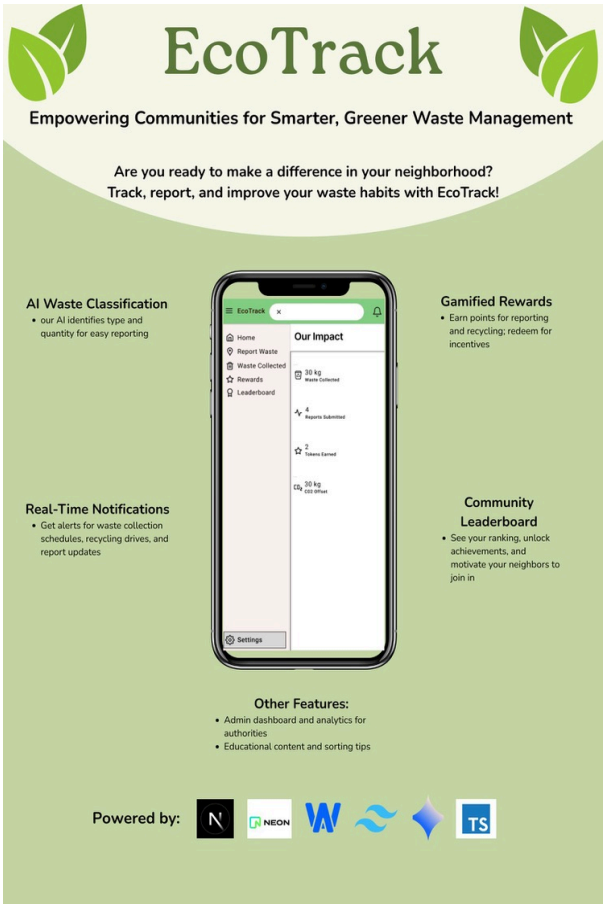
EcoTrack (NUS Orbital Project)

EcoTrack, an AI-powered waste management platform.

Proposed level of achievement: Apollo 11

Promotional Posters



Proof-of-Concept :

https://github.com/user-attachments/assets/7405a2f0-2817-4729-8379-8a0e70848eff

Before using the app

Limitations

🔍 1. AI Accuracy & Dependence
• Limitation: Reliance on Google Gemini AI for waste classification may lead to incorrect or biased predictions, especially for unclear images.
• Impact: Users may get inaccurate rewards or feedback, reducing trust in the system.

⚖️ 2. Verification Bottleneck
• Limitation: Waste report verifications may require human validation in some cases, especially for edge scenarios.
• Impact: Slows down the reward system and notification flow, reducing real-time effectiveness.

🔒 3. Web3Auth Adoption Barrier
• Limitation: Not all users (especially older or less tech-savvy ones) are comfortable with Web3 wallet-based authentication.
• Impact: Limits app accessibility and onboarding rate among the general population.

Milestone 2 (Prototyping)

Motivation

Singapore faces growing challenges in waste management due to rising urbanization and consumption.
Common issues include:

- Low Recycling Rates: Many residents are unsure about proper recycling practices, leading to contamination and reduced recycling efficiency.
- Inefficient Collection: Waste bins often overflow or are collected when not full, resulting in resource wastage.
- Lack of Engagement: Residents have limited visibility into their waste habits and few incentives to improve them.

EcoTrack, an AI-powered waste management platform aims to address these pain points by empowering both residents and municipal authorities with real-time data, actionable insights, and user-friendly tools to promote responsible waste disposal and recycling. It is designed to incentivize and streamline waste reporting and collection. Our goal is to create a community-driven approach to waste management, rewarding users for their eco-friendly actions.

Aim

We aim to allow residents to track their waste disposal and recycling habits, provide timely reminders and educational content to encourage correct recycling. At the same time, we also aim to enable authorities to monitor bin fill levels and optimize collection routes and engage users through gamification and community leaderboards to foster sustainable habits. EcoTrack will empower Singapore's residents and authorities to collaboratively improve waste management practices. By combining education, data-driven insights, and gamification, our app aims to make sustainability a daily habit for all.

User Stories

As a resident who wants to contribute to a cleaner neighborhood, I want to easily report overflowing or illegal waste via the app, so that authorities can respond quickly and efficiently.

As a resident who recycles regularly, I want to track my recycling habits and see my progress, so I can stay motivated and improve my environmental impact.

As a busy user, I want to receive timely notifications about waste collection schedules and recycling drives, so I never miss important dates or opportunities to participate.

As a waste collector, I want to view optimized collection routes and real-time bin fill levels, so I can make my rounds more efficiently and avoid unnecessary trips.

As a municipal officer, I want to access analytics and reports on waste generation and recycling rates, so I can make informed decisions about resource allocation and public outreach.

As a community leader, I want to organize local clean-up events and track participation through the app, so I can foster greater community involvement.

As a user with limited technical skills, I want the app to have an intuitive interface and clear instructions, so I can be incentivised to use the app.

