



Bilkent University

Department of Computer Engineering

CS 491 - Senior Design Project

Project Analysis Report

Project Name: *Deeplay*

Team: *Ludens*

F. Serdar Atalay - 21300738

Ekinsu Bozdağ - 21604089

Gökcan Değirmenci - 21401658

Gökçe Özkan - 21402188

Onur Sönmez - 21300749

Supervisor

Halil Altay Güvenir

Jury Members

Shervin R. Arashloo

Uğur Güdükbay

Innovation Expert

Mustafa Sakalsız

November 12, 2018

Introduction	2
Current Systems	2
Proposed System	4
3.1 Overview	4
3.2 Functional Requirements	4
3.2.1 User Functionality Requirements	4
3.2.2 System Functionality Requirements	5
3.3 Non-functional Requirements	5
3.3.1 Usability	5
3.3.2 Performance	5
3.3.3 Extensibility	6
3.3.4 Security	6
3.3.5 Scalability	6
3.3.6 Robustness	6
3.4 Pseudo Requirements	6
System Model	7
4.1 Use Case Model	7
4.1.1 Use Case Model For Deeplay Platform	7
4.1.2 Use Case Model For Mobile Game	13
4.2 Dynamic Model	18
4.2.1 Sequence Diagrams	18
4.2.1.1 Deeplay User Authentication Sequence Diagram	18
4.2.1.2 Deeplay Data Labeling Sequence Diagram	20
4.2.1.3 Deeplay Multiplayer Game Sequence Diagram	22
4.2.2 Activity Diagram	23
4.3 Class Model	24
User Interface	26
References	32

1. Introduction

In the field of computer science, there are many companies, engineers and data scientists that extensively use deep learning. As the amount of necessary and appropriate data to be used during training increases, the performance increases with deep learning algorithms[1]. With the current advancements on machine learning, especially on its subset area of deep learning, intelligent applications become extremely data-hungry. The need for uncompromised quality, actionable data increases everyday. That is to say, data becomes the “new oil” of our era. People label their data to do training and they need more and more data to get the most accurate results for their systems. Labelling data does not require high level education in the field. Thus, when requirements are well understood, any person with adequate hand-eye coordination can label training data. By bringing crowdsourcing and gamification concepts, our project *Deeplay* promises entertainment to people, while making them help those who are interested in or working on deep learning. *Deeplay* provides a game platform in which there will be different categories of games that mainly aims to make users label data.

2. Current Systems

There are many systems that use crowdsourcing for data labeling. The biggest examples include Google’s Crowdsourcing and Amazon’s Mechanical Turk. Google’s Crowdsourcing does not offer any fee to the contributors, but the users can level up and open badges as they label more data. The Mechanical Turk, however, offers micro payments for labeling, but they limit the amount of users in order to keep the process and the users verifiable and accountable.

Also, we all know that Google uses reCAPTCHA to collect their labeled data by using the slogan “protects your websites from spam and abuse”. Actually, all tech giants like Google, Amazon crowdsource their training data and dominate the marketplace. The ESP Game and Google Image labeler are also the great examples

of crowdsourcing.[4] Each gamified app pairs up anonymous partners with each other for a round of images and label images according to the consensus taken from the users. What they do is actually using human-based computation to solve crowdsourcing problems. Another example is hCaptcha; which is a drop-in replacement for reCaptcha that earns website owners money and help companies get their data labeled.

We intent to gamify the labeling process in order to keep users playing and get entertained in a fun way while labeling our data. Briefly, what makes our project innovative and different from the existing similar projects is that it brings game developers and people who demand labelled data in one domain. It does not serve data for specific companies, but any demander will be able to get their data labelled by using the games in our platform. Giant companies do their crowdsourcing own and dominate the market. We are aiming to provide a platform in which every company may do their own crowdsourcing and do not need to integrate the system into their own platforms just like reCaptcha, hCaptcha. Only requirement for labeled data demanders is to use our existing platform. End-users will be able to select different games according to their taste from the platform and while playing the game, they will be labelling data for the demanders.

3. Proposed System

3.1 Overview

Deeplay is an intelligent data labeling platform which combines machine learning, gamification and human-in-the-loop approaches to provide easy access to actionable data. With the current advancements on machine learning, especially on its subset area of deep learning, software applications become extremely data-hungry. The need for uncompromised quality, actionable data increases every day. That is to say, data becomes the “new oil” of our era. To achieve the most accurate results, our human-powered data labeling system ensures high-quality with machine-driven checks and crowdsourcing algorithms.

3.2 Functional Requirements

3.2.1 User Functionality Requirements

- Any user or foundation who wants to label their AI/ML training data may use the platform as an **end-to-end pipeline for their “data labeling” needs**.
- Labeled data demanders may select the game they want to label according to the category of the game.
- Labeled data demanders and independent game developers may use the blockchain to create smart contracts between each other. Demander may determine the task and monetize the efforts of game developers according to the successful completion of the task within the specified deadline.
- Labeled data demanders may create a labeling task and determine the data annotation constraints including accuracy rate, label count for each data, etc.
- Game developers may create their annotation-based games on the platform by using public Deeplay API.

- Multiple users may play the game in real-time multiplayer game session so that the accuracy and reliability of the labeling would increase.
- End-user may play any game and label random data(e.g. Image segmentation) on the platform. Our system will use human computation for crowdsourcing.

3.2.2 System Functionality Requirements

- The system needs to cross-validate crowdsourced data set of responses according to predefined consensus rules, machine-driven checks and crowdsourcing algorithms.
- The system will monetize the efforts of the game developers according to the value they created.
- The system should support creation of Deeplay Ads that are placed by developers.

3.3 Non-functional Requirements

3.3.1 Usability

- The heart of the platform consists different types of addictive games that made by thinking about the end-user first. So it means that the users should be able to use the platform easily and intuitively.
- Plot and play instructions of the games should be easily understandable.

3.3.2 Performance

- The application should be available to its users 24/7/365 with at most 0.5% overall downtime statistics (99.5% availability).
- Filtration and preprocessing steps of the batch labeling tasks should be fast enough to keep up with the deadlines given by the clients.

3.3.3 Extensibility

- The system should support easy integration for the games. Thus, more games may be integrated into the system in order to increase the number of options for the data labeling. For example, in the future, there might be human-labeled sound clips, or other special-kind of sensitive data which needs to be labeled with the help of gamification.
- The architecture and implementation actively caters to future business needs.

3.3.4 Security

- The system should ensure end-to-end security of sensitive data of users and private information of companies.

3.3.5 Scalability

- The system should be scalable enough to handle the growing amount of work and implement the sharding as a way of distributing data across sets of multiple machines.

3.3.6 Robustness

- The system should be robust. Sudden traffic increase should not be an issue to the system.

3.4 Pseudo Requirements

- To develop the web and mobile app, mainly JavaScript ecosystem and technologies such as React.js, React Native(with needed native plugins) and Node.js will be used.
- To develop machine learning system that powers our *quality assurance engine*, Python and its data science ecosystem technologies such as PySpark, Keras will be used.
- To ensure best practices among our data and REST API services, *microservices* architecture will be designed and used.

- To ensure security, reliability and portability of our platform, application services will be containerized with Docker.
- To provide best-in-class UI/UX design for our platform, various design & prototyping tools such as Adobe XD and Adobe Illustrator will be used.

4. System Model

4.1 Use Case Model

4.1.1 Use Case Model For Deeplay Platform

Scenario 1

Use case name: Sign Up

Participating actor: Data Demander, Game Developer, Player

Entry condition:

The user enters to [Deeplay.io](https://deeplay.io)

Main flow of events:

1. User clicks Sign Up button on landing page.
2. User selects an signup method
 - a. Sign up with Google
 - i. Authorize Deeplay with Google
 - b. Sign up with GitHub
 - i. Authorize Deeplay with Github
 - c. Or, Sign up with user credentials(Regular)
 - i. User enters the credentials including password, email and username.

3. The system validates the authority of user and give necessary permissions.

Exit condition:

The user is directed to the dashboard after successful sign up.

Scenario 2

Use case name: Log In

Participating actor: Data Demander, Game Developer, Player

Entry condition:

User has a Deeplay account.

