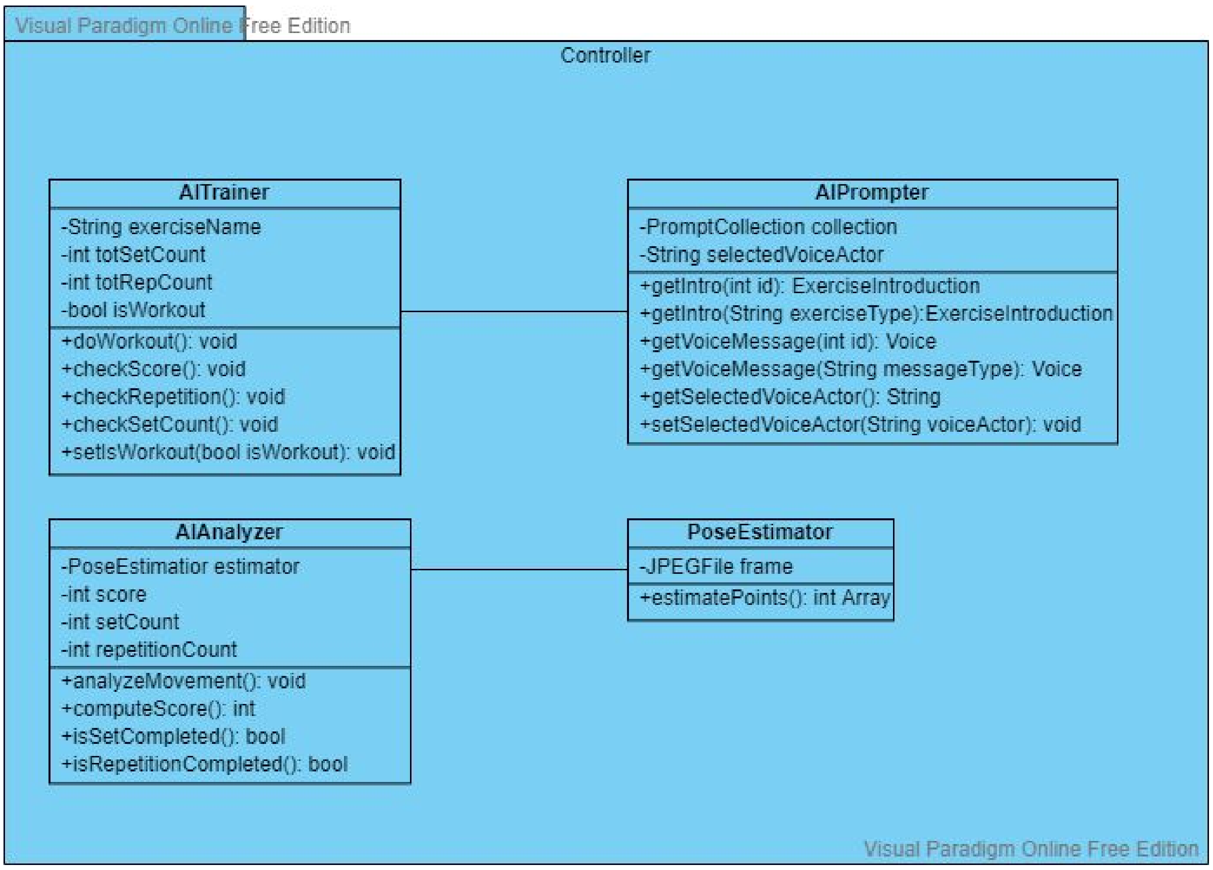
Visual Paradigm Online Free Edition
Controller
AITrainer
-String exerciseName
-int totSetCount
-int totRepCount
-bool isWorkout
+doWorkout(): void
+checkScore(): void
+checkRepetition(): void
+checkSetCount(): void
+setIsWorkout(bool isWorkout): void
AIPrompter
-PromptCollection collection
-String selectedVoiceActor
+getIntro(int id): ExerciseIntroduction
+getIntro(String exerciseType):ExerciseIntroduction
+getVoiceMessage(int id): Voice
+getVoiceMessage(String messageType): Voice
+getSelectedVoiceActor(): String
+setSelectedVoiceActor(String voiceActor): void
AIAnalyzer
-PoseEstimatior estimator
-int score
-int setCount
-int repetitionCount
+analyzeMovement(): void
+computeScore(): int
+isSetCompleted(): bool
+isRepetitionCompleted(): bool
PoseEstimator
-JPEGFile frame
+estimatePoints(): int Array
Visual Paradigm Online Free Edition

***Figure 1: Controller Package Diagram***

***AITrainer***

This class is responsible for fetching the analysis of the movement from *AIAnalyzer* and communicating with the *AIPrompter*.

***AIPrompter***

This class is responsible for prompting the user the necessary introductions and warnings related to the exercise.

***AIAnalyzer***

This class uses the PoseEstimator and according to the pose estimation, it analyzes the movements, creates score and feedback, and counts the set and the repetitions of a certain exercise.

***PoseEstimator***

This class gets the data from CamView and estimates the pose.

## 2.1.2 View

The view subsystem is responsible for receiving and arranging inputs and posting them to the server and also receiving, arranging and displaying the data sent from the server.

