# Astrogram

Project submitted to the
SRM University – AP, Andhra Pradesh
as a part of Full Stack Course which is the part of

**Bachelor of Technology**

in

**Computer Science and Engineering**
School of Engineering and Sciences

**Submitted by**

Ashok Aryal (AP23110011540)
Tushiq Shreyas Sankara (AP23110011165)
RAVALOSON Iavy Finaritra (AP23110011176)


Under the Guidance of

**Mr. Himanshu Mishra**
Department of CSE

SRM University – AP
Neerukonda, Mangalagiri, Guntur

# Introduction:-

Astrogram is a modern React-based web application that connects users directly with space imagery and information from NASA. It uses the official **NASA Astronomy Picture of the Day (APOD) API** to fetch real-time images, titles, and explanations of astronomical objects.

The application provides a **Discover** feed to explore daily and historical APOD entries, a personalised **Stellarium** section where users can save their favourite images, and an **Account** area to manage profile settings. Built entirely using **React.js**, the project focuses on clean UI, smooth navigation, authentication, and API integration.

Astrogram aims to make space more accessible and engaging, blending scientific content with an intuitive and visually rich interface.

# Scenario-Based Intro:-

Imagine you are a space enthusiast who loves checking NASA's Astronomy Picture of the Day. Instead of manually visiting the website and searching for previous days, you open **Astrogram**.

On the **Discover** page, you see a stunning visualization of a black hole and its accretion disk. A short description explains what you are seeing, and with a single click on **"Save to Stellarium"**, the image gets added to your personal collection.

Later, you log in from another device. You open the **My Stellarium** page and find all your favourite images neatly arranged as a curated gallery of the cosmos. You can revisit them anytime, read their descriptions, and continue exploring more NASA images.

Astrogram turns NASA's scientific data into a personalised, interactive experience that encourages curiosity and continuous learning about the universe.

# Target Audience:-

Astrogram is designed for a wide range of users:

- **Space Enthusiasts:** People who are passionate about astronomy, galaxies, planets, and cosmic phenomena.
- **Students and Learners:** School and college students who want to visually explore space concepts and use NASA images in their projects.

- **Educators:** Teachers who need a simple tool to showcase real NASA images and explanations in the classroom.
- **Developers and Tech Learners:** Those who want to understand how to build a real-world React application with API integration and authentication.
- **General Public:** Anyone interested in seeing beautiful space images in a modern, easy-to-use web interface.

# Project Goals and Objectives:-

## Goals

- Build a **React.js** application that fetches data from NASA's APOD API.
- Provide a clean, modern **UI/UX** with smooth navigation between pages.
- Implement **user authentication** (signup and login).
- Allow users to **save and manage favourite** NASA images in a personalised section called Stellarium.

## Objectives

- Integrate NASA's APOD endpoint using `fetch` or `axios`.
- Use **React Router** for page navigation (Discover, Stellarium, Account, etc.).
- Use **state management** to handle user data, favourites, and authentication status.
- Design components that are responsive and work on both desktop and laptop screens.

# Key Features:-

- **Discover Feed:** Shows the Astronomy Picture of the Day along with its title, date, and description. Users can explore different days and view high-quality space images.
- **Save to Stellarium:** Logged-in users can save any NASA APOD image to their personal Stellarium collection with a single click.

- **My Stellarium (Favourites Gallery):** Displays all the saved images as a beautiful card-based gallery with titles and dates. This acts as the user's curated collection of the cosmos.
- **User Authentication:** Signup and login functionality is implemented so that each user has their own favourites and profile.
- **Account Settings Page:** Allows the user to update display name and password and also provides an option to delete the account.
- **Modern Space-Themed UI:** The entire application uses a dark, starry background with clean typography and card layouts to match the theme of space.