Main flow of events:

1. User clicks Login button on landing page.
2. User selects authorization method
 - a. Login with Google
 - i. Authorize Deeplay with Google
 - b. Login with Github
 - i. Authorize Deeplay with Github
 - c. Login with user credentials(Regular)
 - i. User enters the credentials including password, email(or username).
3. The system validates the authority of user and give necessary permissions.

Exit condition:

The user is directed to the dashboard after successful login.

Alternative flow of events:

1. User forgets his/her password.
2. User selects "forgot password" option.
3. The system sends a verification code to the email address.

4. User is directed to change password screen.
5. User resets the password using the verification code.

Scenario 3

Use case name: Select Category

Participating actor: Player

Entry condition:

The player has opened the “deeplay” platform.

Main flow of events:

1. The system display different categories on the screen.
2. Player views different categories.
3. Player clicks the option s/he want to play.

Exit condition:

Player sees different games related to the selected category on the screen.

Alternative flow of events:

Player exits “deeplay” platform.

Scenario 4

Use case name: Select Game

Participating actor: Player

Entry condition:

Player is directed to the list of the games related to the chosen category

Main flow of events:

1. Player view different game options
2. Player clicks one of the games
3. There is a brief description of the game displayed on the screen
4. Player starts the game

Exit condition:

Player is directed to the selected game screen.

Alternative flow of events:

1. Player may not be satisfied with the games under the current category.
2. Player selects a different category.

Scenario 5

Use case name: Create Label Task

Participating actor: Data Demander

Entry condition:

Data demander must be registered and signed in.

Main flow of events:

1. Data demander has some data that needs to be labeled.
2. Data demander selects the annotation type and game to create labeling tasks.
 - a. The system displays all games and their descriptions on the screen.
 - b. The system displays different types of annotations.
3. Data demander uploads the dataset that will be labeled.
 - a. He/she selects the desired data from his/her local machine.
 - b. Or, the user may drag the files into the file upload section
4. Data demander specifies the constraints for the labeled data such as
 - a. Selecting the accuracy constraint.
 - i. Accuracy rate of labeled data.

b. Selecting the view constraint.

- i. How many times one data should be labeled by a player.

Exit condition:

Data demander confirms all the annotation rules and creates data labeling task by clicking button on the dashboard.

Scenario 6

Use case name: View Labeled Data

Participating actor: Data Demander

Entry condition:

Data demander has uploaded some data to the platform.

Main flow of events:

1. Data demander opens admin dashboard.
2. Data demander tracks the status of submitted data and his/her past activities.
 - a. Completed annotations.
 - b. Pending annotations.
 - c. In-progress annotations.

Exit condition:

Data demander confirms the labeling task and downloads labeled dataset when the task is finished.

Alternative flow of events:

Data demander cancels the annotation before the task finished.

Scenario 7

Use case name: Integrate Game

Participating actor: Game Developer

Entry condition:

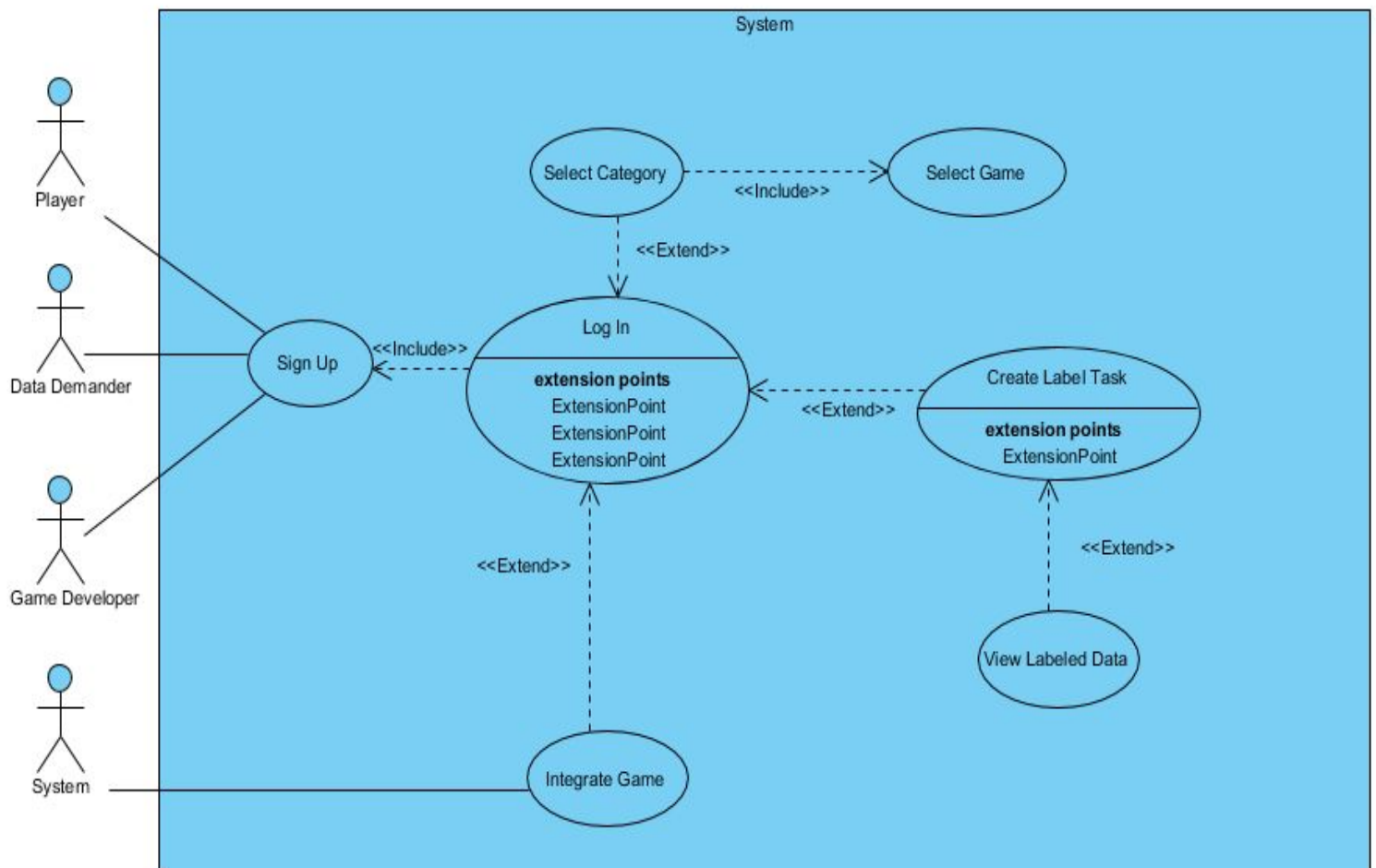
The user must be registered into Deeplay.io.

Main flow of events:

1. Registered user generates personal API access token on user settings section.
2. User opens deeplay.io/docs and read the documentation to learn how to use public API.
3. API docs will have instructions to show how new games will be integrated into Deeplay platform.
 - a. Like how to use Deeplay websocket API for adding annotation-based multiplayer games.
4. Developer needs to use the public api with access token to integrate it into their system.

Exit condition:

After integration with public API, the game will be added into the deeplay mobile app which contains the variety of published games.



4.1.2 Use Case Model For Mobile Game

Scenario 8

Use case name: Play Game

Participating actor: Player

Entry condition:

Player selected a game from the platform

Main flow of events:

1. The game has 2 options.

2. User decides whether to play as one player or multiplayer.
3. User selects one of the options.

Exit condition:

Player is directed to the game and game starts.

Alternative flow of events:

User goes back to the menu.

Scenario 9

Use case name: Play 1 Player

Participating actor: Player

Entry condition:

User selects 1 player option.

Main flow of events:

1. As the game starts, timing starts to count down.
2. Player label images during this period.
3. Player submits his/her answer or result and goes to the next image.

Exit condition:

- User pauses the game.
- Time is up.

Scenario 10

Use case name: Pause

Participating actor: Player

Entry condition:

1 Player option is played by the user.

Main flow of events:

1. User presses pause button.
2. Game stops.
3. Exit and resume options are displayed.

Exit condition:

- Player is directed to the main menu.
- Player resumes the game.

