Technical Overview

User flow

How users receive data

Users are given three structures at the beginning, immediately after registration:

1. Telescope
2. Weather Balloon
3. Greenhouse

Users can then configure these structures each week to focus on specific projects (like Cloudspotting vs Dust devil hunting) and are provided with a collection of data points (referred to as `#anomalies`) throughout the rest of the week; these are classified (see below) via the structures.

`anomalies` table definition:

```SQL
create table
public.anomalies (
        id bigint generated by default as identity,
        content text null,
        anomalytype text null,
        type text null,
        avatar_url text null,
        created_at timestamp with time zone not null default now(),
        configuration jsonb null,
        "parentAnomaly" bigint null,
        "anomalySet" text null,
        constraint baseplanets_pkey primary key (id),
        constraint anomalies_parentAnomaly_fkey foreign key ("parentAnomaly") references
anomalies (id)
) tablespace pg_default;
```

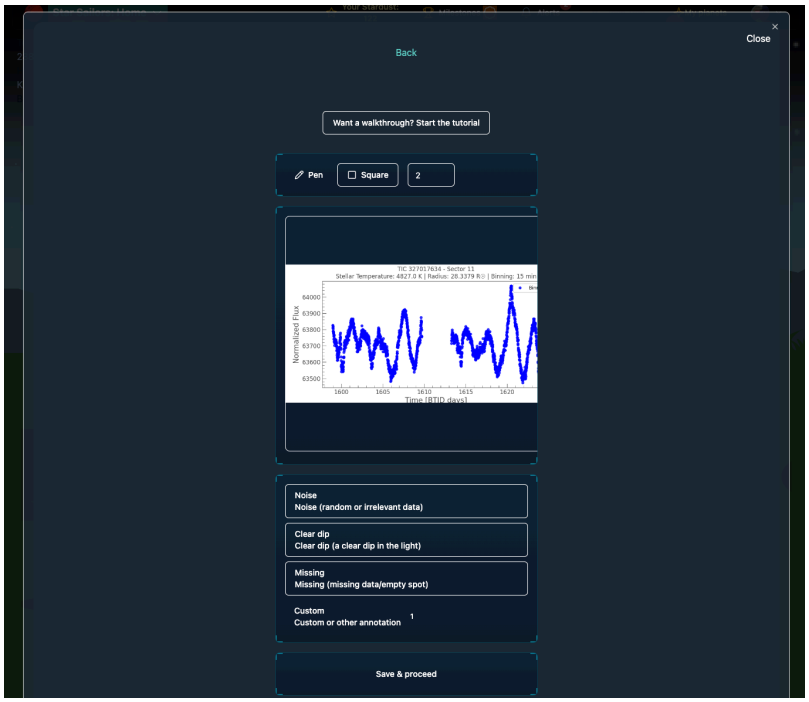
For example, your Telescope may be configured to return exoplanet (lightcurve) data for the `#PLanetHunters` project. This involves the frontend sending a request to receive a random anomaly of the correct type, and then returning its data and any associated images:

```
const fetchAnomalies = async () => {
        if (!session) {
                return;
        }

        try {
                const { data: anomalyData, error } = await supabase
```

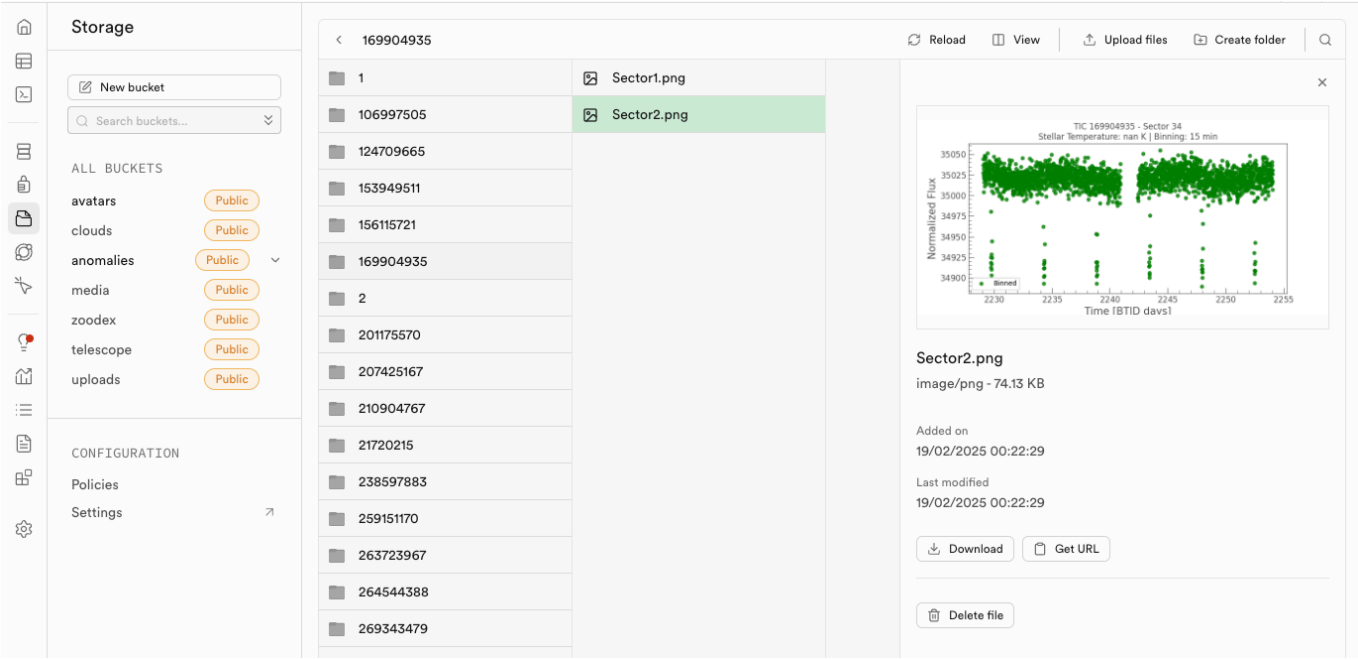
```
                        .from('anomalies')
                        .select('*')
                        .eq('anomalySet', 'telescope-tess')
                        .eq('id', anomalyid);

                setAnomaly(anomalyData[0]);

setImageUrl(`${supabaseUrl}/storage/v1/object/public/anomalies/${anomalyid}/Sector1.png`);
        } catch (error: any) {
                console.error(error);
                return;
        }
};
```

So, we first fetch the `anomalies` table, with `anomalySet` referring to the project (so `#telescope-tess` refers to all `#anomalies` part of the `#PLanetHunters` project). Then, we have a consistent image/file structure, where each project has its own section inside a storage bucket in our Supabase instance, grouped by the `#anomalyid` value being returned after fetching.