# Pre-Requisites:-

To develop and run the Astrogram project, the following prerequisites are required:

- **Node.js and npm:** Required to create and run the React project, install dependencies, and manage packages.
- **React.js Basics:** Understanding of components, props, hooks (`useState`, `useEffect`), and React Router.
- **HTML, CSS, and JavaScript:** Basic knowledge to structure pages, style components, and implement interactivity.
- **NASA API Key:** A personal API key obtained from the NASA developer portal to access the APOD API.
- **Code Editor:** A development environment like Visual Studio Code for writing and organising the code.
- **Version Control (Optional):** Git and GitHub for managing versions and backups of the project.

## Project Structure:-

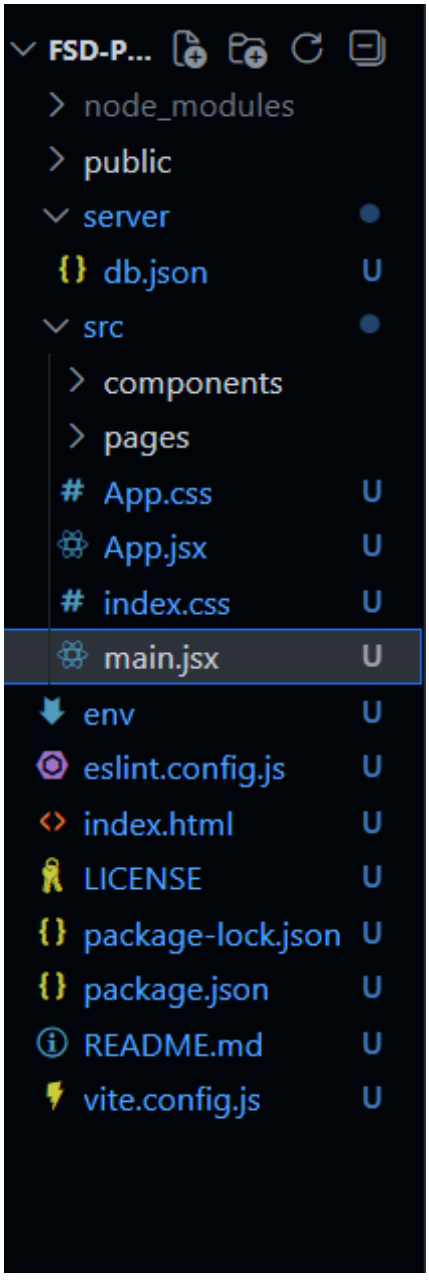



**Figure X: Project Structure**

# Project Demo:-

Demo Link: https://bit.ly/Astrogram-Demo

Here we show the working demo of the Astrogram application and how the user interacts with the main features such as Discover, Stellarium, and Account Settings.

# Milestone 1: Project Setup and Configuration:-

- Installation of Node.js, React.js and required libraries.
- Creation of the React project using Vite.
- Setting up folder structure and routing.

# Milestone 2: Web Development:-

## 1. Setup React Application:-

- Created React application using Vite.
- Connected App component with React Router.
- Verified app is running at `http://localhost:5173/`.

## 2. UI Component Design:-

- Designed Navbar, Discover, Stellarium, and Account Settings components.
- Applied a dark, space-themed layout with responsive design.
- Implemented reusable PictureCard and Loader components.