Scenario 11

Use case name: Play Multiplayer

Participating actor: Player

Entry condition:

Player selects multiplayer option from play game use case.

Main flow of events:

1. Player is paired with another user who wants to play the game as multiplayer at the same time.
2. Both player labels the images coming to the screen.
3. The players do not see the label of each other.
4. Players have 3 rights to label different objects. When one of the player labels wrongly, they both lose one right.
5. As both of the players label approximately the same thing, next image comes.

Exit condition:

- Players labels differently more than 3 times and game is over.
- The player exits the game and return to main menu.

Scenario 12

Use case name: Exit

Participating actor: Player

Entry condition:

User plays multiplayer game option.

Main flow of events:

1. User wants to stop the game.
2. User presses exit button.

Exit condition:

- User is directed to the main menu.

Scenario 13

Use case name: View Profile

Participating actor: Player

Entry condition:

User enters a game.

Main flow of events:

1. User can see how much deepoint s/he gained in total
2. Player can see the scores.
3. Player can explore new games.
4. Player can see how many hours s/he played the game up to that point.
5. Player gets notification when there are online players for multiplayer game.

Exit condition:

Player starts the game.

Alternative Flow of Events:

Player changes the settings.

Scenario 14

Use case name: Change Settings

Participating actor: Player

Entry condition:

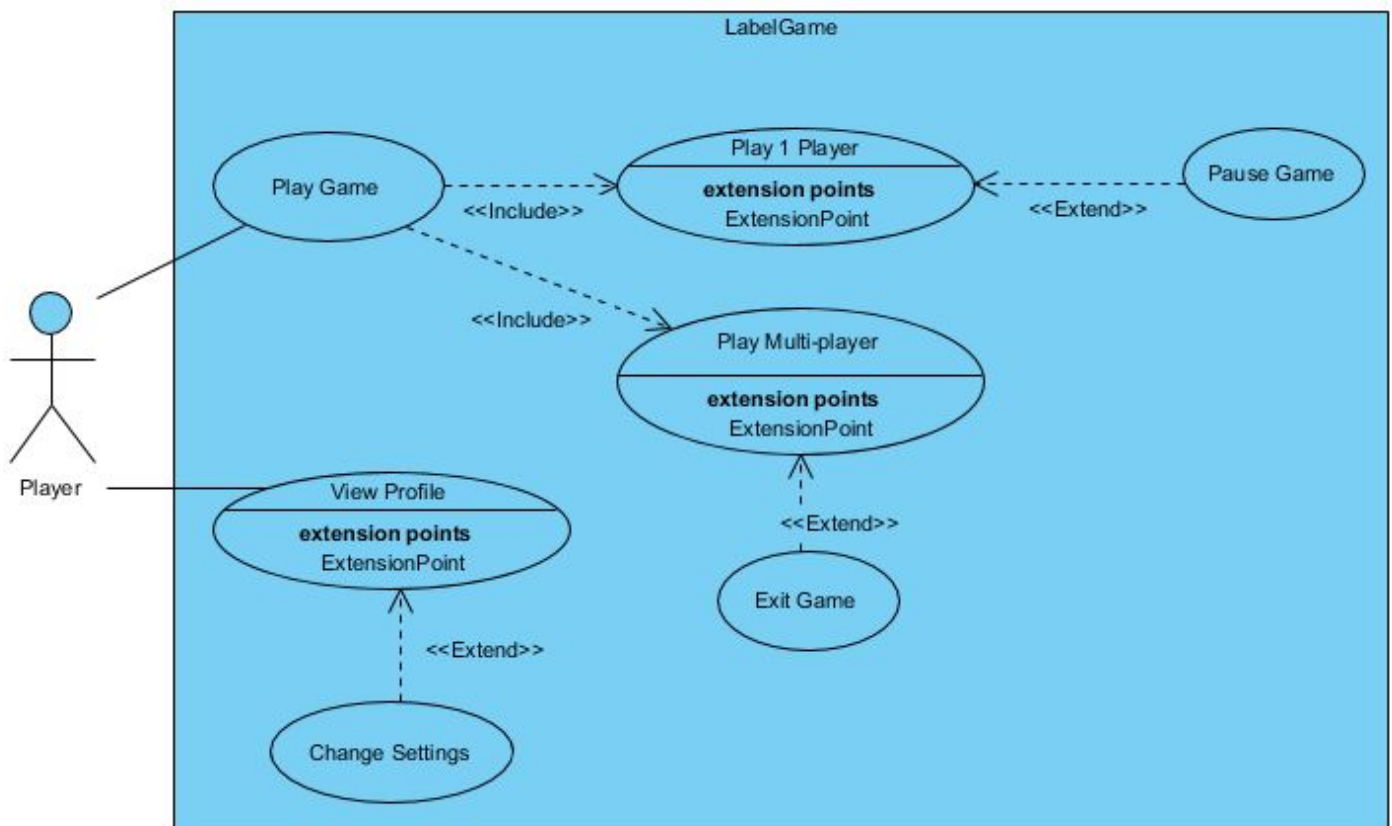
Player enters the profile page.

Main flow of events:

1. Player can turn on/off the notifications.
2. Player can turn on/off background music

Exit condition:

Player goes back to profile page.

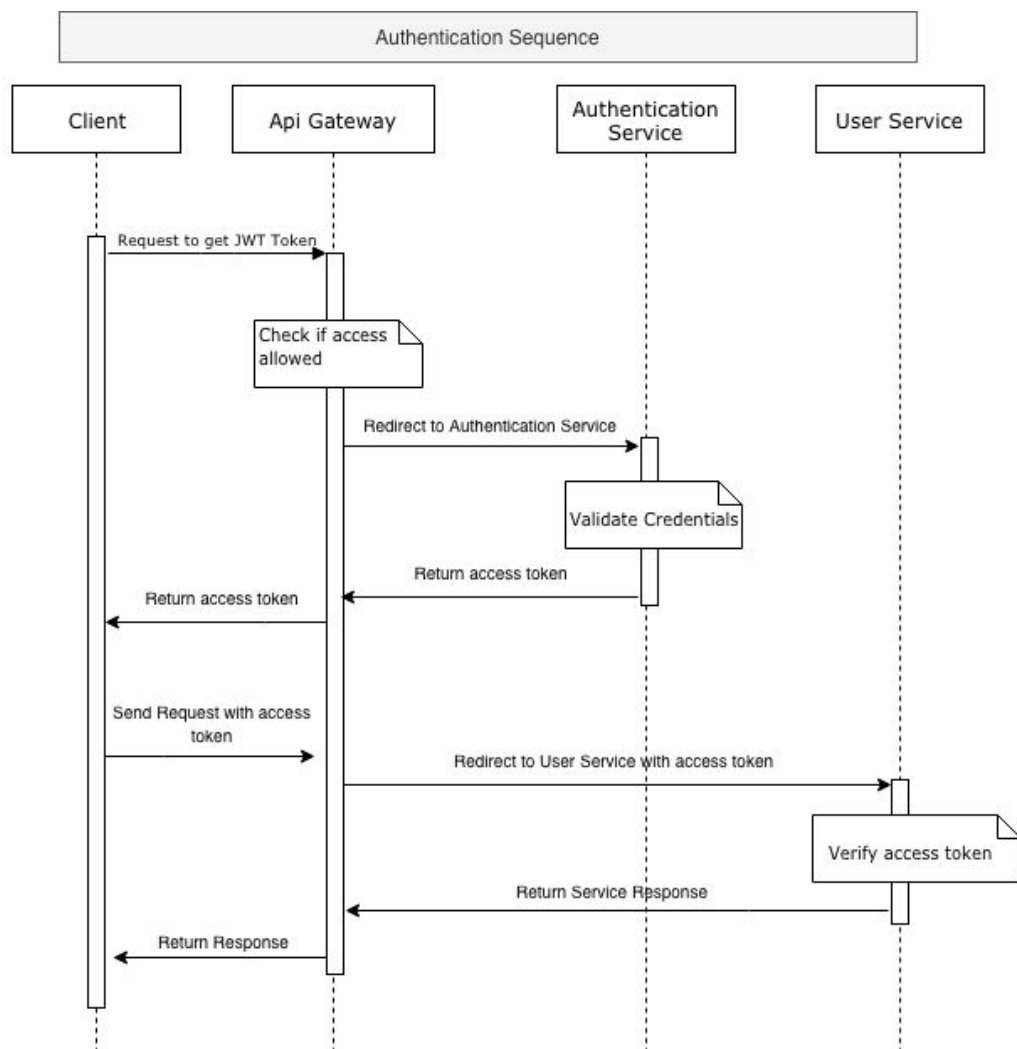


4.2 Dynamic Model

4.2.1 Sequence Diagrams

This section shows the sequence of events that external actors generate. Diagrams under this section mostly describe the inter-system events of Deeplay.io instead of focusing only use-case(UI) related events. Deeplay uses microservice architecture for all their services and thereby interaction between microservices will be shown with the help of sequence diagrams. Sequence diagrams were drawn by referencing IBM UML Basics[5].