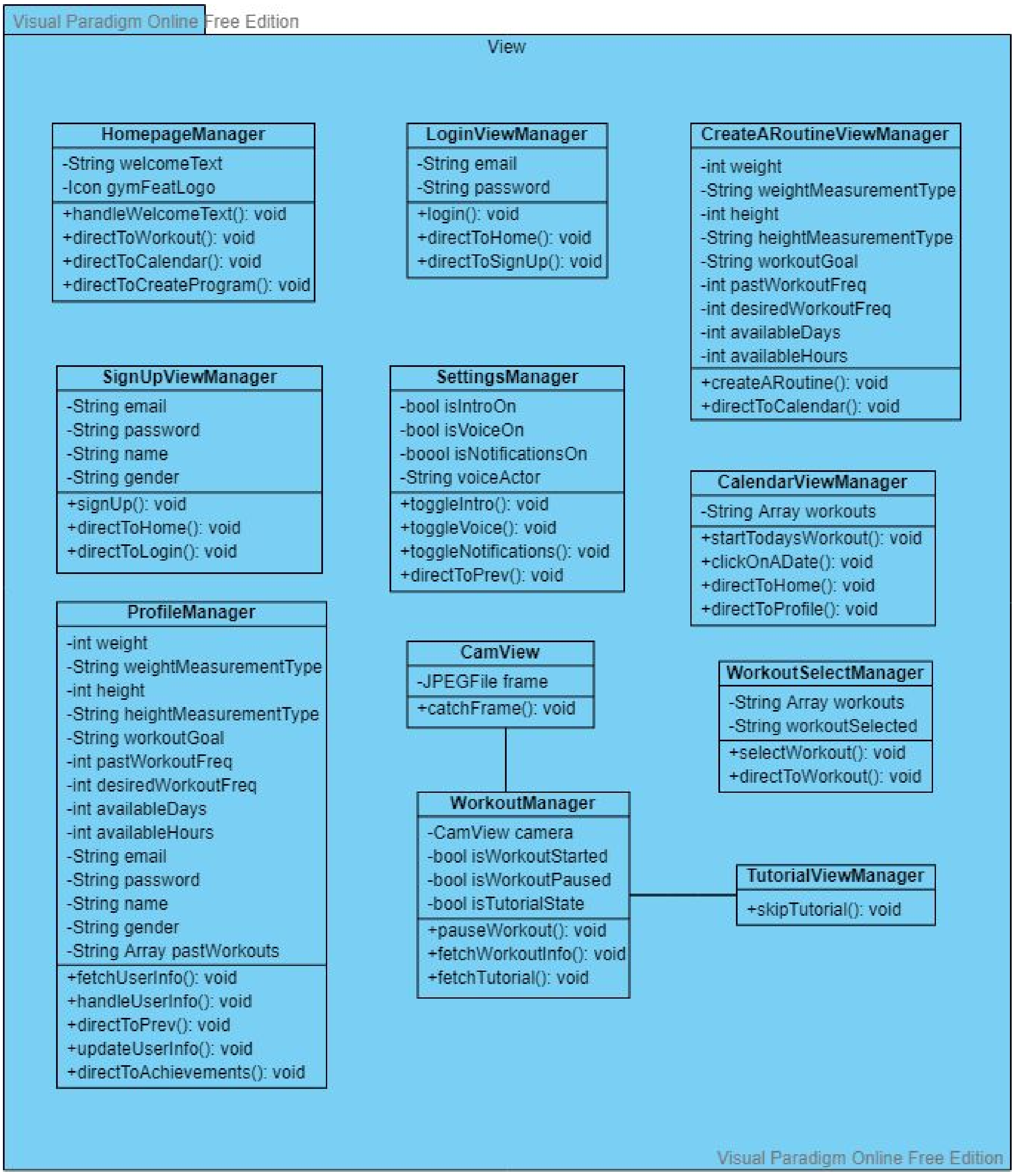

***Figure 2: View Package Diagram***

***HomepageManager***

This view is the main view that will display the information about the application mission.

***LoginViewManager***

This view is to receive input from the user for login.

***SignUpViewManager***

This view is to receive user information for signing up the user.

***SettingsViewManager***

This class is responsible for allowing the user to change the settings of the application.

***CreateARoutineViewManager***

This view is to receive user input for arranging a workout program for them.

***CalendarViewManager***

This view is the main page after sign-in of the user (whether anonymously or with profile). It will display a real calendar with the user's workouts.

***ProfileManager***

This view is responsible to display the logged in user information. It will allow the user to modify the information. It will also display the finished workouts by the user.

***CamView***

This view is using the user webcam to detect the movement of users and send the movements to the *PoseEstimator*.

***WorkoutSelectManager***

With this class, the user is able to select workouts and download them to ease access during the workout session.

***WorkoutManager***

This class is responsible for managing the workout session and fetching the selected workout for the user.

***TutorialViewManager***

This class is responsible for showing the tutorial video to the user.

***AchievementsViewManager***

This class is responsible for the view of achievements.

### 2.1.3 Data

This subsystem manages the local storage, in which the introduction videos and the voices related to a certain exercise are kept.