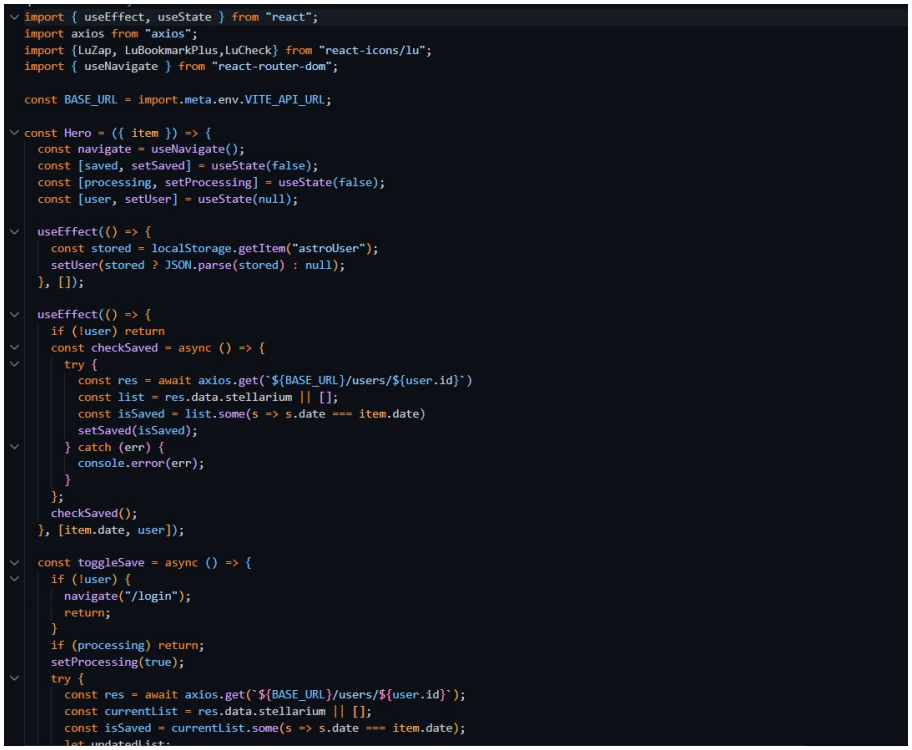
## 3. Implement Frontend Logic:-

- Integrated NASA APOD API service for fetching data.
- Implemented authentication using AuthContext.
- Connected Stellarium favourites logic with user actions.

# Discover Component:-

## Code Description:-

- Fetches NASA APOD data using the NASA API.
- Displays the image, title, description, and date on the home page.
- Includes a "Save to Stellarium" button for logged-in users.
- Loads data on component mount using `useEffect`.



```
import { useEffect, useState } from "react";
import axios from "axios";
import {LuZap, LuBookmarkPlus,LuCheck} from "react-icons/lu";
import { useNavigate } from "react-router-dom";

const BASE_URL = import.meta.env.VITE_API_URL;

const Hero = ({ item }) => {
  const navigate = useNavigate();
  const [saved, setSaved] = useState(false);
  const [processing, setProcessing] = useState(false);
  const [user, setUser] = useState(null);

  useEffect(() => {
    const stored = localStorage.getItem("astroUser");
    setUser(stored ? JSON.parse(stored) : null);
  }, []);

  useEffect(() => {
    if (!user) return
    const checkSaved = async () => {
      try {
        const res = await axios.get(`${BASE_URL}/users/${user.id}`)
        const list = res.data.stellarium || [];
        const isSaved = list.some(s => s.date === item.date)
        setSaved(isSaved);
      } catch (err) {
        console.error(err);
      }
    };
    checkSaved();
  }, [item.date, user]);

  const toggleSave = async () => {
    if (!user) {
      navigate("/login");
      return;
    }
    if (processing) return;
    setProcessing(true);
    try {
      const res = await axios.get(`${BASE_URL}/users/${user.id}`);
      const currentList = res.data.stellarium || [];
      const isSaved = currentList.some(s => s.date === item.date);
      let updatedList;
```

**Figure 1: Discover Component Code (replace later)**

# Stellarium Component:-

## Code Description:-

- Displays all saved favourite NASA APOD images.
- Reads user favourites from storage or backend.
- Uses reusable `PictureCard` components for each item.