4.2.1.1 Deeplay User Authentication Sequence Diagram

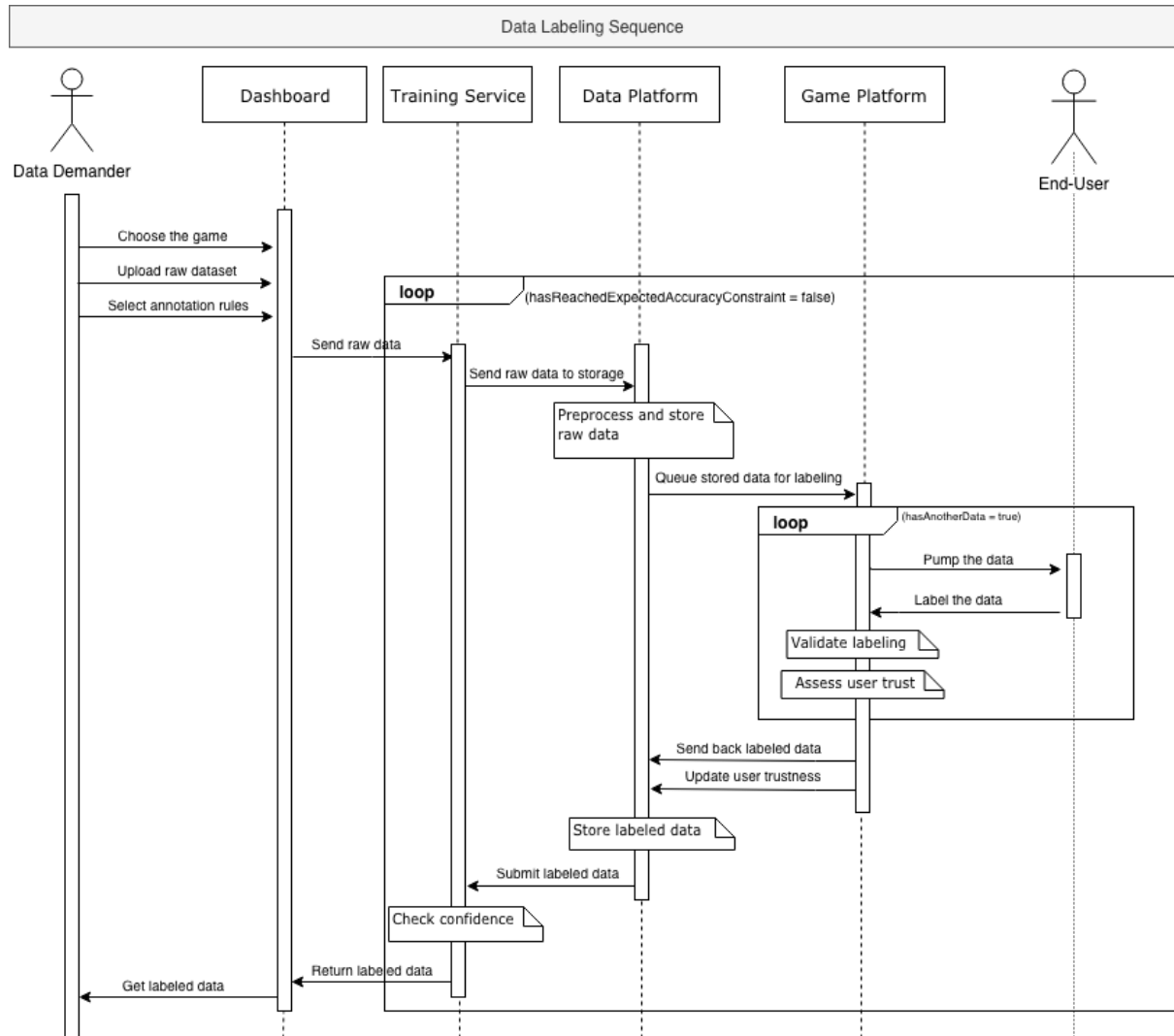


Deeplay will use JWT-based(JSON Web Token) “stateless” authentication to secure the services. Each client action like accessing the labeled dataset, using admin dashboard, tracking the status of queued data will be only allowed for the authenticated user.

Api Gateway will be the entry point for the application programming interface(API). All requests will be distributed to defined group of microservices by using Api Gateway[6].

Authentication Sequence Diagram shows how the services send responses to the requests with the granted authorization. *User Service* might be the any service that require authorization.

4.2.1.2 Deeplay Data Labeling Sequence Diagram



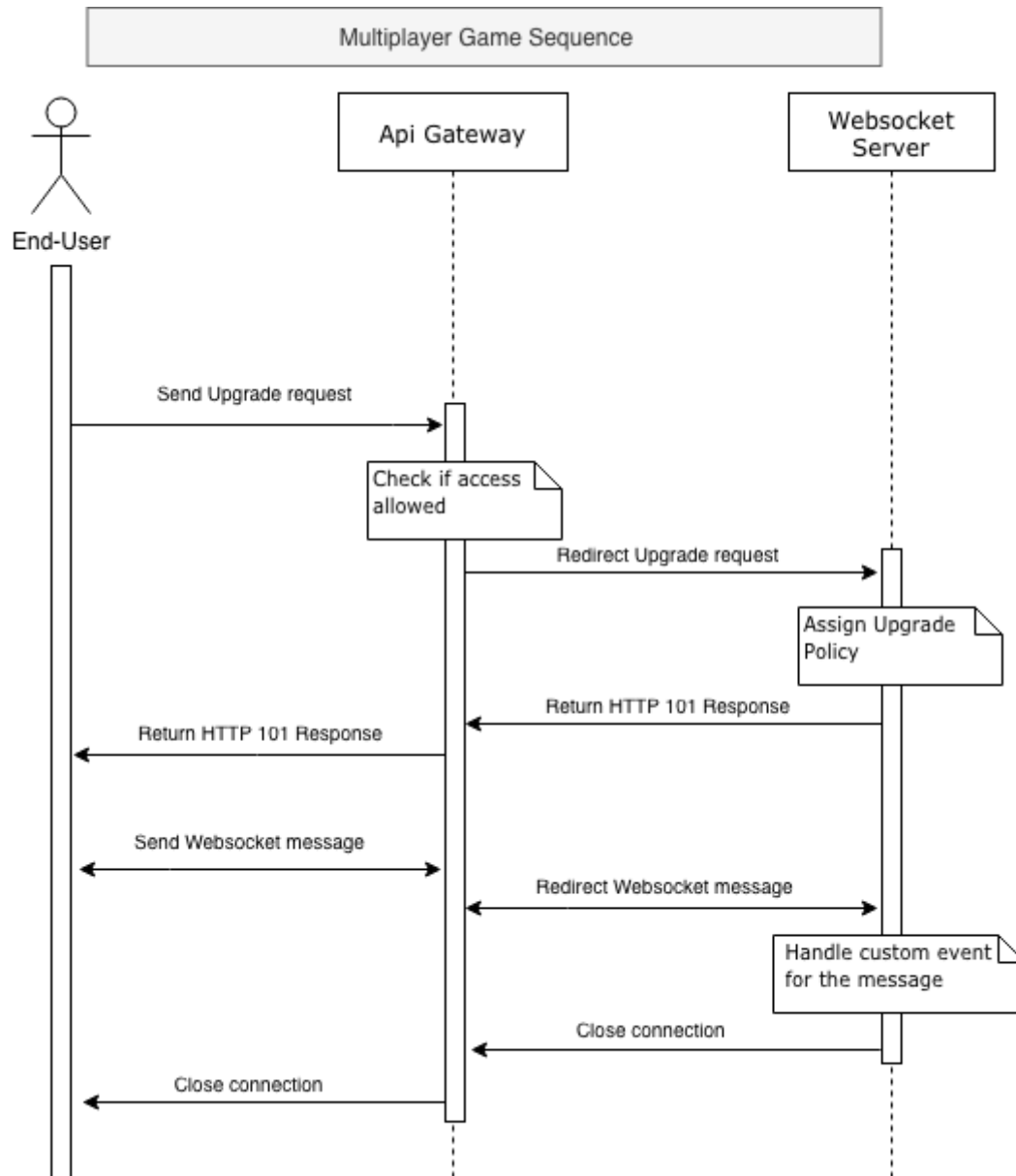
This diagram shows how Deeplay platform creates a stream for the data labeling tasks. Our system combines machine learning, gamification and human-in-the-loop approaches to generate accurate results. After data demander creates a labeling task on the dashboard, raw data will be sent to our *Data Pipeline* which has an automated process including following steps:

- Distributing and storing raw data on our data platform, most probably in cloud storage.
- Preprocessing and training raw data with our initial dataset to make accuracy checks on the game. Because users will collect points according to the their

labels and we need initial results to compare user accuracy and rewarding them accordingly.

- Submitting the data to our game platform and preparing crowdsourced data.
 - **For multiplayer games**, assign same data for paired users and label them according to the consensus taken from users. The intersections of their drawings will be taken as a label if intersections of these plains are large enough(with some constraint).
 - **For single player games**, our system will submit some mixed data that include pre-labeled data which is acquired from our initial dataset and unlabeled data. With this mechanism, we may measure the user reliability for the data annotation.
- Getting labeling results from the game and measuring user trust for the completed tasks.
- At the end, our system will make confidence check for the labeled data using Machine learning techniques.
- This process will continue until the produced data meets the demands for the client. For example, user may specify accuracy constraint such as 0.66 while submitting the data. If the produced results do not reach this threshold, we will repeat and reproduce this labeling task until it reaches expectations.

4.2.1.3 Deeplay Multiplayer Game Sequence Diagram

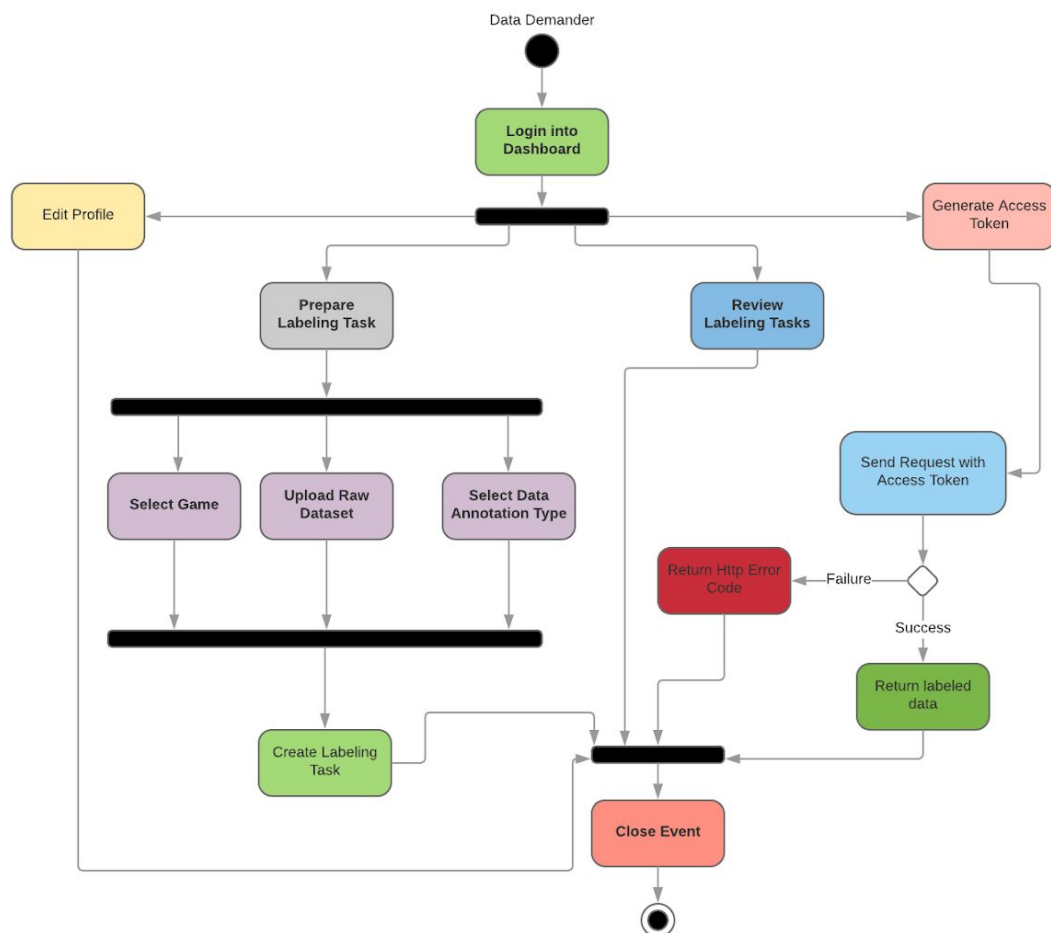


This diagram shows the architecture of multiplayer game server. We will use websocket protocol for creating a fast two-way(bidirectional) communication channel between the user and server. Websockets help to create real-time multiplayer engine with low latency connection mechanism[7]. After the client connects to the socket and sends upgrade request to the server, custom Websocket messages will be handled as described on above diagram.

4.2.2 Activity Diagram

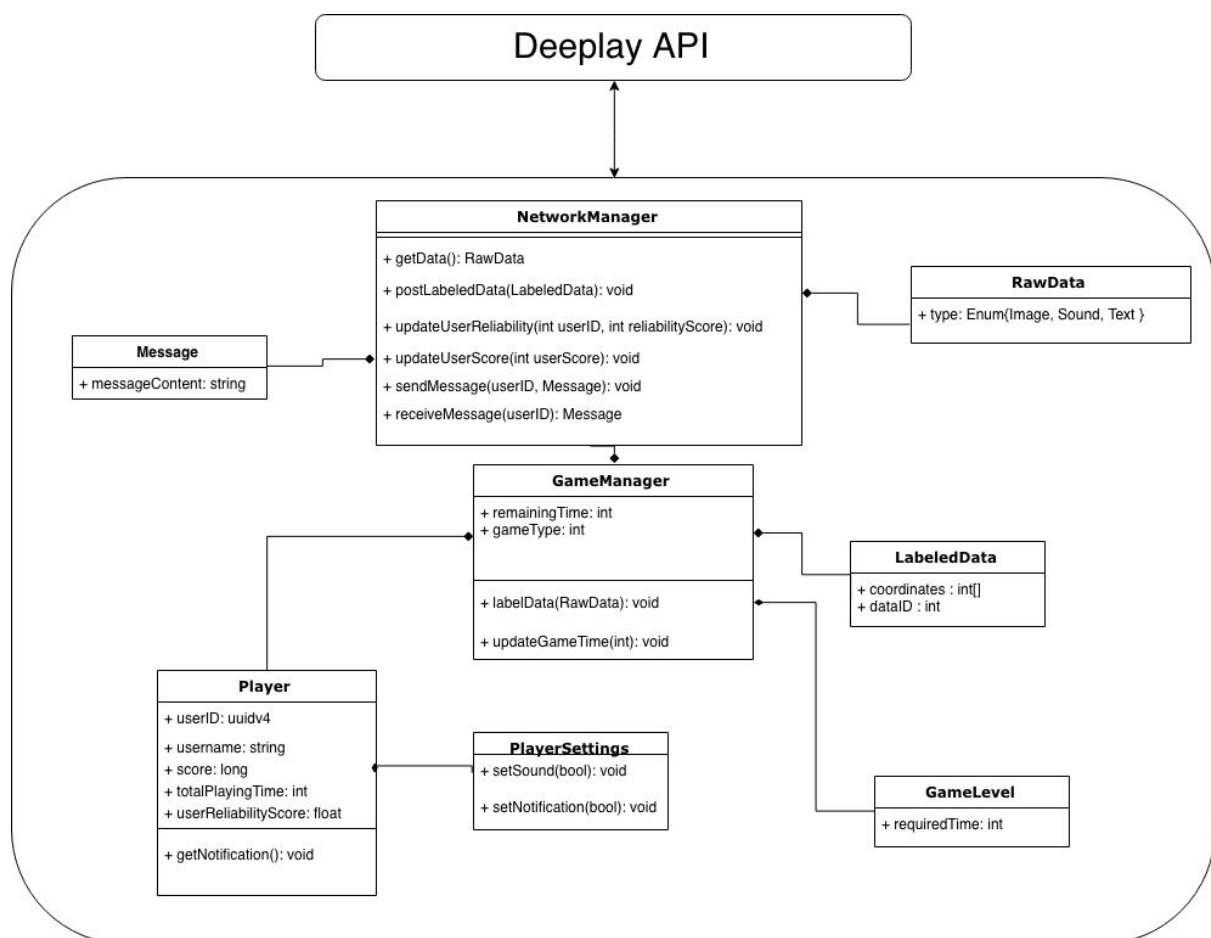
Activity diagram of the registered data demander user after accessing the dashboard is as follows.