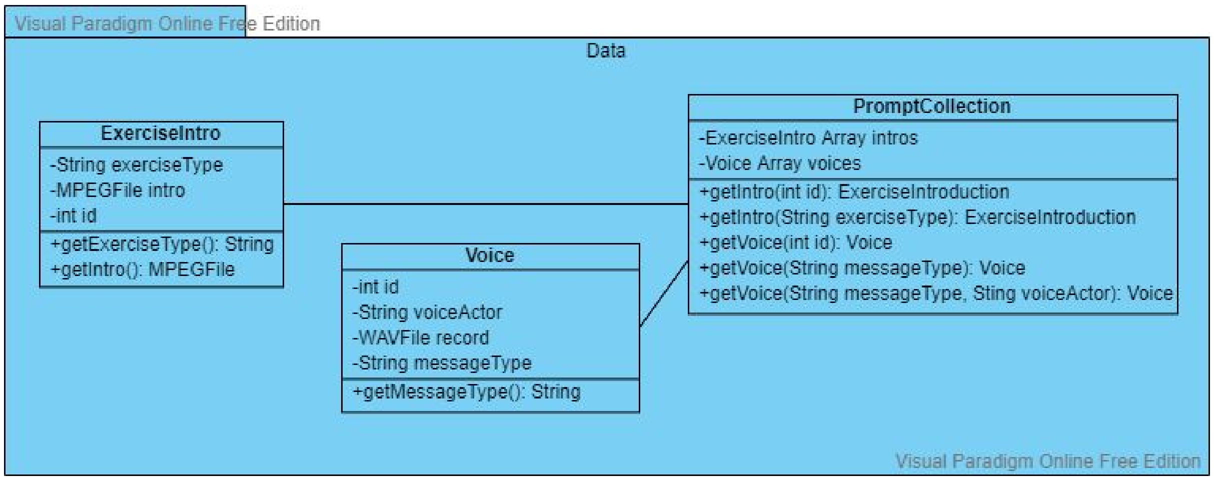
Visual Paradigm Online Free Edition
Data
ExerciseIntro
-String exerciseType
-MPEGFile intro
-int id
+getExerciseType(): String
+getIntro(): MPEGFile
Voice
-int id
-String voiceActor
-WAVFile record
-String messageType
+getMessageType(): String
PromptCollection
-ExerciseIntro Array intros
-Voice Array voices
+getIntro(int id): ExerciseIntroduction
+getIntro(String exerciseType): ExerciseIntroduction
+getVoice(int id): Voice
+getVoice(String messageType): Voice
+getVoice(String messageType, Sting voiceActor): Voice
Visual Paradigm Online Free Edition

***Figure 3: Data Package Diagram***

***PromptCollection***

This class holds the array of ExerciseIntroduction, which includes the information and the video file of each introduction video, and the array of Voice, which consists of the recording and the related information to that voice.

***ExerciseIntroduction***

This class is responsible for holding the introduction video of a single exercise and the related information to that single exercise.

***Voice***

This class is responsible for keeping the voice recording and information of the recording.

## 2.2 Server

### 2.2.1 Controller

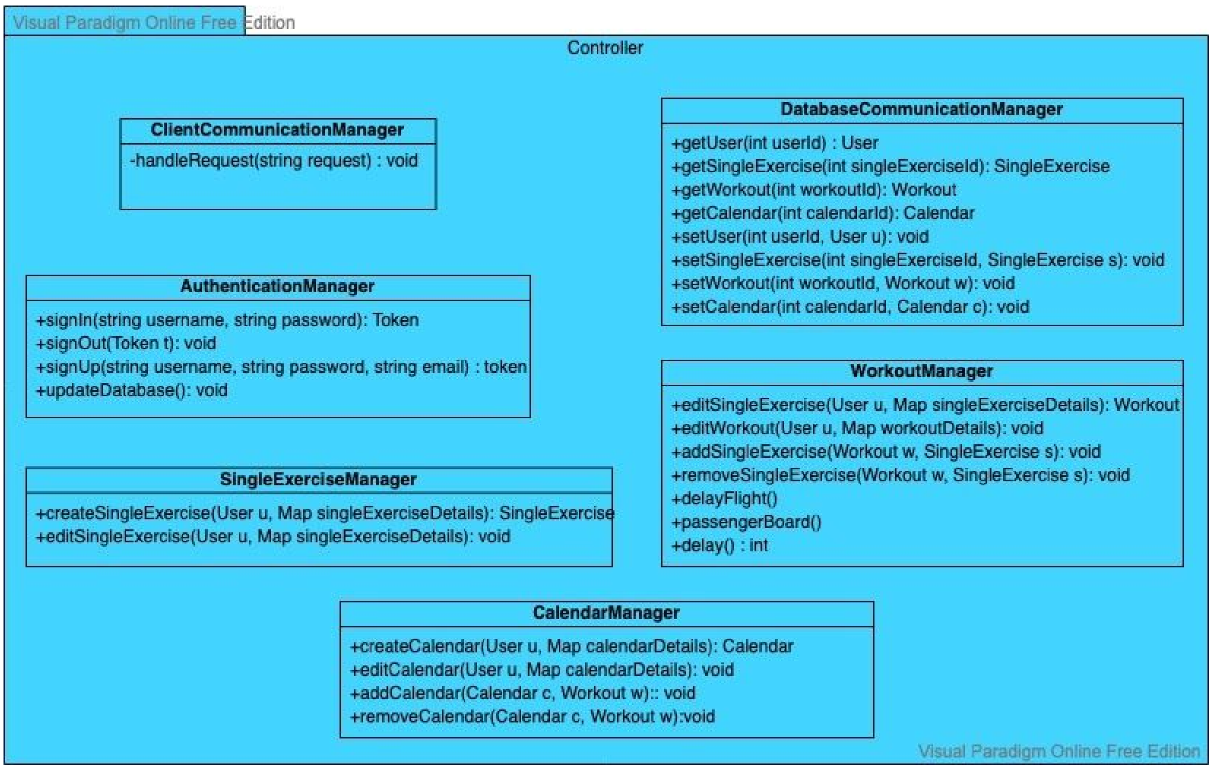
Visual Paradigm Online Free Edition
Controller
ClientCommunicationManager
-handleRequest(string request) : void
DatabaseCommunicationManager
+getUser(int userId) : User
+getSingleExercise(int singleExerciseId): SingleExercise
+getWorkout(int workoutId): Workout
+getCalendar(int calendarId): Calendar
+setUser(int userId, User u): void
+setSingleExercise(int singleExerciseId, SingleExercise s): void
+setWorkout(int workoutId, Workout w): void
+setCalendar(int calendarId, Calendar c): void
AuthenticationManager
+signIn(string username, string password): Token
+signOut(Token t): void
+signUp(string username, string password, string email) : token
+updateDatabase(): void
WorkoutManager
+editSingleExercise(User u, Map singleExerciseDetails): Workout
+editWorkout(User u, Map workoutDetails): void
+addSingleExercise(Workout w, SingleExercise s): void
+removeSingleExercise(Workout w, SingleExercise s): void
+delayFlight()
+passengerBoard()
+delay() : int
SingleExerciseManager
+createSingleExercise(User u, Map singleExerciseDetails): SingleExercise
+editSingleExercise(User u, Map singleExerciseDetails): void
CalendarManager
+createCalendar(User u, Map calendarDetails): Calendar
+editCalendar(User u, Map calendarDetails): void
+addCalendar(Calendar c, Workout w):: void
+removeCalendar(Calendar c, Workout w):void
Visual Paradigm Online Free Edition