Then, we present the image to the modal that represents the user's structure (in this case, the Telescope) (please note that this design is subject to change):



Storage bucket > file (example):



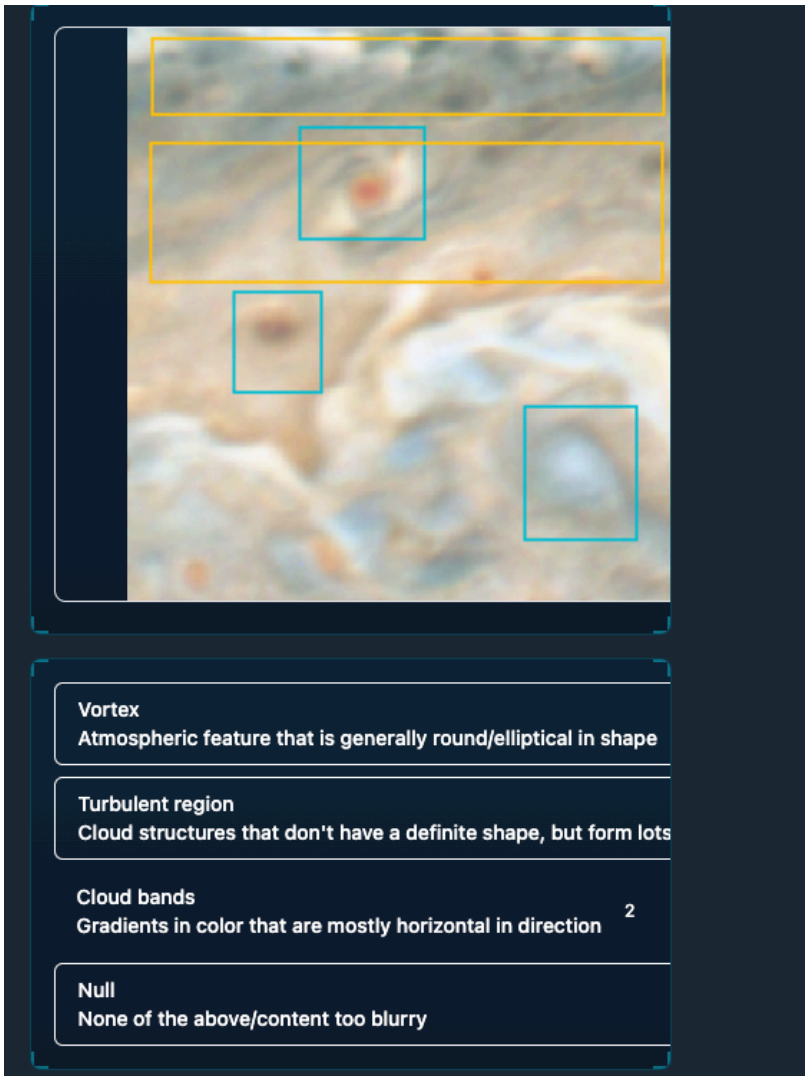
Classifying data

There are two main ways for users to make contributions (inputs) - classifications (of external research data), and providing their own research data (for example, photos - which for now isn't relevant to OnOrbit).

We have a relatively simple `classifications` table, which stores a record of every *primary* post/contribution of this type:

```SQL
create table
public.classifications (
        id bigint generated by default as identity,
        created_at timestamp with time zone not null default now(),
        content text null,
        author uuid null,
        anomaly bigint null,
        media json null,
        classificationtype text null,
        "classificationConfiguration" jsonb null,
        constraint classifications_pkey primary key (id),
        constraint classifications_anomaly_fkey foreign key (anomaly) references anomalies
(id),
        constraint classifications_author_fkey foreign key (author) references profiles (id)
) tablespace pg_default;
```

When users annotate an "anomaly", we record it as a collection of answers from the user, identifying what they see. This content is then sent back to the relevant researchers; we've done our best to keep the primary data format from the original data source (e.g. zooniverse projects).



	63	2024-12-20 11:06:50.580573+00	I see some possible rings around the edge	4d9a57e6-ea3c-4836-aa18-1c3ee...	8423850804	[],"https://hlufptwhzkpkkjztimzo.supabas	cloud
	68	2025-01-28 12:28:00.23171+00	I see some small loops that appear indicat	4d9a57e6-ea3c-4836-aa18-1c3ee...	9490482202	[],["https://hlufptwhzkpkkjztimzo.supaba	cloud
	78	2025-02-27 13:27:06.350462+0	I think there's a small loop, but it's quite h	4d9a57e6-ea3c-4836-aa18-1c3ee...	84238508	[],["https://hlufptwhzkpkkjztimzo.supaba	cloud
	79	2025-02-27 13:33:02.76879+00	I think there's definitely one loop, maybe	4d9a57e6-ea3c-4836-aa18-1c3ee...	9490482201	[],["https://hlufptwhzkpkkjztimzo.supaba	cloud
	80	2025-03-01 09:15:55.904716+0	One point of interest that could either be	4d9a57e6-ea3c-4836-aa18-1c3ee...	84238508	[],["https://hlufptwhzkpkkjztimzo.supaba	cloud
Expand row		2025-03-14 10:41:35.247109+00	A few arches towards the right of the pan	4d9a57e6-ea3c-4836-aa18-1c3ee...	75332022	[],["https://hlufptwhzkpkkjztimzo.supaba	cloud
	81	2025-03-06 09:52:49.670263+(	And some blurry stuff, too	4d9a57e6-ea3c-4836-aa18-1c3ee...	77764157	[],["https://hlufptwhzkpkkjztimzo.supaba	lidar-jovianVortexHunter
	86	2025-03-23 15:58:52.38655+00	The beginning/bottom of what looks like	4d9a57e6-ea3c-4836-aa18-1c3ee...	77762176	[],["https://hlufptwhzkpkkjztimzo.supaba	lidar-jovianVortexHunter
	19	2024-07-06 11:20:23.607282+0	parabolic curves with mostly the same flu	ba8e612a-7d21-47b0-86fc-62c177...	2	[],"https://hlufptwhzkpkkjztimzo.supabas	lightcurve
	25	2024-07-08 15:28:49.621386+0	Most of the dips appear to be a similar mi	ba8e612a-7d21-47b0-86fc-62c177...	2	[],"https://hlufptwhzkpkkjztimzo.supabas	lightcurve