USER DASHBOARD ACTIVITY DIAGRAM



4.3 Class Model

Class model for our one of boundary games is as follows. Deeplay is mostly network-based platform and we use microservices architecture for the rest of our system. Procedural or functional programming languages such as JavaScript, Golang, Rust will be used instead of OOP-based ones. That is why rest of system will be shown with dynamic models instead of class modelling.



- Every player will have specific userID and username. The score, totalPlayingTime and userReliabilityScore will be recorded according to the

performance of his/her play. User reliability will be determined by the system as the player plays correctly or label data wrongly.

Player will getNotification when there are people willing to play multiplayer game at that moment.

The user will be able to change basic settings. In PlayerSettings class, there will be two main functions, setSound and setNotification. With those, player will be able to turn on and off the music. Also, s/he will be able to arrange notification preference.

- In GameManager, the type of the currently selected game will be recorded (1 player or multiplayer). According to the requiredTime of the current level, which is specified in GameLevel class, the remainingTime during the game will be updated with updateGameTime function. Game manager will manage how the users label the data. It will both enable access to the network by using NetworkManager. Also, the game mechanism will be controlled by this class.
- NetworkManager is an interface to access our network and use microservices to make user actions. For example, it will access Websocket Server to send and receive messages for real-time multiplayer game. So it will have several functions which are mapped to basic HTTP requests(GET, UPDATE, DELETE, POST) for our resources.

5. User Interface

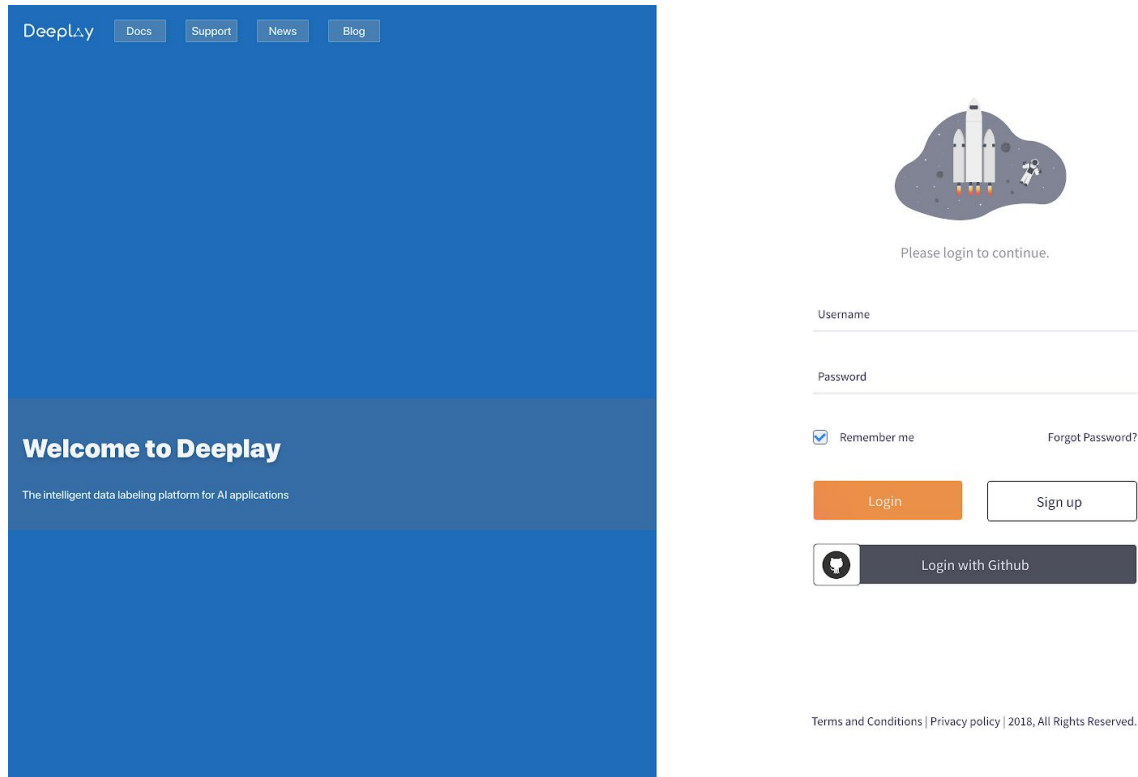


Figure 1 - Login/Signup Page

- The first page user (client) encounters when she enters the **app.deeplay.io** is the login/signup page. He/she can get more info about the app by clicking one of the links on the top-right corner of the page ('Docs', 'Support', 'News' or 'Blog'). We have used clean and simple UI design paradigms to align items in this page. The *“Call to Action”* UI element in this view is the orange “Login” button.

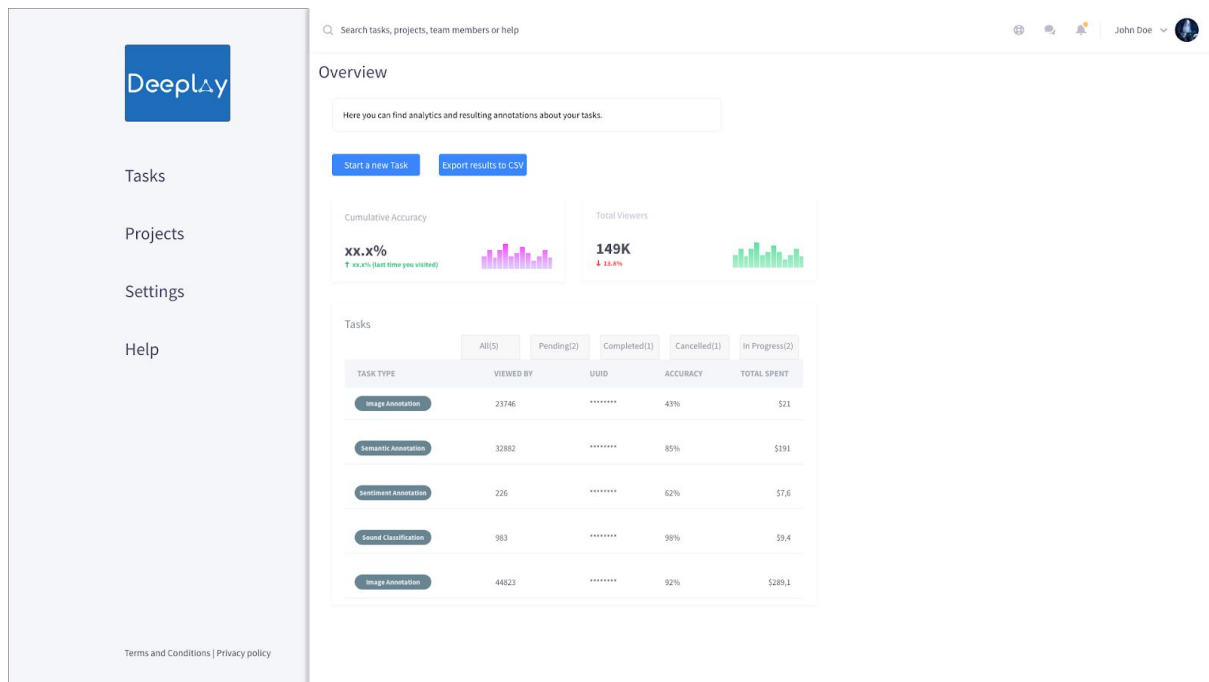


Figure 2 - Overview Page (Tasks + Analytics)

- The “Overview” page renders after user is successfully authenticated. This page shows analytics about the Data Annotation/Classification tasks user has created in the platform. The user can easily start a new Task or Export results from the completed Tasks. On the other hand, we have a sticky Search and AppBar on top of the Overview section that shows the user details, notifications(e.g. “Task 3 has been completed!”) and support links in a compact form.