*Figure 4: Controller Package Diagram*

***ClientCommunicationManager***

This class is responsible for managing the communication between the server and the client. Each request made to the server side will be routed in this class.

***DatabaseCommunicationManager***

This class is responsible for managing the communication between the server's logic tier and data tier. This class will create objects from the data fetched from the database.

***AuthenticationManager***

This class is responsible for carrying out authentication activities such as signing in, signing up, signing out etc.

***SingleExerciseManager***

This class is responsible for creating and editing single exercises for a user.

***WorkoutManager***

This class is responsible for creating and editing workouts for a user.

***CalendarManager***

This class is responsible for managing the workout program of a certain user.

### 2.2.2 Model

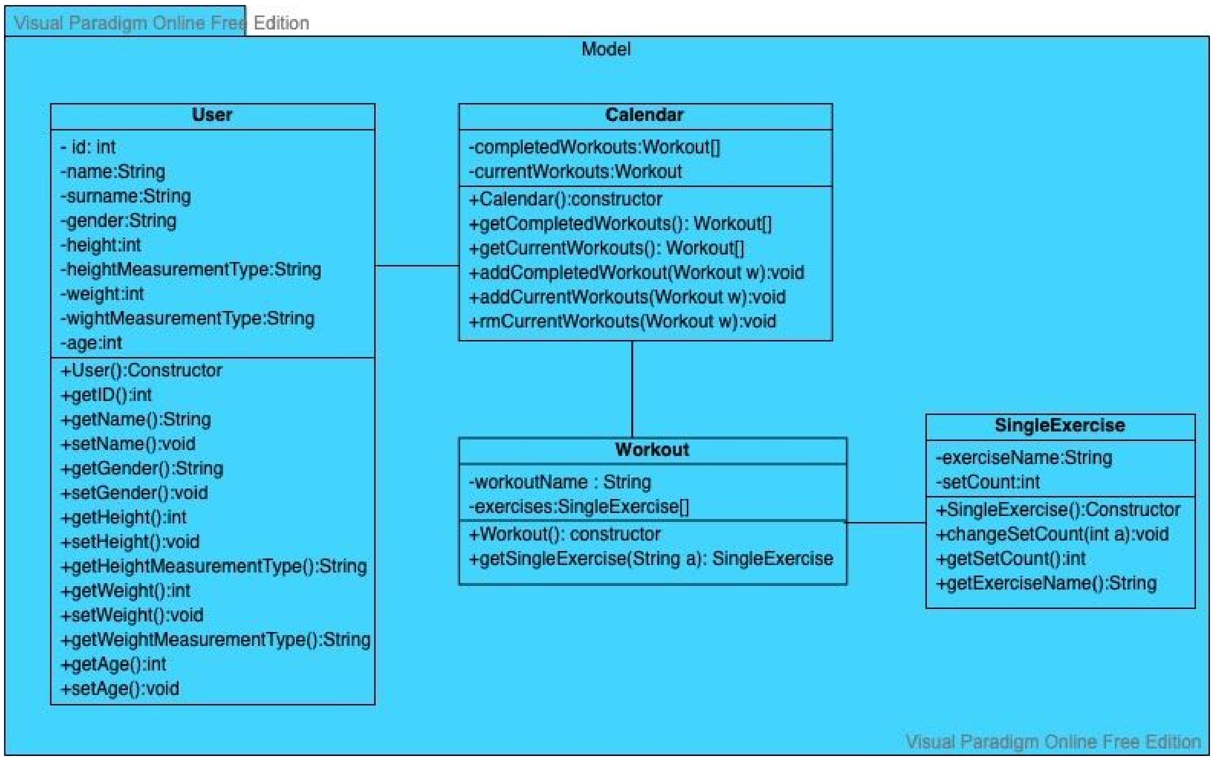
Visual Paradigm Online Free Edition
Model
User
- id: int
-name:String
-surname:String
-gender:String
-height:int
-heightMeasurementType:String
-weight:int
-wightMeasurementType:String
-age:int
+User():Constructor
+getID():int
+getName():String
+setName():void
+getGender():String
+setGender():void
+getHeight():int
+setHeight():void
+getHeightMeasurementType():String
+getWeight():int
+setWeight():void
+getWeightMeasurementType():String
+getAge():int
+setAge():void
Calendar
-completedWorkouts:Workout[]
-currentWorkouts:Workout
+Calendar():constructor
+getCompletedWorkouts(): Workout[]
+getCurrentWorkouts(): Workout[]
+addCompletedWorkout(Workout w):void
+addCurrentWorkouts(Workout w):void
+rmCurrentWorkouts(Workout w):void
Workout
-workoutName : String
-exercises:SingleExercise[]
+Workout(): constructor
+getSingleExercise(String a): SingleExercise
SingleExercise
-exerciseName:String
-setCount:int
+SingleExercise():Constructor
+changeSetCount(int a):void
+getSetCount():int
+getExerciseName():String
Visual Paradigm Online Free Edition