Classifications are posted via a tsx component:

```
const createPost = async () => {
        const flattenedOptions = classificationOptions.flat();
        const classificationOptionsObj = Object.fromEntries(

        Object.entries(selectedOptions).map(([key, value]) => [

        flattenedOptions.find((option) => option.id === parseInt(key))?.text ||

        "",

        value,
```

```
        ])

        );



        const classificationConfiguration = {

        classificationOptions: classificationOptionsObj,

        additionalFields,

        parentPlanetLocation: parentPlanetLocation || null,

        activePlanet: activePlanet?.id,

        createdBy: inventoryItemId ?? null,

        classificationParent: parentClassificationId ?? null,

        annotationOptions: annotationOptions,

        };


try {
        const { data: classificationData, error: classificationError } =
                await supabase
                        .from("classifications")
                        .insert({
                                author: session?.user?.id,
                                content,
                                media: [uploads, assetMentioned],
                                anomaly: anomalyId,
                                classificationtype: anomalyType,
                                classificationConfiguration,
                        })

                        .single();

        if (classificationError) {
                alert("Failed to create classification. Please try again.");
                return;
        } else {
                setClassificationOutput(classificationConfiguration);
                setContent("");
                setSelectedOptions({});
                setUploads([]);
                setPostSubmitted(true);
        };

        const { data: profileData, error: profileError } = await supabase
                .from("profiles")
                .select("classificationPoints")
                .eq("id", session?.user?.id)
                .single();
```

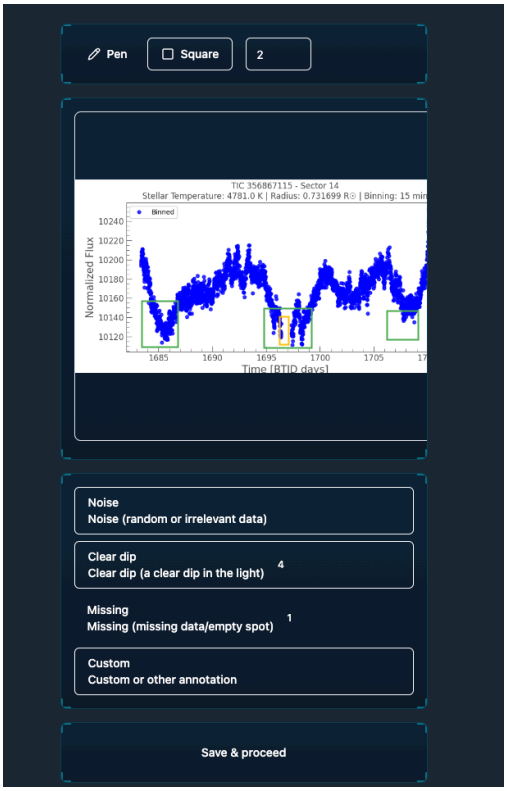
```
        if (profileError) throw profileError;
        const newClassificationPoints = (profileData?.classificationPoints || 0) + 1;

        const { error: updatePointsError } = await supabase
                .from("profiles")
                .update({ classificationPoints: newClassificationPoints })
                .eq("id", session?.user?.id);
        if (updatePointsError) throw updatePointsError;
        router.refresh();
        window.location.reload();
} catch (error: any) {

        console.error("Unexpected error:", error);

};



router.refresh();

};
```

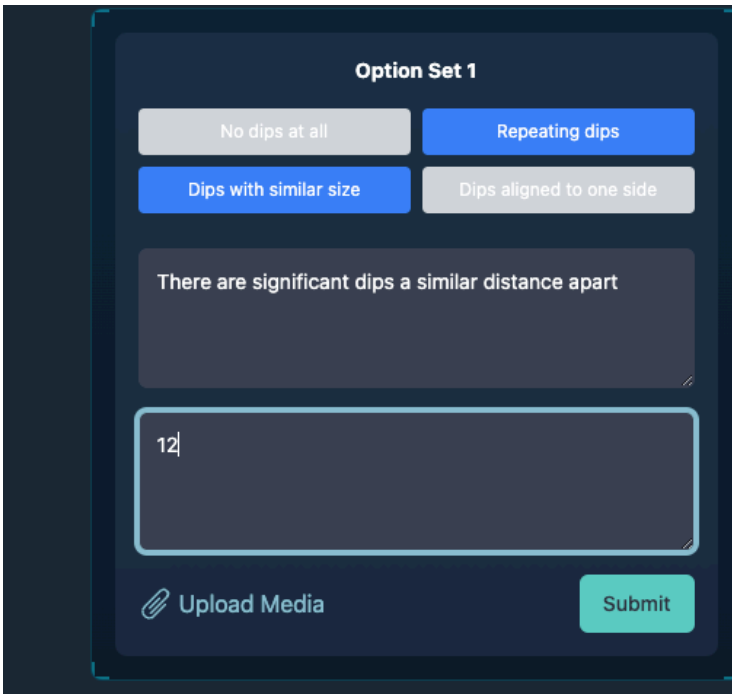
Projects

Planet Hunters

1. Dataset: `lightkurve` (python package), Planet Hunters TESS (secondary)
2. Goal: Find and catalogue exoplanet candidates

Classification process:



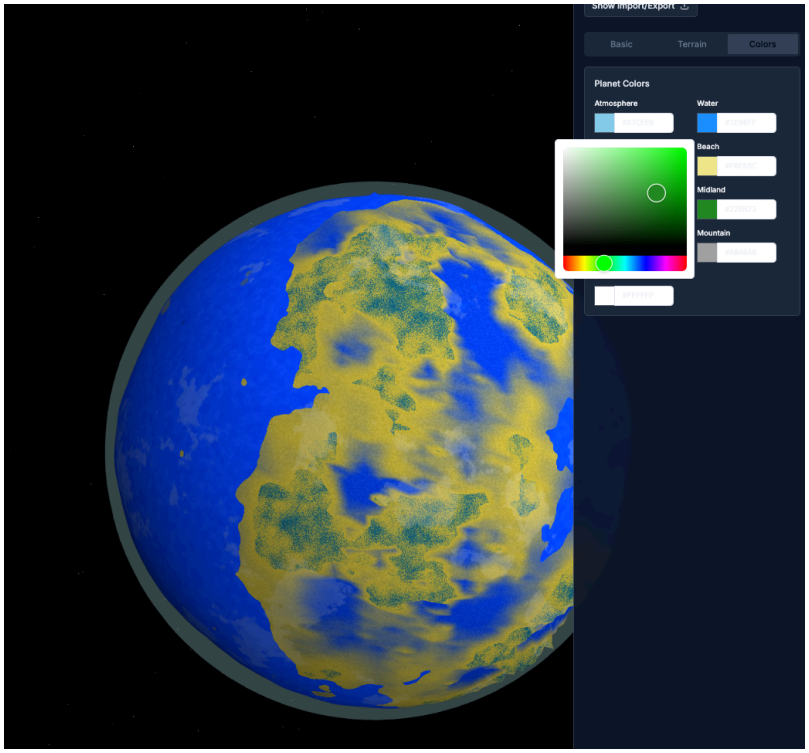


Step 1 - users annotate the graph
Step 2 - users answer some questions about what they see, and input their proposed planet period (both input forms are labelled)

Reward for user:

1. A planet they can explore, build on, collect resources, etc (later)
2. A special planet customiser ("paint your planet"), allowing them to share their discoveries and combine their imagination and "real" data

Note - this is a preview of the current "painter":

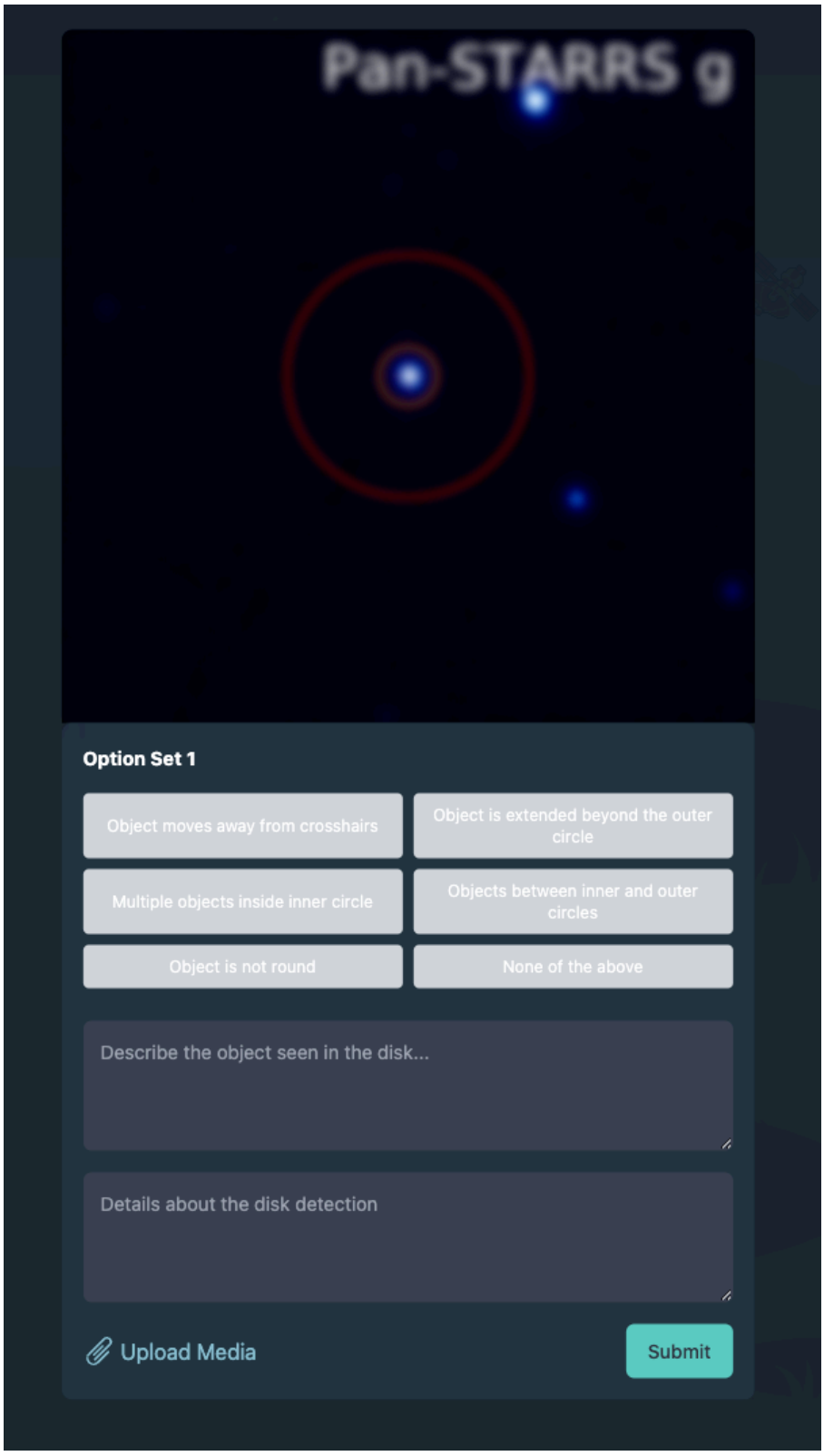


This component is built using `3js`, but I'm fairly confident we could convert it to a Unity build considering that's how Star Sailors started originally (pre-2024).

A lot of the other projects don't have a concrete reward beyond being another "data-source" or location to explore/work with in-game. Having said that, we do have ideas - but we want to track user behaviour a bit more first before committing to potentially less popular projects strongly right now. Rewards and other incentives/narrative elements will of course be distributed retrospectively to all users for all classifications.