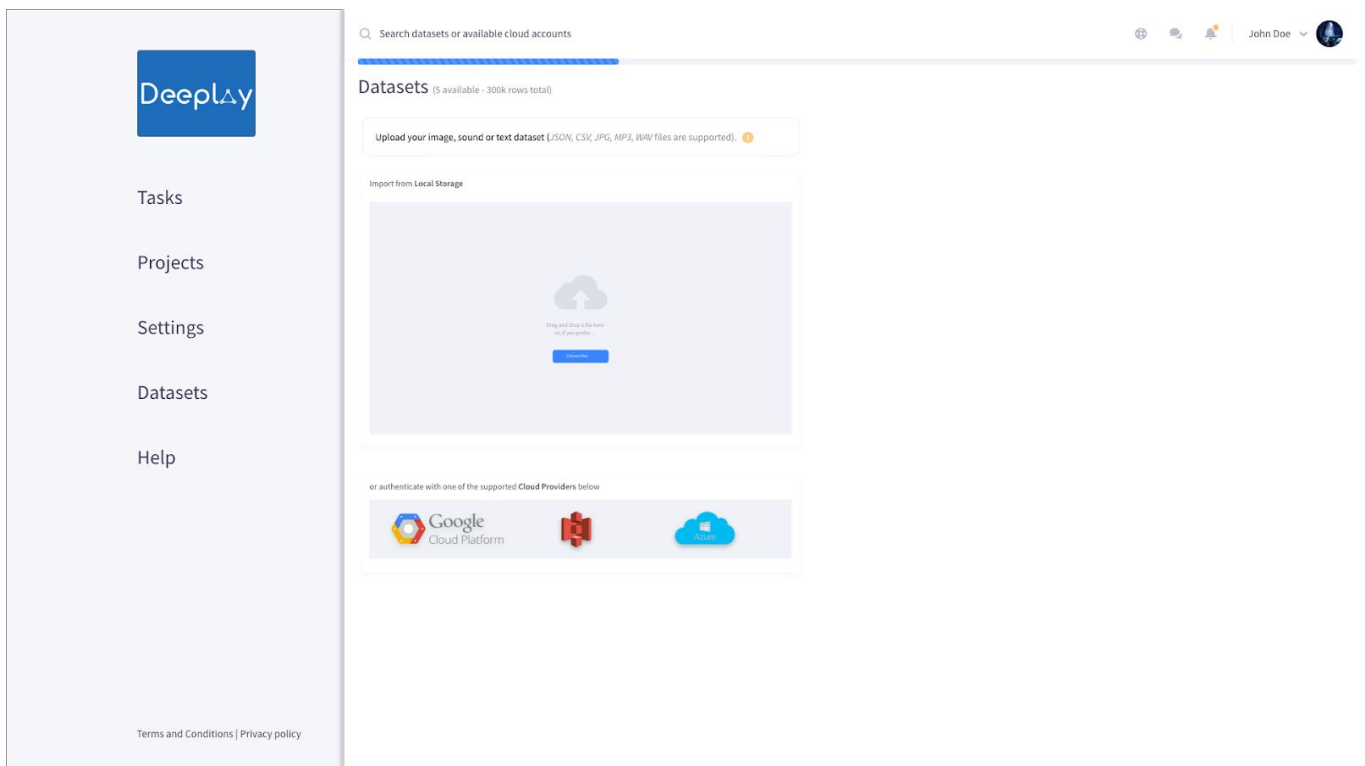
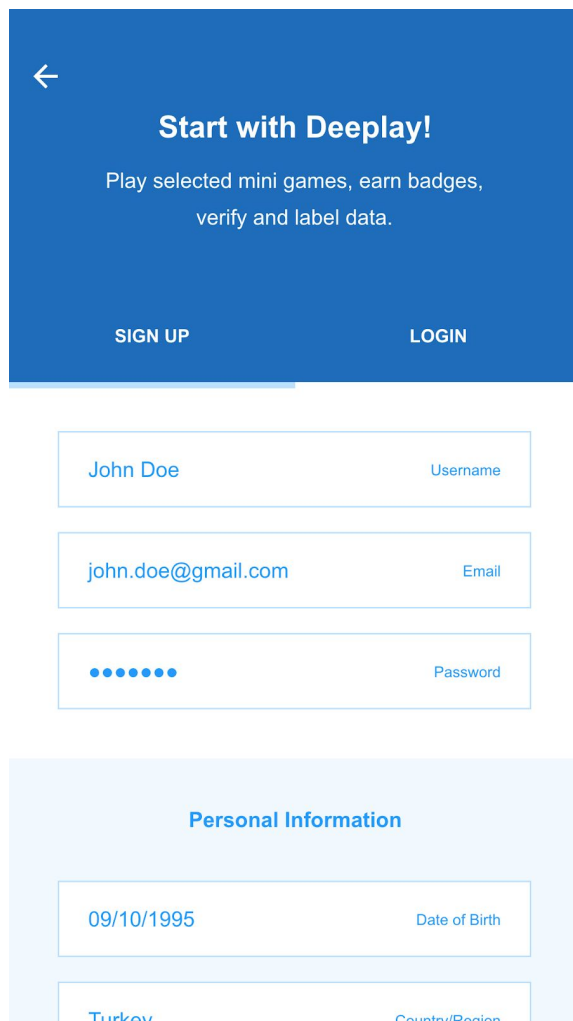


Figure 3 - Dataset Upload/View Page

- In “Datasets” tab, we have Upload and View sections. When uploading files, the user may choose from *local storage* or connect to one of the supported cloud providers. On top of the page, we have a loading indicator that shows the current status of the upload processes.



The screen is divided into two main sections. The top section has a dark blue background with a white back arrow in the top left corner. It features the text "Start with Deeplay!" in white, followed by "Play selected mini games, earn badges, verify and label data." in a smaller white font. At the bottom of this section are two white buttons: "SIGN UP" and "LOGIN". The bottom section has a light blue background and contains three input fields for login: "Username" (with "John Doe" entered), "Email" (with "john.doe@gmail.com" entered), and "Password" (with six dots entered). Below these is a "Personal Information" section with two more input fields: "Date of Birth" (with "09/10/1995" entered) and "Country/Region" (with "Turkey" entered).