***Figure 5: Model Package Diagram***

***User***

This class is a model data class responsible for holding user data such as name, surname, height, weight, date of birth etc.

***SingleExercise***

This class is a model data class responsible for holding data of a single exercise.

***Workout***

This class is a model data class responsible for holding workout data. A workout consists of several single exercises.

***Calendar***

This class is a model data class, which is responsible for holding the workout schedule information.

# 3. Class Interfaces

## 3.1 Client

### 3.1.1 Controller

| class AITrainer |
|---|
| This class is responsible for fetching the analysis of the movement from *AIAnalyzer* and communicating with the *AIPrompter*. |
| Properties |
| private String exerciseName<br>private int totSetCount<br>private int totRepCount<br>public boolean isWorkout |
| Methods |
| public void doWorkout(): it constantly checks the score, the repetition count, and the set count and when necessary, calls the *AIPrompter*<br>private void checkScore(): checks the score and if it is below a threshold, then calls the AIPrompter to prompt a warning to user, otherwise it calls the *AIPrompter* to prompt a motivating message<br>private void checkRepetition(): this function checks the repetition count and whenever a repetition is completed, it calls *AIPrompter* to prompt a message to the user<br>private void checkSetCount(): this function checks the set count and whenever a set is completed, it calls *AIPrompter* to prompt a message to the user<br>public void setIsWorkout(boolean isWorkout): sets the property, *isWorkout*. |

| class AIPrompter |
|---|
| This class is responsible for prompting the user the necessary introductions and warnings related to the exercise and it communicates with AITrainer. |
| Properties |
| private PromptCollection collection<br><br>private String selectedVoiceActor |
| Methods |
| private ExerciseIntroduction getIntro(int id): returns the exercise introduction by id<br><br>public ExerciseIntroduction getIntro(String exerciseType): this function computes the id of the desired exercise introduction, calls getIntro(int id) method and returns the exercise introduction by exercise type<br><br>private Voice getVoiceMessage(int id): returns the *Voice* object by id<br><br>public Voice getVoiceMessage(String messageType): this function computes the id of the desired voiceMessage, calls getVoiceMessage(int id) method and returns the *Voice* object by message type<br><br>public String getSelectedVoiceActor(): returns the name of the selected voice actor<br><br>public void setSelectedVoiceActor(String voiceActor): sets the voice actor |

| class AIAnalyzer |
|---|
| This class uses the PoseEstimator and according to the pose estimation, it analyzes the movements, creates score and feedback, and counts the set and the repetitions of a certain exercise. |
| Properties |
| private PoseEstimator estimator<br><br>private int score<br><br>private int setCount |

| |
|---|
| private int repetitionCount |
| Methods |
| public void analyzeMovement(): this function analyzes the movement by computing score, checking if the current repetition, and the current set is completed.<br><br>private int computeScore(): returns the score<br><br>private boolean isSetCompleted(): returns true, if the current set is completed<br><br>private boolean isRepetitionCompleted(): returns true, if the current repetition is completed |

| class PoseEstimator |
|---|
| This class gets the data from CamView and estimates the pose. |
| Properties |
| private JPEGFile frame |
| Methods |
| public int Array estimatePoints(): returns the array of estimated location of body points on the frame |

### 3.1.2 View

The view classes for the application are explained. The *render* function is left out since it is the default function for displaying components on screen hence it is included in all of the views.