Disk Detective

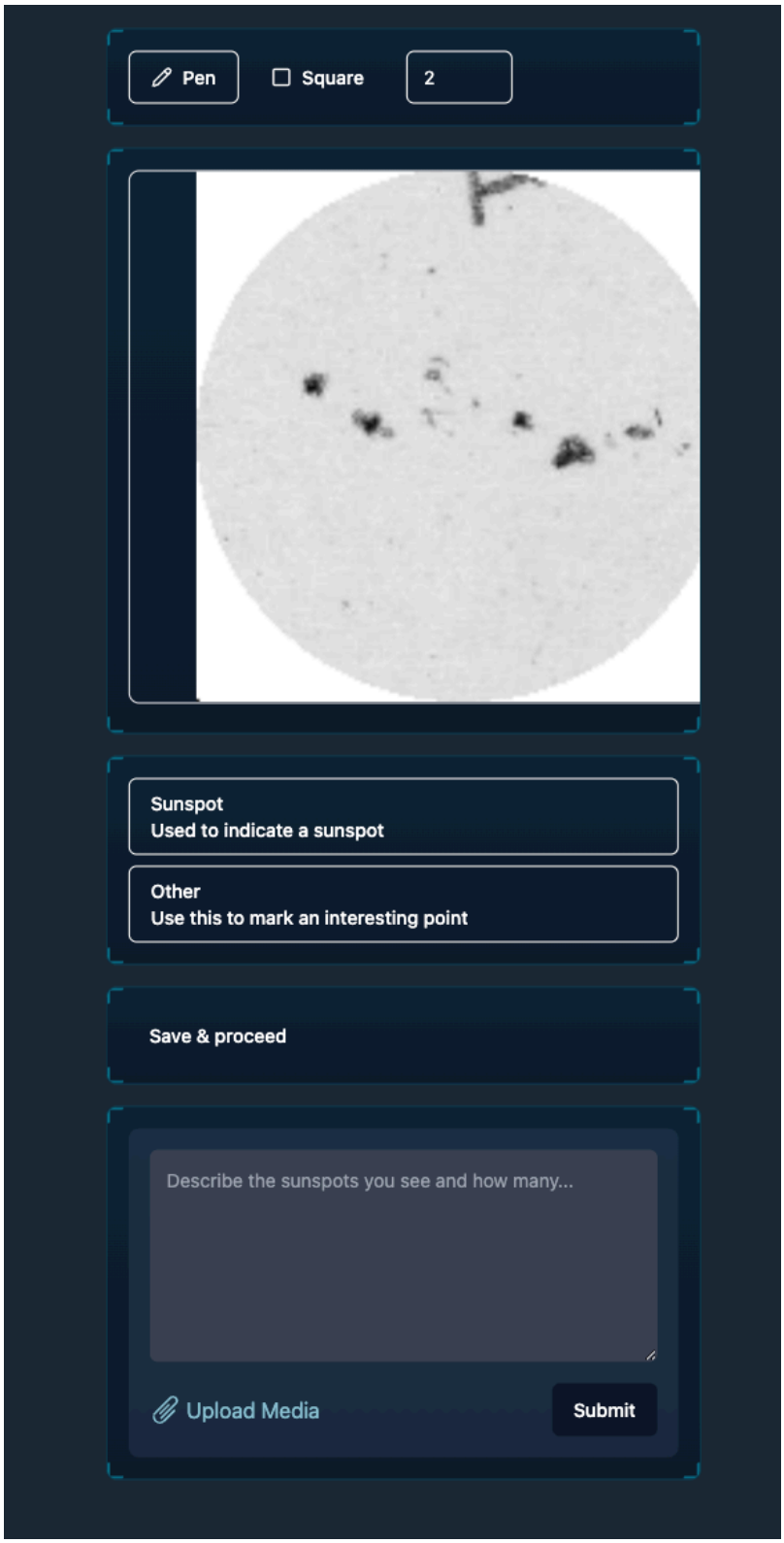
1. Datasets: DiskDetective
2. Goal: Catalogue potential debris disks to find early planet formation events



1. Main goal during the initial classification process is to keep a similar format to the live Zooniverse project with some additional custom data.
2. No concrete plans to integrate annotation in, yet.