Figure 4 - Mobile Login/Sign up

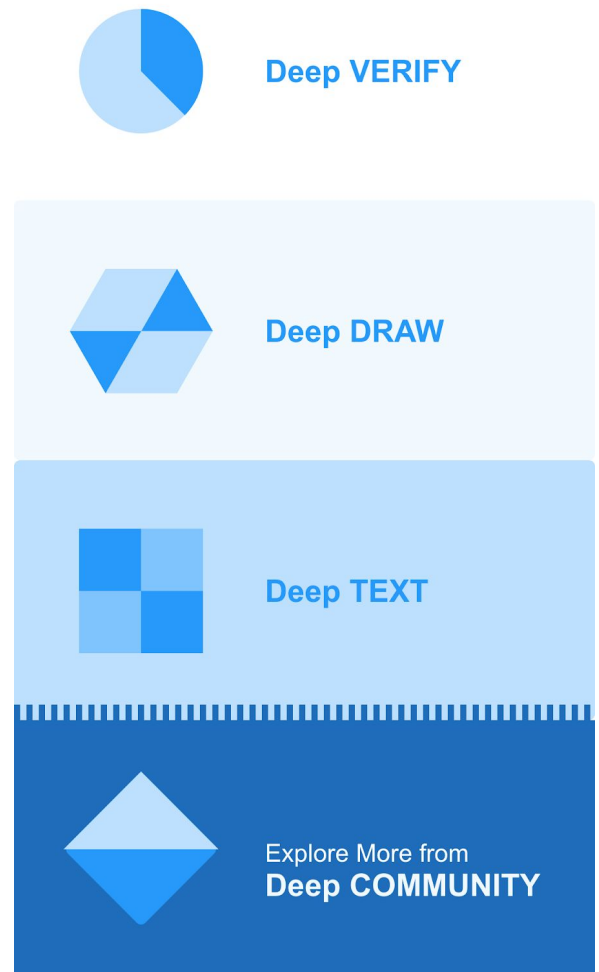


Figure 5 - Mobile Home Page

- The user (game player), after successfully authenticated, is redirected to the Home Page. Our home page displays available games in a stacked list format.

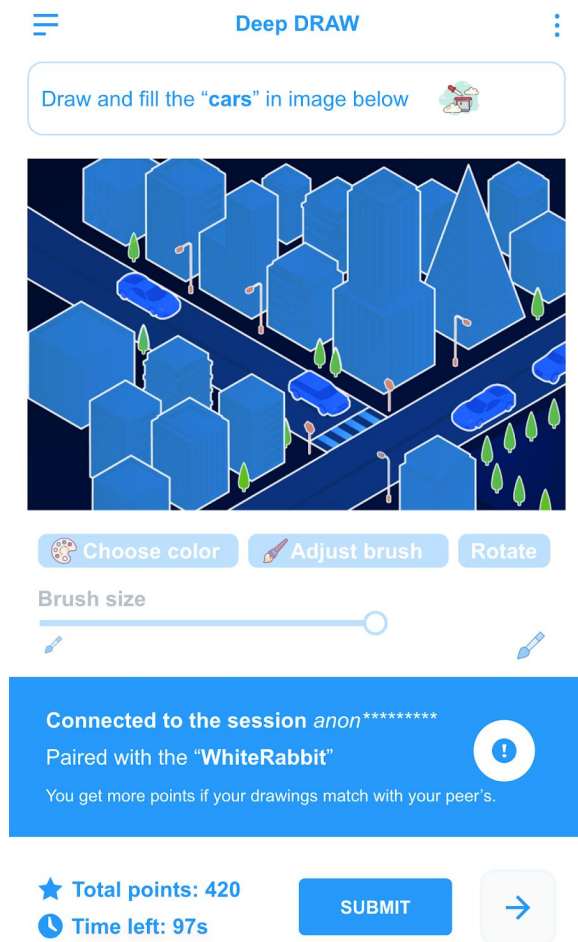


Figure 6 - Mobile Game - Draw

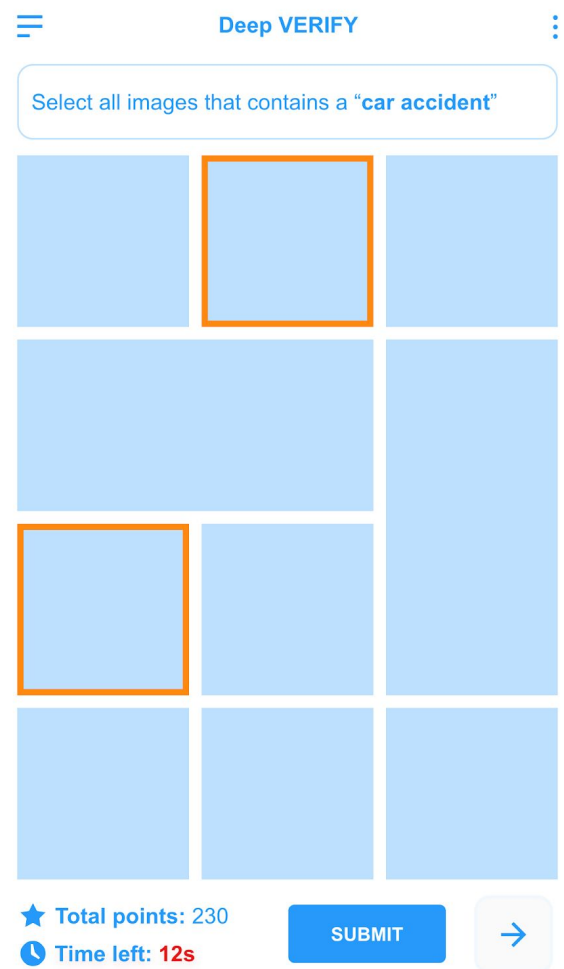


Figure 7 - Mobile Game - Verify

- These are the early concept UI drawings for our mobile games that showcases Data Labeling, Gamification(contest + points + badges) and Multiplayer features.

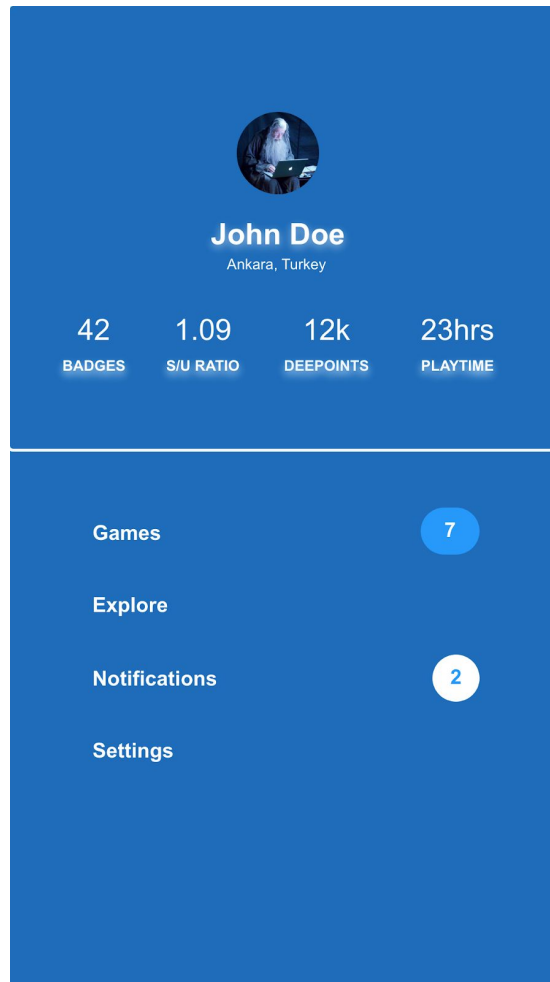


Figure 8 - User Profile

- In profile page, the user may see his total points, badges, total playtime and Successful/Unsuccessful ratio. Also, he can go back to Games or Explore new games, take a look at the notifications and tweak the Settings if needed.

6. References

- [1] J. Brownlee, "What is Deep Learning?", *Machine Learning Mastery*, 2018.
[Online]. Available: <https://machinelearningmastery.com/what-is-deep-learning/>
- [2] "Open Images Dataset V4", *Storage.googleapis.com*, 2018. [Online]. Available:
<https://storage.googleapis.com/openimages/web/index.html>
- [3] Code of Ethics National Society of Professional Engineers, nspe.org, 2016.
<https://www.nspe.org/resources/ethics/code-ethics/>.
- [4] Rethinking the ESP Game, 2008, [Online]. Available:
<https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2008-132.pdf>
- [5] The Sequence Diagram, 2004, [Online] Available:
<https://www.ibm.com/developerworks/rational/library/3101.html>
[Accessed: 11- Nov- 2018].
- [6] Pattern: API Gateway / Backend for Front-End, [Online]. Available:
<https://microservices.io/patterns/apigateway.html>
- [7] The WebSocket API (WebSockets), 2018, [Online]. Available:
https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
- [8] Adobe XD Design & Prototype Tool, [Online]. Available:
<https://www.adobe.com/tr/products/xd.html>