src > pages > Stellarium.jsx > Stellarium

```jsx
import { useEffect, useState } from "react";
import axios from "axios";
import { LuLock, LuBookmark } from "react-icons/lu";
import SavedCard from "../components/SavedCard";
import { Link } from "react-router-dom";
import Loader from "../components/Loader";

const BASE_URL = import.meta.env.VITE_API_URL;

const Stellarium = () => {
  const [savedItems, setSavedItems] = useState([]);
  const [loading, setLoading] = useState(true);
  const [user, setUser] = useState(null);

  useEffect(() => {
    const stored = localStorage.getItem("astroUser");
    setUser(stored ? JSON.parse(stored) : null);
  }, []);

  useEffect(() => {
    if (!user) {
      setSavedItems([]);
      setLoading(false);
      return;
    }
    let mounted = true;
    const loadSaved = async () => {
      setLoading(true);
      try {
        const res = await axios.get(`${BASE_URL}/users/${user.id}`);
        if (!mounted) return;
        setSavedItems(res.data.stellarium || []);
      } catch (err) {
        console.error("Error loading stellarium:", err);
      } finally {
        if (mounted) setLoading(false);
      }
    };
    loadSaved();
    return () => {
      mounted = false;
    };
  }, [user]);
```

**Figure 2: Stellarium Component Code (replace later)**

# Authentication (Login / Signup):-

## Code Description:-

- Handles user login and signup using `AuthContext`.
- Collects input (email, password, etc.) and validates data.
- Updates global authentication state after successful login/signup.