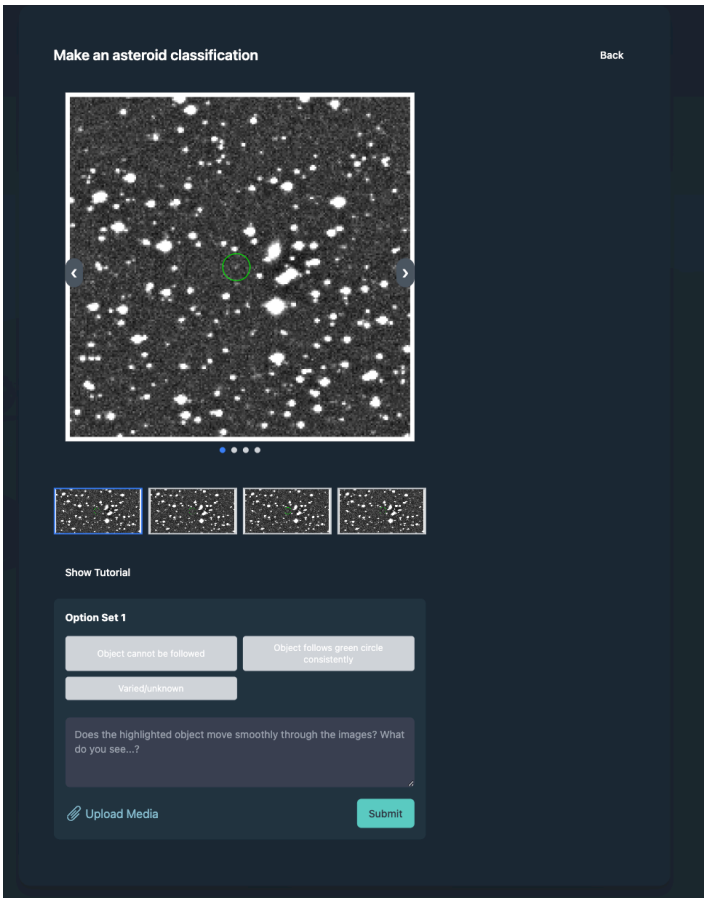
Sunspots

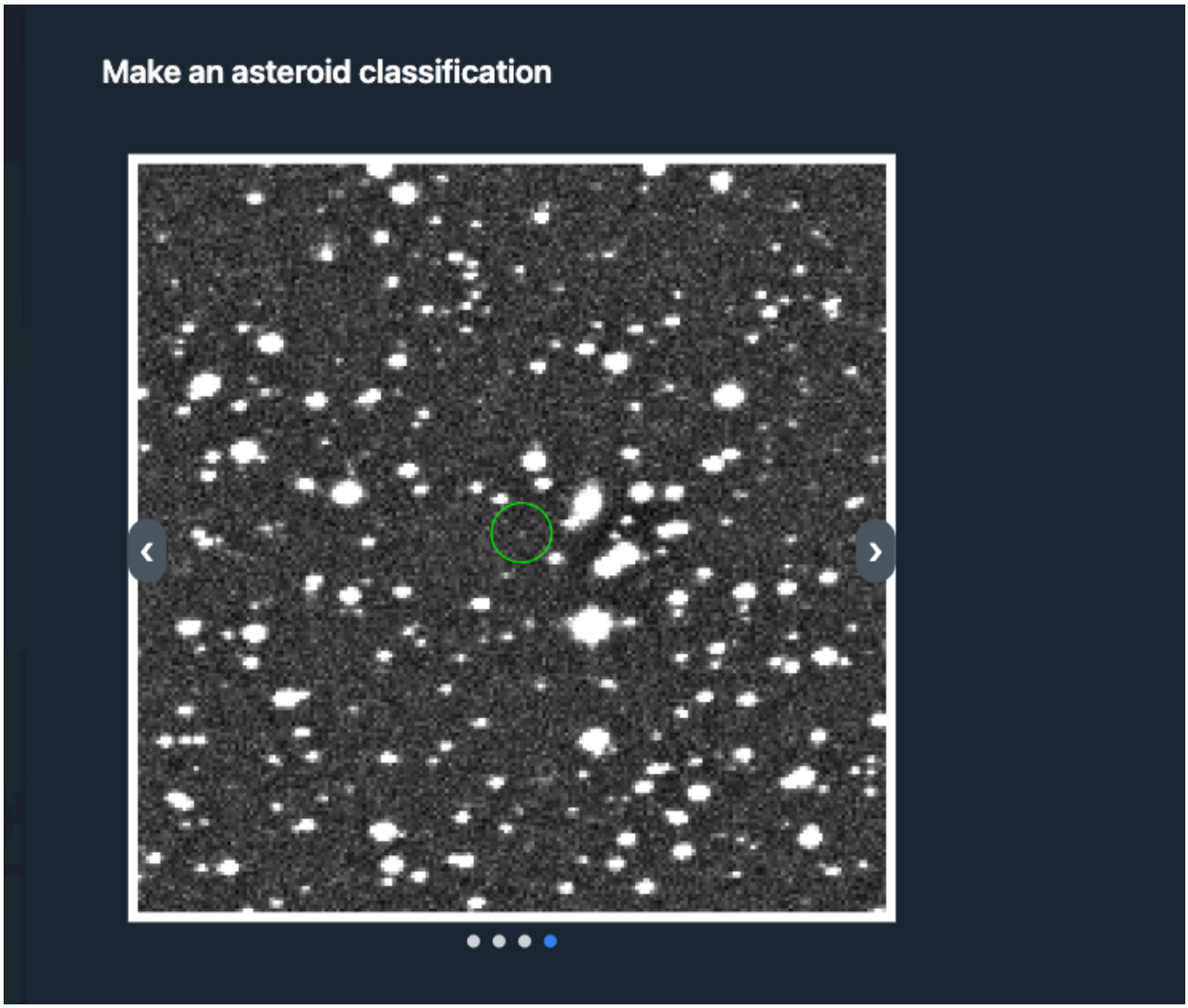
1. Datasets: Sunspot Detective
2. Goal: catalogue the number of sunspots from historical dataset



Daily Minor Planet

1. Dataset: The DailyMinorPlanet
2. Goal: Users annotate images to identify new asteroid [candidates]

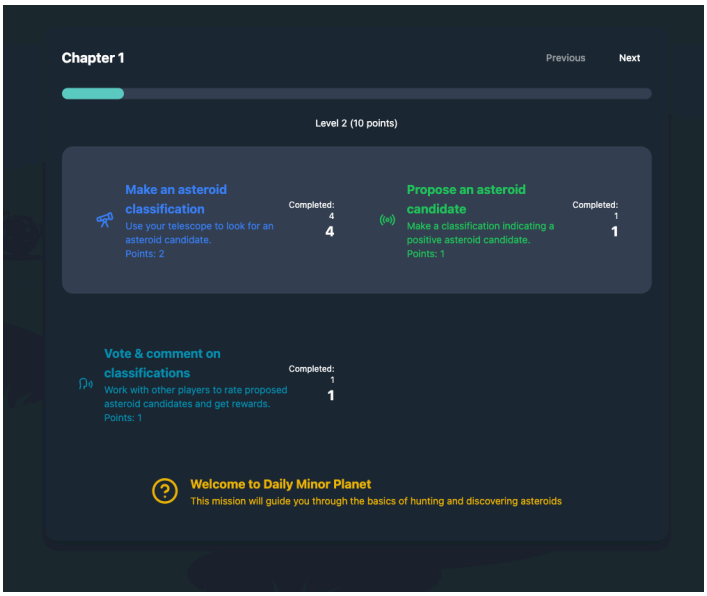

Make an asteroid classification
Back
Show Tutorial
Option Set 1
Object cannot be followed
Object follows green circle consistently
Varied/unknown
Does the highlighted object move smoothly through the images? What do you see...?
Upload Media
Submit

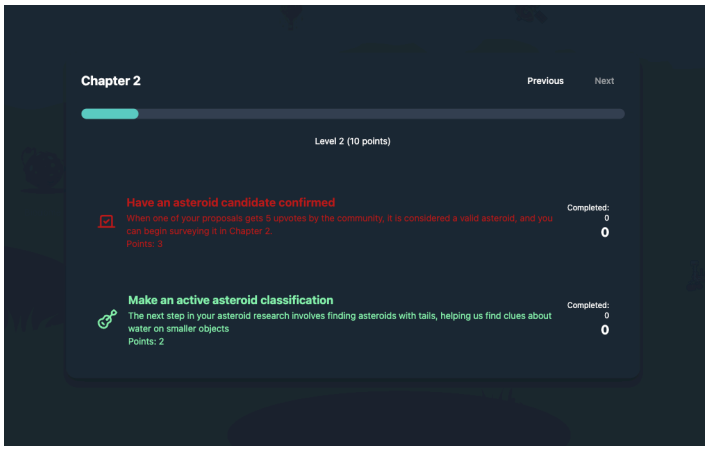


Note: like with #Disk-Detective , we currently don't have annotation features for DMP project, however it is something we would like to add.