| class HomePageManager |
|---|
| This view is the main view that will display the information about the application mission. |
| Properties |
| private String welcomeText<br><br>private Icon gymfeatLogo |

| Methods |
|---|
| private void handleWelcomeText(): creates and handles the welcome text for displaying<br><br>private void directToWorkout(): directs user to workout list page<br><br>private void directToCalendar(): directs user to calendar page<br><br>private void directToCreateProgram(): directs user to create a routine page |

| class LoginViewManager |
|---|
| This view is to receive input from the user for login. |
| Properties |
| private String email<br><br>private String password |
| Methods |
| public void login(): sends the information to back-end to log the user in<br><br>public void directToHome(): directs user to the home page<br><br>public void directToSignUp(): directs user to the sign up page |

| class SignUpViewManager |
|---|
| This view is to receive user information for signing up the user. |
| Properties |
| private String email<br><br>private String password<br><br>private String name<br><br>private String gender |
| Methods |

| |
|---|
| public void signUp(): sends the information to back-end to sign the user up<br><br>public void directToHome(): directs user to the home page<br><br>public void directToLogin(): directs user to the login page |

| class SettingsManager |
|---|
| This class is responsible for allowing the user to change the settings of the application. |
| Properties |
| private boolean isIntroOn<br><br>private boolean isVoiceOn<br><br>private boolean isNotificationsOn<br><br>private String voiceActor |
| Methods |
| public void toggleIntro(): turns on the introduction tutorials at the beginning of each single exercise<br><br>public void toggleVoice(): turns on the voice to get warnings and prompts during the workout<br><br>public void toggleNotifications(): toggles the notification preference<br><br>public void selectVoiceActor(): selects the<br><br>public void directToPrev(): directs the user to the previous page |

| class CreateARoutineViewManager |
|---|
| This view is to receive user input for arranging a workout program for them. |
| Properties |
| private int weight<br><br>private String weightMeasurementType |

| |
|---|
| private int height<br><br>private String heightMeasurementType<br><br>private String workoutGoal<br><br>private int pastWorkoutFreq<br><br>private int desiredWorkoutFreq<br><br>private int availableDays<br><br>private int availableHours |
| Methods |
| public void createARoutine(): sends the information to the back-end<br><br>public void directToCalendar(): directs user to the calendar page |

| class CalendarViewManager |
|---|
| This view is the main page after sign-in of the user (whether anonymously or with profile). It will display a real calendar with the user's workouts. |
| Properties |
| private String Array workouts |
| Methods |
| public void startTodaysWorkout(): directs user to workout page<br><br>public void directToHome(): directs user to home page<br><br>public void clickOnADate(): shows the workout information of that date to the user<br><br>public void directToProfile(): directs user to the profile page |

| class ProfileManager |
|---|

| |
|---|
| This view is responsible to display the logged in user information. It will allow the user to modify the information. |
| Properties |
| private String email<br>private String password<br>private String name<br>private String gender<br>private int weight<br>private String weightMeasurementType<br>private int height<br>private String heightMeasurementType<br>private String workoutGoal<br>private int pastWorkoutFreq<br>private int desiredWorkoutFreq<br>private int availableDays<br>private int availableHours<br>private String Array pastWorkouts |
| Methods |
| pripublicvate void fetchUserInfo(): fetches current user's information<br>public void handleUserInfo(): handles user information to show<br>public void updateUserInfo(): updates the user information that is changed by the user<br>public void directToPrev(): directs user to the previous page<br>public void directToAchievements(): directs user to achievements view |

| class AchievementsViewManager |
|---|

| |
|---|
| This class is responsible for the view of achievements. |
| Properties |
| private String Array achievements<br><br>private Icon Array achievementLogos |
| Methods |
| public void fetchAchievements(): fetches the achievements related to a certain user<br><br>public void fetchAchievementLogos(String Array achievements): fetches the logos of achievements<br><br>public void directToProfile(): directs the user back to the profile page |

| class CamView |
|---|
| This view is using the user webcam to detect the movement of users and send the movements to the *PoseEstimator*. |
| Properties |
| private JPEGFile frame |
| Methods |
| public void catchFrame(): catches the frame and sends it to the controller for further process |

| class WorkoutSelectManager |
|---|
| With this class, the user is able to select workouts and download them to ease access during the workout session. |
| Properties |
| private String Array workouts<br><br>private String workoutSelected |
| Methods |
| public void selectWorkout(): selects the workout that is chosen by the user and shows its |

| |
|---|
| details<br><br>public void directToWorkout(): directs the user to a specific workout |

| class WorkoutManager |
|---|
| This class is responsible for managing the workout session and fetching the selected workout for the user. |
| Properties |
| private CamView camera<br><br>private boolean isWorkoutStarted<br><br>private boolean isWorkoutPaused<br><br>private boolean isTutorialState |
| Methods |
| public void pauseWorkout(): pauses the workout in process<br><br>public void startWorkout(): starts the workout<br><br>public void fetchTutorial(String exerciseType): fetches the tutorial if “Show Tutorial” is on, and makes isTutorialState true<br><br>public void fetchWorkoutInfo(): fetches the information of the workout |

| class TutorialViewManager |
|---|
| This class is responsible for showing the tutorial video to the user. |
| Properties |
| |
| Methods |
| public void skipTutorial(): this function skips the tutorial. |

### 3.1.3 Data

| class PromptCollection |
|---|
| This class holds the array of ExerciseIntroduction, which includes the information and the video file of each introduction video, and the array of Voice, which consists of the recording and the related information to that voice. |
| Properties |
| private ExerciseIntroduction Array intros<br><br>private Voice Array voices |
| Methods |
| public ExerciseIntroduction getIntro(int id): returns the ExerciseIntroduction according to the id<br><br>public ExerciseIntroduction getIntro(String exerciseType): returns the ExerciseIntroduction according to the type of the exercise<br><br>public Voice getVoice(int id): returns the voice according to its id<br><br>public Voice getVoice(String messageType): returns the voice according to the message type, whose voice actor is the default one<br><br>public Voice getVoice(String messageType, String voiceActor): returns the voice according to the message type and the voice actor |

| class ExerciseIntroduction |
|---|
| This class is responsible for storing the introduction and the related single exercise type. |
| Properties |
| private int id<br><br>private String exerciseType<br><br>private MPEGFile intro |
| Methods |
| public int getId(): returns the id of the exercise introduction<br><br>public String getExerciseType(): returns the exerciseType property of the class<br><br>public MPEGFile getIntro(): returns the MPEGFile intro video of the class |

| class Voice |
|---|
| This class is responsible for storing the voice and the related information. |
| Properties |
| private int id<br><br>private String messageType<br><br>private String voiceActor<br><br>private WAVFile record |
| Method |
| public int getId(): returns the id of the voice<br><br>public String getMessageType(): returns the messageType property of the class<br><br>public String getVoiceActor(): returns the voice actor of the voice<br><br>public WAVFile getRecord(): returns the record |

