Photo Tagging (Where's Waldo) Project Ideation

A Photo Tagging app presents users with a busy and crowded illustration that contains many different people, objects, or places. The user’s task is to find a particular character that is hidden somewhere in the illustration.

# Goals

The project’s primary goal is to build an app that, when it’s finished, will feel very similar to a photo tagging app.

* Display a large photograph containing several elements the user is meant to find, e.g. Waldo, Jesus, Wilma etc. You can even name your own if you’d like to use custom pictures.
* The user will make selections for each character, and they will be given feedback on whether they are correct or not.

# Task

1. Think about what you’ll need to do to get all this working together. This is where it’s super helpful to think it completely through on paper or a whiteboard ahead of time! A few minutes of thought can save you from wasting an hour on coding.
2. Choose a photo.
3. Build the front-end functionality without actually using any calls to the back end yet. Specifically,

* Create the functionality that pops the targeting box.
* Create dropdown menu on the screen when the user clicks on the photo and removes it when the user clicks away.

1. Identify each character’s exact location on the photo using its pixel position and save that to your database.
2. Hook up the functionality for validating with your backend whether or not the user has clicked the right place for the character they selected from the dropdown.

* **When the user clicks the photo, it should place a targeting box around the portion of the photo the user has clicked. That box should contain a list of possible characters.**
* Depending on how you are getting the coordinates of a user’s clicks, different screen sizes may produce different coordinates. This could cause your app to record coordinates properly on a large screensize, but not smaller ones. Knowing this, you may need to implement methods to your click logic that will normalize coordinates across different screensizes.
* When the user selects one of the possible characters, you should check the backend to see if that character is within the targeting box.
* Provide the user with appropriate feedback (e.g. if wrong, an error message). If correct, place a marker on the photo in the character’s location.
* In either case, remove the targeting box until the user clicks again.

1. Tie it into your frontend so you can seamlessly select characters, validate them, and place the appropriate markers on the map if the selection was correct.
2. Keep track of how long it takes from when the photo is first loaded to when the user finally identifies all the characters. (It is advisable to do this on the server side, otherwise the user could hack their score.)
3. Display their score/time when they successfully identify all characters.
4. Create a popup that asks them to enter their name for the high scores table if they have earned it.
5. Record the provided name and completion time. (This will get a bit tricky since you’ll have anonymous users you need to keep track of!)
6. Play with it!
7. Push your solution to GitHub and deploy it to any of the hosting options.
8. **Extra credit:** Load many images into your database and allow the user to select from among them before starting the game.

# Structure

* Front-end app for the game and admin dashboard.
* Back-end app for data management.

# User privileges

* **Guest:** Unauthenticated user
* **Member:** Authenticated user
* **Admin:** An administrator

|  |  |  |  |
| --- | --- | --- | --- |
| **Privilege** | **Guest** | **Member** | **Admin** |
| Create an account | No | Yes | Yes |
| Play game | Yes | Yes | Yes |
| Add image | No | No | Yes |
| Update images | No | No | Yes |
| Access all images | No | Yes | Yes |
| Access leaderboard | No | Yes | Yes |
| Access dashboard | No | No | Yes |
| Add name to leaderboard | No | Yes | Yes |
| Add score to leaderboard | Yes | Yes | Yes |
| Delete user’s account | No | No | Yes |

# Applications objectives

## Pages

* **Homepage:**
* Welcome user
* User login/signup
* Game configuration
* Play UI
* Leaderboard navigation
* Dashboard navigation (admin-only)
* **Game page:**
* No navbar
* Image only
* **Leaderboard**
* Top 10 scores
* Play Game UI
* **Sign-up:** Registration page for members to create a new account as a staff or admin.
* First name
* Last name
* Username (unique)
* Email (unique)
* Password
* Admin passcode
* **Log-in page:** Allow staff to log in to their account.
* Email
* Password
* **Dashboard Homepage (admin-only)**
* List all images
* Add image UI
* Edit image’s data UI
* Delete image UI
* Unpublish/publish image UI
* **Add Image (admin-only)**
* Name (unique)
* URL (unique)
* Items to find’s information: Name, Image’s URL and Position (center-x, center-y, start-x, start-y, end-x, end-y)
* UI to add more items
* UI to delete each item
* UI to submit image
* **Update Image (admin-only)**
* Name (unique)
* URL (unique)
* Items to find’s information: Name, Image’s URL and Position (center-x, center-y, start-x, start-y, end-x, end-y)
* UI to add more items
* UI to delete each item
* UI to submit image