Reward:

1. Another location and resource source/site
2. Access to the #ActiveAsteroids project:





Each project has multiple activities (commenting, voting on classifications, etc). Some projects have multiple datasets that can be unlocked. Active Asteroids is the logical next step for the Asteroid classification project [group].

DMP: >> Active Asteroids

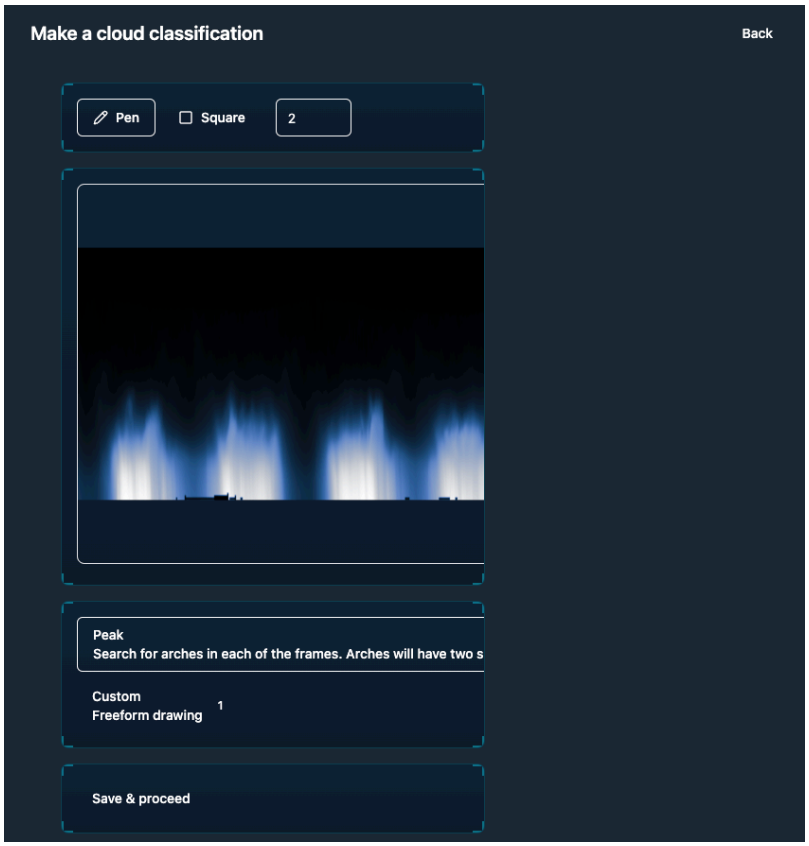
1. Dataset for active asteroids
2. Goal: find asteroids that resemble comets to help astronomers learn more about water transportation in the early days of the #Solar-System

Other projects

1. Greenhouse -
    1. Annotate Burrowing Owl behaviour
    2. Annotate Iguana behaviour
    3. Annotate penguin behaviour
    4. Annotate plankton behaviour
2. Weather Balloon (Geology/Meteorology)
    1. AI4M
    2. Planet Four

Cloudspotting on Mars

1. Dataset: Cloudspotting on Mars
2. Goal: Find clouds during "Mars Year 29" and identify how mesospheric clouds change during different seasons on Mars

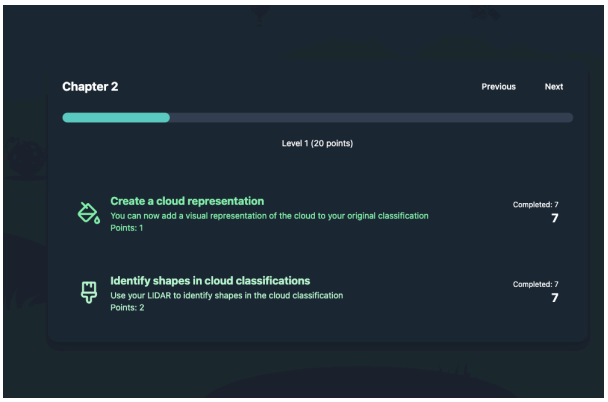


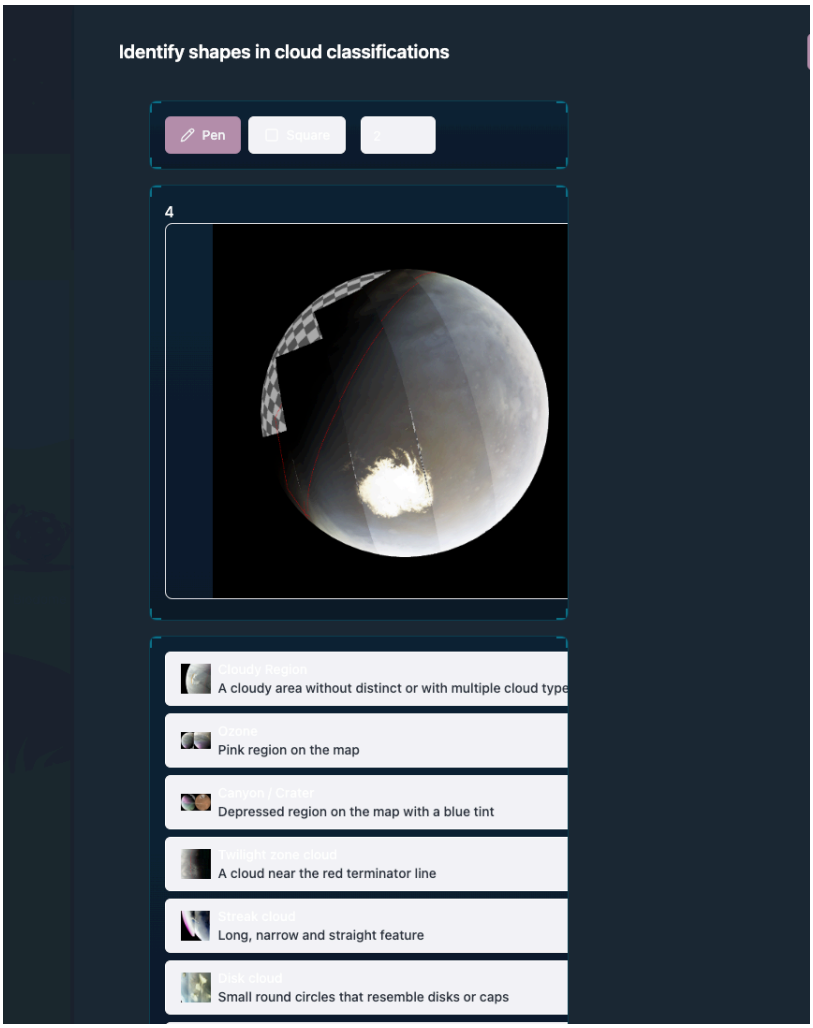
Reward for user:

1. Users can then participate in the next phase of Cloudspotting, which is "Cloudspotting on Mars: Shapes" (see below)

Cloudspotting on Mars: Shapes

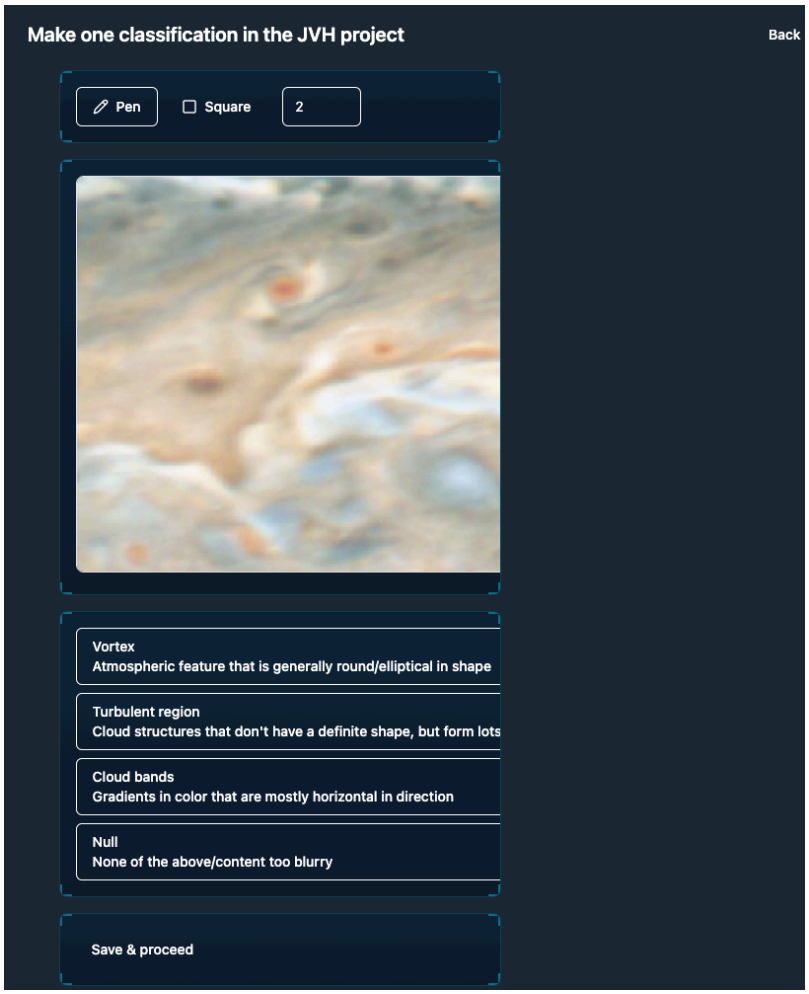
1. Dataset: Cloudspotting on Mars: Shapes
2. Goal: Further investigations of how clouds form on Mars





Jovian Vortex Hunters

1. Dataset: [Jovian Vortex Hunters]
2. Goal: Identify the diversity of cloud structures on Jupiter and what leads to this diversity



🖼️ Types of Data and Projects

Image Types and File Formats

Users interact with a variety of astrophysical data, primarily focusing on:

- **Light Curves**: Time-series data representing the brightness of celestial objects over time. These are often visualised as graphs.
- **Spectral Data**: Information about the spectrum of light from celestial objects, useful for determining composition and other properties.
- **Anomaly Visualisations**: Graphical representations highlighting unusual patterns or features in the data.

These data types are typically stored in formats such as CSV for raw data and PNG or JPEG for visualizations.

Available Projects

The platform hosts multiple projects, each corresponding to different datasets or research objectives. Examples include:

- **Exoplanet Detection**: Users analyse light curves to identify potential exoplanets.(GitHub)
- **Variable Star Classification**: Classifying stars based on variability patterns.

- **Anomaly Detection**: Identifying unusual patterns that may indicate new or rare astrophysical phenomena.

Each project provides specific datasets and classification tasks tailored to its objectives.

🎮 User Interaction and Classification Process

Game Mechanics

The platform employs gamification to enhance user engagement:

- **Missions and Quests**: Users undertake missions that guide them through classification tasks, providing structure and goals.
- **Achievements and Rewards**: Successful classifications and contributions earn users badges, points, or other virtual rewards.
- **Interactive Tutorials**: New users are onboarded through tutorials that explain the classification process and tools.

Classification Workflow

1. **Data Presentation**: Users are presented with data visualisations (e.g., light curves).
2. **Analysis Tools**: Interactive tools allow users to zoom, annotate, and manipulate the data for better analysis.
3. **Classification Submission**: Users classify the data based on observed patterns and submit their findings.
4. **Community Review**: Submissions may be reviewed or discussed within the community for validation.

🔄 Data Flow and Backend Integration

Supabase Integration

Supabase serves as the backend infrastructure, providing:

- **Authentication**: Managing user sign-ups, logins, and session handling.
- **Database**: Storing user data, classifications, and project information in a PostgreSQL database.
- **Storage**: Handling file uploads, such as user-submitted images or annotations.

Frontend Architecture

Built with Next.js and React, the frontend includes:

- **Components**: Reusable UI elements for displaying data, forms, and interactive tools.
- **Pages**: Routes corresponding to different views, such as project overviews, classification interfaces, and user profiles.
- **Hooks**: Custom React hooks for managing state and side effects, particularly for data fetching and user interactions.

🔬 Integration with Lightkurve and Similar Services

The platform leverages external services like `#Lightkurve` to enrich its datasets:

- **Data Acquisition**: Lightkurve is used to download and process light curve data from missions like Kepler and TESS.
- **Preprocessing**: Data is cleaned and formatted into CSV files, which are then uploaded to Supabase storage.
- **Visualization**: Processed data is visualized using Python libraries, and the resulting images are stored for user interaction.