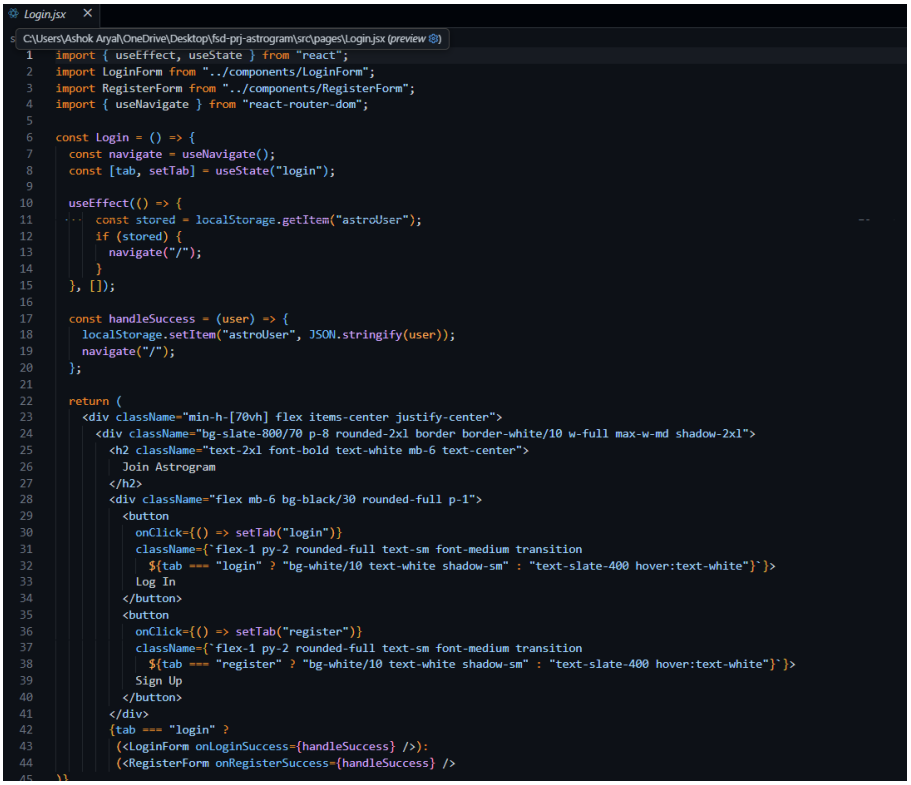



**Figure 3: Authentication Code (replace later)**

# Account Settings Component:-

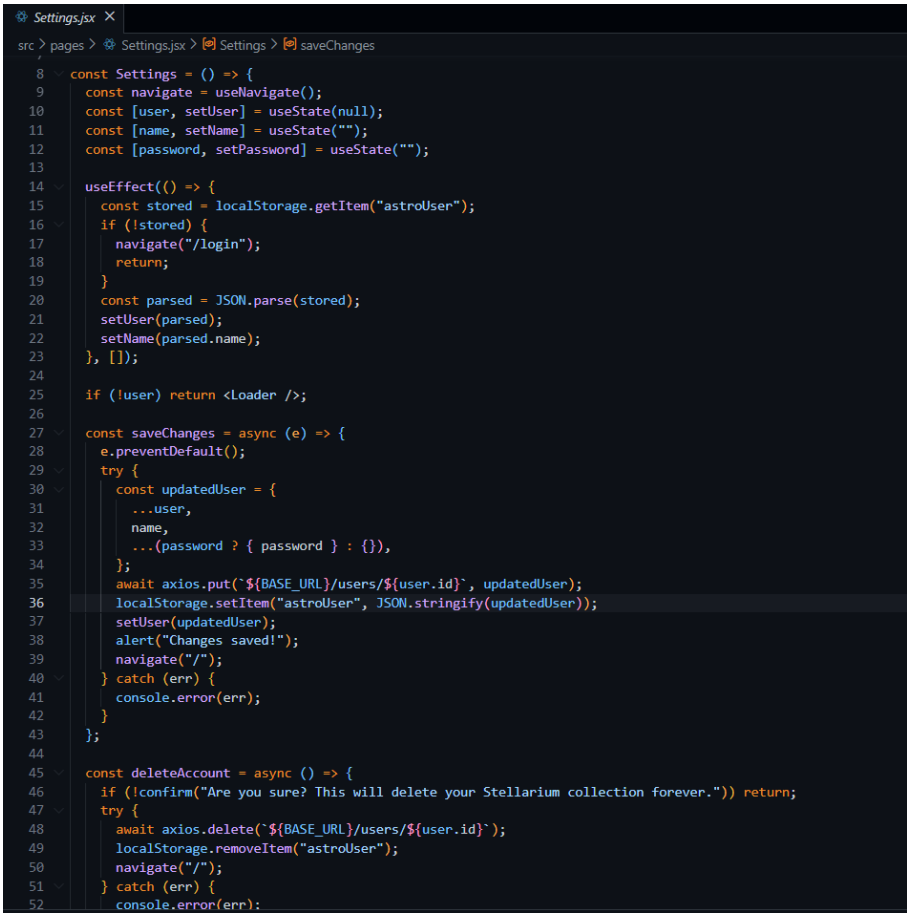



**Figure X: Account Settings Component Code (replace later)**

## Code Description:-

- Allows users to edit their display name and password.
- Provides an option to delete the user account permanently.
- Uses the authentication context to read and update profile details.
- Shows a simple, user-friendly UI for managing account settings.

# Project Execution:-

After completing the development, run the project using:

```
npm run dev
```

Below are the screenshots of the Astrogram application.