## 3.2 Server

### 3.2.1 Controller

| class ClientCommunicationManager |
|---|
| This class is responsible for handling the request made to the server and sending its response. |
| Properties |
| |
| Methods |
| public void handleRequest(string request): receives the request in string format and handles and routes the request to the other classes. |

| class DatabaseCommunicationManager |
|---|
| This class is responsible for communicating with the database and creating model classes using the data stored in the database. |
| Properties |
| |
| Methods |
| public User getUser(int userId): using the userId, fetches the user data from the database, creates a user object and returns the user object.<br><br>public SingleExercise getSingleExercise(int singleExerciseId): using the singleExerciseId, fetches the single exercise data from the database, creates a single exercise object and returns the object.<br><br>public Workout getWorkout(int workoutId): using the workoutId, fetches the workout data from the database, creates a workout object and returns the object.<br><br>public Calendar getCalendar(int calendarId): using the calendarId, fetches the calendar data from the database, creates a calendar object and returns the object.<br><br>public void setUser(int userId, User u): takes the edited user information and updates the user data in the database using the userId. If the userId is not existent, creates a new user.<br><br>public void setSingleExercise(int singleExerciseId, SingleExercise s): takes the edited single exercise and updates the single exercise data in the database using the singleExerciseId.  If the singleExerciseId is not existent, creates a new single exercise.<br><br>public void setWorkout(int workoutId, Workout w): takes the edited workout information and updates the workout data in the database using the workoutId. If the workoutId is not existent, creates a new workout.<br><br>public void setCalendar(int calendarId, Calendar c): takes the edited calendar information and updates the calendar data in the database using the calendarId. If the calendarId is not existent, creates a new calendar. |

| class AuthenticationManager |
|---|

| |
|---|
| This class is responsible for authentication use cases such as signing in, signing out and signing up. |
| Properties |
| |
| Methods |
| public Token signIn(string username, string password): signs in the user and returns a token representing the session state of the user.<br><br>public void signOut(Token t): signs out the user correlated with the session token, makes the token void.<br><br>public Token signUp(string username, string password, string email): signs up the user using the credentials they give.<br><br>private void updateDatabase(): uses the DatabaseCommunicationManager class to update the database when a new user is created. This method is called inside the signIn method. |

| class SingleExerciseManager |
|---|
| This class is responsible for creating and editing single exercises. |
| Properties |
| |
| Methods |
| public SingleExercise createSingleExercise(User u, Map singleExerciseDetails): creates the requested single exercise for the given user.<br><br>public void editSingleExercise(User u, Map singleExerciseDetails): edits the single exercise of the user. |

| class WorkoutManager |
|---|
| This class is responsible for creating and editing workouts. |
| Properties |

| |
|---|
| |
| Methods |
| public Workout createWorkout(User u, Map workoutDetails): creates the requested workout for the given user.<br><br>public void editWorkout(User u, Map workoutDetails): edits the workout of the user.<br><br>public void addSingleExercise(Workout w, SingleExercise s): adds single exercise to the workout.<br><br>public void removeSingleExercise(Workout w, SingleExercise s): removes single exercise from the workout. |

| class CalendarManager |
|---|
| This class is responsible for creating and editing calendar. |
| Properties |
| |
| Methods |
| public Calendar createCalendar(User u, Map calendarDetails): creates the requested calendar for the given user.<br><br>public void editCalendar(User u, Map calendarDetails): edits the calendar of the user.<br><br>public void addCalendar(Calendar c, Workout w): adds workout to the calendar.<br><br>public void removeCalendar(Calendar c, Workout w): removes workout from the calendar. |

### 3.2.2 Model

| class User |
|---|
| This class is a model data class responsible for holding user data. |
| Properties |

| |
|---|
| private int id<br>private String name<br>private String surname<br>private string gender<br>private int height<br>private String heightMeasurementType<br>private int weight<br>private String weightMeasurementType<br>private int age |
| Methods |
| User() : constructor for this class<br>public int getid(): returns the ID of the user<br>public String getName(): returns the name of the user<br>public void setName(): sets the name of the user<br>public string getGender(): returns the gender of the user<br>public void setGender(): sets the gender of the user<br>public int getHeight(): returns the height of the user<br>public void setHeight(): sets the height of the user<br>public String getHeightMeasurementType(): returns the height measurement type<br>public int getWeight(): returns the weight of the user<br>public void setWeight(): sets the weight of the user<br>public String getWeightMeasurementType(): returns the weight measurement type<br>public int getAge(): returns the age of the user<br>public void setAge(): sets the age of the user |

| class SingleExercise |
|---|
| This class is a model data class responsible for holding data of a single exercise. |
| Properties |
| private String exerciseName |