## Backend

* Provide RESTful APIs for Photo Tagging apps.
* Create users with statuses that determines what they can do on the game app.
* Allow admins to create and manage images.
* Allow administrators to manage users’ access to the restricted privileges.
* Sanitize and validate forms’ data.

# Models

## users

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | first\_name | last\_name | email | password |
| 1 | Oluwatobi | Sofela | contact@codesweetly.com | test |
| 2 | Sarah | Precious | sp@example.com | example |
| 3 | Dav | Emma | de@sample.com | sample |

## statuses

|  |  |
| --- | --- |
| id | name |
| 1 | Gamer |
| 2 | Admin |
| 3 | Anonymous |

## user\_status

|  |  |  |
| --- | --- | --- |
| id | user\_id | status\_id |
| 1 | 3 | 1 |
| 2 | 2 | 2 |

## leaderboard

|  |  |  |
| --- | --- | --- |
| id | user\_id | time |
| 1 | 3 | 30 |
| 2 | 1 | 100 |

## image

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| id | name | url | width | height | published | published\_date |
| 1 | Donut Pile | https://www.image-1.com | 1500 | 1088 | false | null |
| 2 | Something wonderful | https://www.image-2.com | 1277 | 1920 | true | 2025-03-07 |
| 3 | Time and seconds | https://www.image-3.com | 3000 | 3000 | false | null |

## character

|  |  |
| --- | --- |
| id | name |
| 1 | Carrot |
| 2 | Pizza |

## character\_image\_position

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| id | character\_id | image\_id | center\_x (px) | center\_y (px) | start\_x (px) | start\_y (px) | end\_x (px) | end\_y (px) |
| 1 | 2 | 3 | 0.447 | 0.356 | 0.419 | 0.325 | 0.476 | 0.387 |
| 2 | 1 | 2 | 0.969 | 0.306 | 0.951 | 0.278 | 0.987 | 0.335 |

# Tech stack

* **Frontend:** Astro.js, React.js
* **Backend:** Node.js, Express.js
* **Database:** PostgreSQL, Prisma ORM
* **Authentication:** Passport.js (local strategy), jsonwebtoken
* **Validation:** express-validator
* **Styling:** Tailwind CSS
* **Deployment:** Koyeb, Netlify

# Usage

## Backend

1. Clone the project

```bash

git clone https://github.com/oluwatobiss/find-x-backend.git

```

2. Navigate into the project repo

```bash

cd find-item-backend

```

3. Install dependencies

```bash

npm install

```

4. Create an environment variable file

```bash

touch .env

```

5. Define the project's environment variables

```

DB\_URI="postgresql://username:user\_password@localhost:5432/find\_x\_backend"

PORT=3000

JWT\_SECRET="example\_jwt\_secret"

ADMIN\_CODE="example-pass"

DATABASE\_URL=${DB\_URI}?schema=public

```

6. Migrate the project's schema to your database

```bash

npx prisma migrate dev

```

7. Start the server

```bash

npm run start

```

## Frontend

**Note:** The backend must be running for the frontend to function appropriately.

1. Clone the project

```bash

git clone https://github.com/oluwatobiss/find-x-frontend.git

```

2. Navigate into the project repo

```bash

cd find-x-frontend

```

3. Install dependencies

```bash

npm install

```

4. Create an environment variable file

```bash

touch .env

```

5. Define the project's environment variables

```

PUBLIC\_BACKEND\_URI="http://localhost:3000"

PUBLIC\_FRONTEND\_URI="http://localhost:4321"

```

6. Start the server

```bash

npm run dev

```