# Home / Discover Page

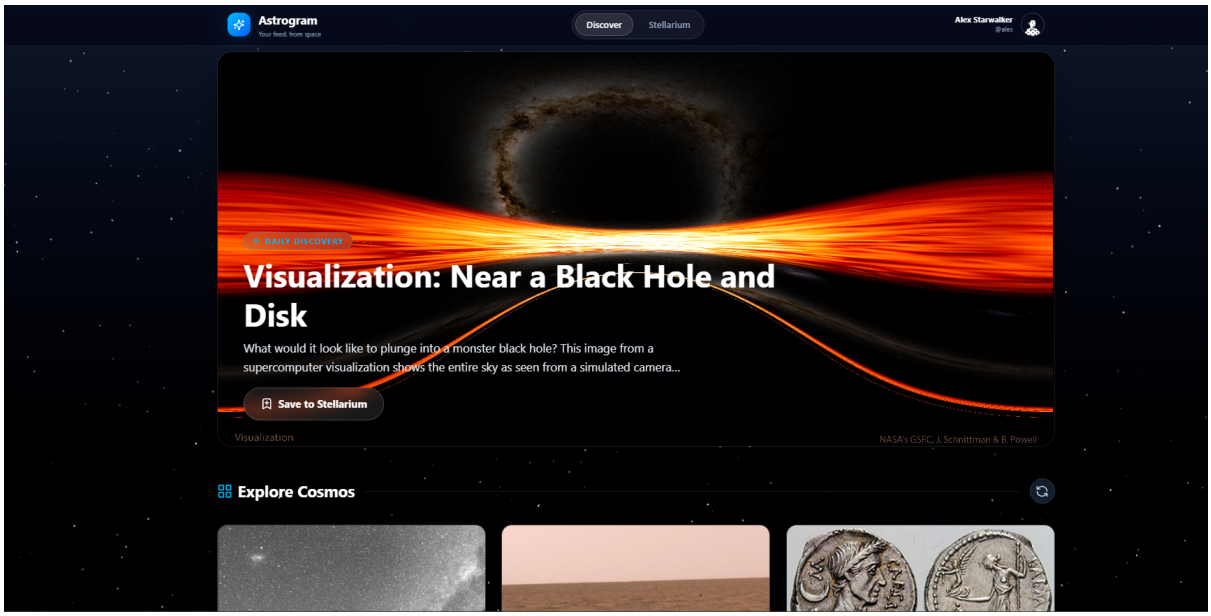
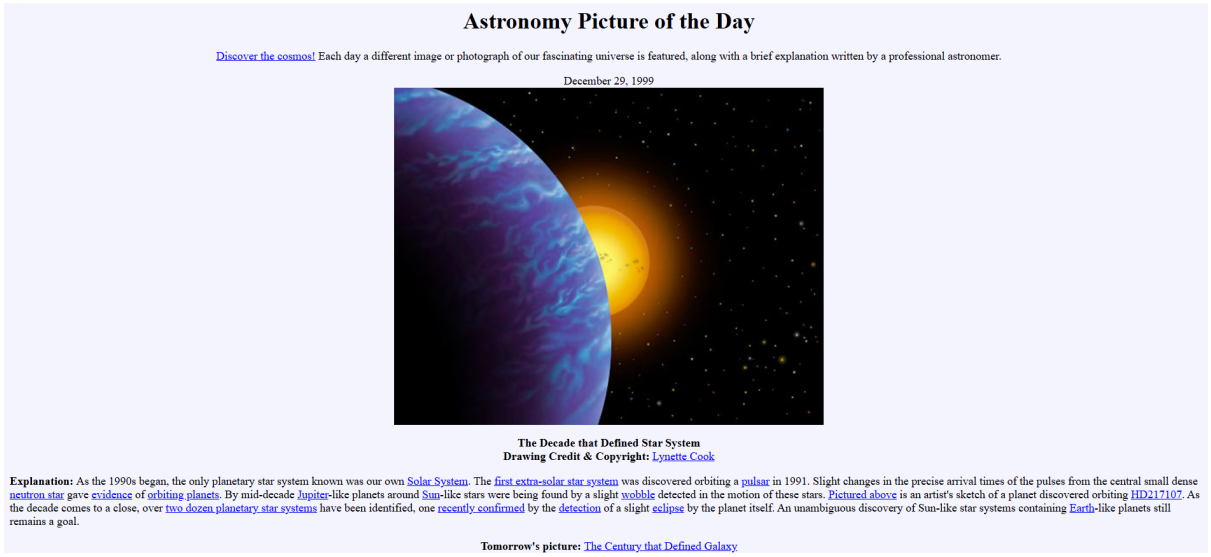



Figure 1: Discover Page

# Original NASA APOD Page



**Astronomy Picture of the Day**

Discover the cosmos! Each day a different image or photograph of our fascinating universe is featured, along with a brief explanation written by a professional astronomer.