| |
|---|
| private int setCount |
| Methods |
| SingleExercise(): constructor of this class<br>public void changeSetCount(int a): changes the set count<br>public int getSetCount(): returns the set count<br>public String getExerciseName(): returns the exercise name |

| class Workout |
|---|
| This class is a model data class responsible for holding workout data. A workout consists of several single exercises. |
| Properties |
| private String workoutName<br>private SingleExercise[] exercises |
| Methods |
| Workout(): constructor method for this class<br>public SingleExercise getSingleExercise(String ExerciseName): returns the SingleExercise |

| class Calendar |
|---|
| This class is a model data class, which is responsible for holding the workout schedule information. |
| Properties |
| private Workout[]  completedWorkouts |

| |
|---|
| private Workout[] currentWorkouts |
| Methods |
| Calendar(): constructor method for this class<br>public Workout[] getCompletedWorkouts(): returns the completed workouts<br>public Workout[] getCurrentWorkouts(): returns the completed workouts<br>public void addCompletedWorkout(Workout w): adds the workout to the completed workouts<br>public void addCurrentWorkouts(Workout w): adds the workout to the current workouts<br>public void rmCurrentWorkouts(Workout w): removes the workout from the current workouts |

# 4. Glossary

As the abbreviations are already defined in section 1.4, this section contains the information in which pages these words are being used.

Back-end: 5, 16, 17, 18

Class: 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29

Client: 5, 6, 12, 13, 23

Controller: 5, 6, 7, 11, 13, 20, 23

Method: 4, 5, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29

Model: 4, 5, 12, 13, 24, 26, 27, 28

Property: 13, 22, 23

Server: 6, 8, 11, 23

UI: 6

View: 5, 6, 7, 8, 10, 15, 16, 17, 18, 19, 20

# 5. References

[1] "What Is the Importance of Sports in Our Life?," *Impoff*, 2020. [Online]. Available: https://impoff.com/importance-of-sports/. [Accessed: 12- Oct- 2020].

[2] "Sports and Mental Health | Newport Academy," *Newport Academy*, 2020. [Online]. Available: https://www.newportacademy.com/resources/mental-health/sports-and-mental-health/. [Accessed: 12- Oct- 2020].

[3] "Top 6 benefits of sports and physical activity on mental health – Sport Energy," *Sportenergy.club*, 2020. [Online]. Available: https://sportenergy.club/2020/02/21/top-6-benefits-of-sports-and-physical-activity-on-mental-health/. [Accessed: 12- Oct- 2020].

[4] M. Weber, "Exercise During Coronavirus: Tips for Staying Active - HelpGuide.org," *Helpguide.org*, 2020. [Online]. Available: https://www.helpguide.org/articles/healthy-living/exercise-during-coronavirus.htm. [Accessed: 12- Oct- 2020].

[5] H. Publishing, "10 tips to prevent injuries when you exercise - Harvard Health," *Harvard Health*, 2020. [Online]. Available: https://www.health.harvard.edu/pain/10-tips-to-prevent-injuries-when-you-exercise. [Accessed: 12- Oct- 2020].

[6] "1016-1987 - IEEE Recommended Practice for Software Design Descriptions," *IEEE Xplore*. [Online]. Available: https://ieeexplore.ieee.org/document/565312. [Accessed: 07-Feb-2021].

[7] *Opennetworking.org*, 2021. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/UML_Modeling_Guidelines_V1.0.pdf. [Accessed: 07- Feb- 2021].

[8] Amit, "All You Need to Know About UML Diagrams: Types and 5+ Examples," *Tallyfy*, 26-Feb-2020. [Online]. Available: https://tallyfy.com/uml-diagram/#what_is_the_use_of_uml. [Accessed: 07-Feb-2021].

[9] "Front-End vs Back-End vs Full Stack Web Developers," *Udacity*, 10-Jan-2020. [Online]. Available: https://blog.udacity.com/2014/12/front-end-vs-back-end-vs-full-stack-web-developers .html. [Accessed: 27-Dec-2020].

[10] "Software Engineering: Object Oriented Design - javatpoint," *www.javatpoint.com*. [Online]. Available: https://www.javatpoint.com/software-engineering-object-oriented-design. [Accessed: 08-Feb-2021].

[11] "Client-Server Model," *GeeksforGeeks*, 15-Nov-2019. [Online]. Available: https://www.geeksforgeeks.org/client-server-model/. [Accessed: 08-Feb-2021].

[12] "MVC," *MVC (Model-View-Controller) Definition*. [Online]. Available: https://techterms.com/definition/mvc#:~:text=Stands%20for%20%22Model%2DVie w%2D,for%20developing%20modern%20user%20interfaces. [Accessed: 08-Feb-2021].

[13] *Study.com*. [Online]. Available: https://study.com/academy/lesson/oop-object-oriented-programming-objects-classes-i nterfaces.html. [Accessed: 08-Feb-2021].

[14] *Java Fundamentals Tutorial: Object Oriented Programming in Java*. [Online]. Available: https://www.protechtraining.com/content/java_fundamentals_tutorial-object_oriented. [Accessed: 08-Feb-2021].

[15] T. T. Contributors, "What is User Interface (UI)?," *SearchAppArchitecture*, 13-Aug-2019. [Online]. Available: https://searchapparchitecture.techtarget.com/definition/user-interface-UI. [Accessed: 27-Dec-2020].