December 29, 1999

**The Decade that Defined Star System**
**Drawing Credit & Copyright:** Lynette Cook

**Explanation:** As the 1990s began, the only planetary star system known was our own Solar System. The first extra-solar star system was discovered orbiting a pulsar in 1991. Slight changes in the precise arrival times of the pulses from the central small dense neutron star gave evidence of orbiting planets. By mid-decade Jupiter-like planets around Sun-like stars were being found by a slight wobble detected in the motion of these stars. Pictured above is an artist's sketch of a planet discovered orbiting HD217107. As the decade comes to a close, over two dozen planetary star systems have been identified, one recently confirmed by the detection of a slight eclipse by the planet itself. An unambiguous discovery of Sun-like star systems containing Earth-like planets still remains a goal.

**Tomorrow's picture:** The Century that Defined Galaxy

Figure 2: NASA APOD Source Page

# My Stellarium Page


Figure 3: Stellarium Favourites Gallery

# Account Settings Page

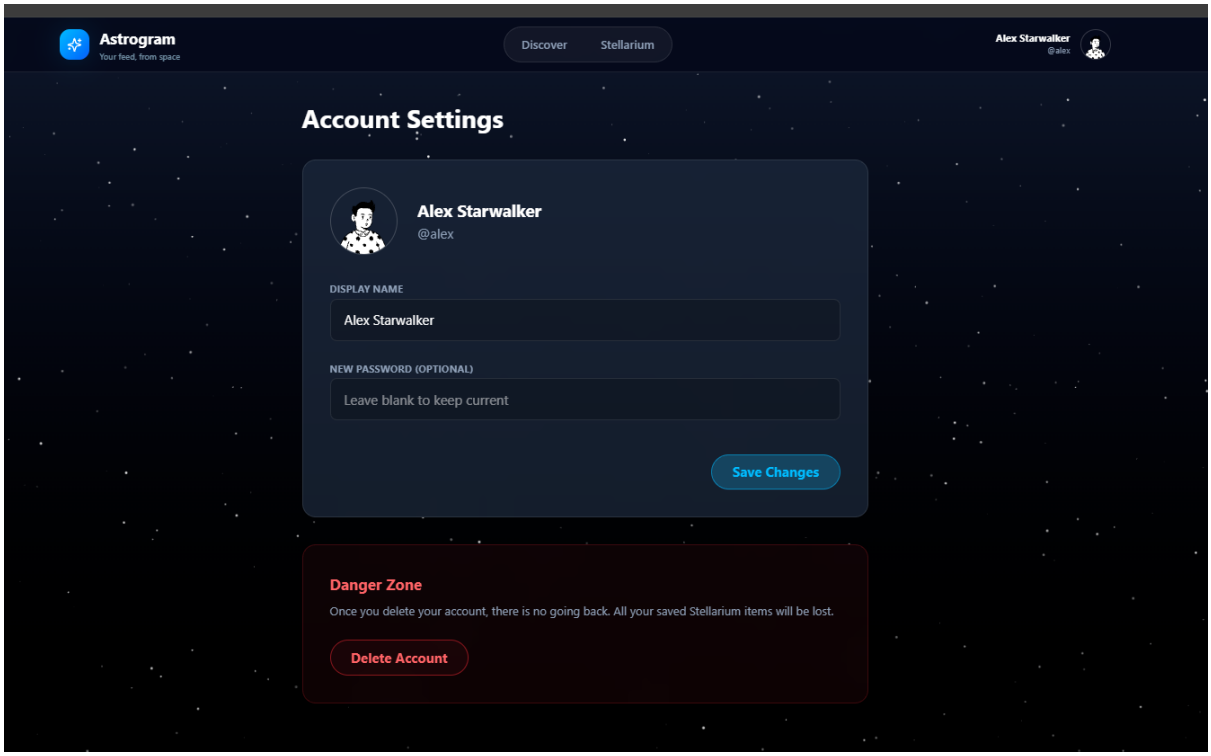



Figure 3: Account Settings