As a resident who sometimes forgets to sort waste properly, I want the app to provide educational content and AI-powered waste classification, so I can learn and improve my sorting habits.

Features

1. User Acount Authentication [Completed]

Description

As EcoTrack is a personalized waste management and rewards platform, each user must be authenticated with a unique account.
EcoTrack implements a Web3 wallet-based authentication system using Web3Auth, allowing users to:

- Log in using a Web3 wallet (e.g., Google, Facebook, or other OAuth providers via Web3Auth)
- Create an account automatically on first login (no manual registration required)
- Securely store user identity and session using cryptographic signatures
- Log out and clear session data

Upon successful authentication, the user's information (email, name, wallet address) is retrieved and stored in the database if not already present. All authentication and user management logic is handled via the Web3Auth SDK and backend API calls.

Implementation Philosophy

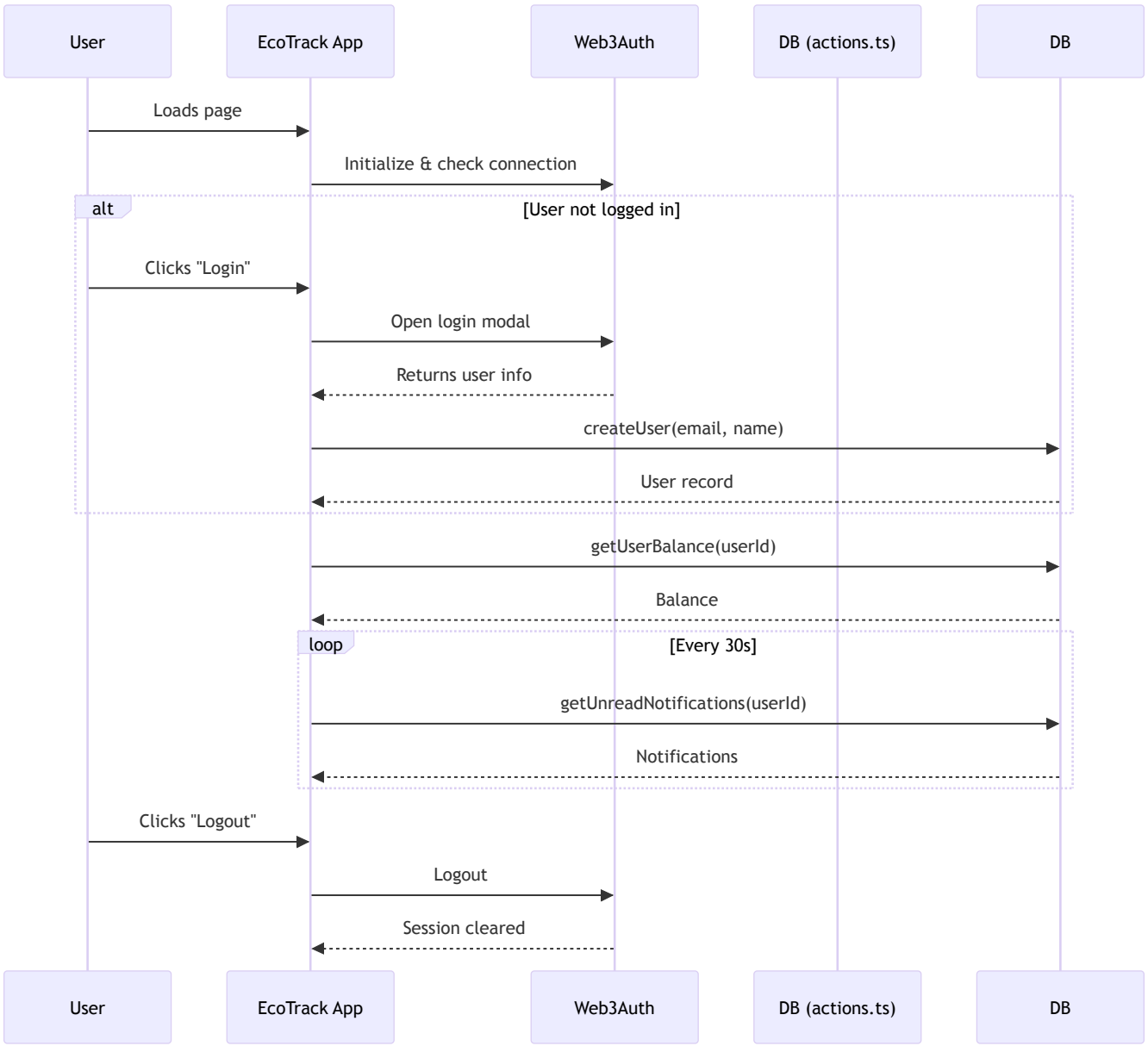
Implementation Philosophy
EcoTrack’s authentication flow is designed for seamless onboarding and security:

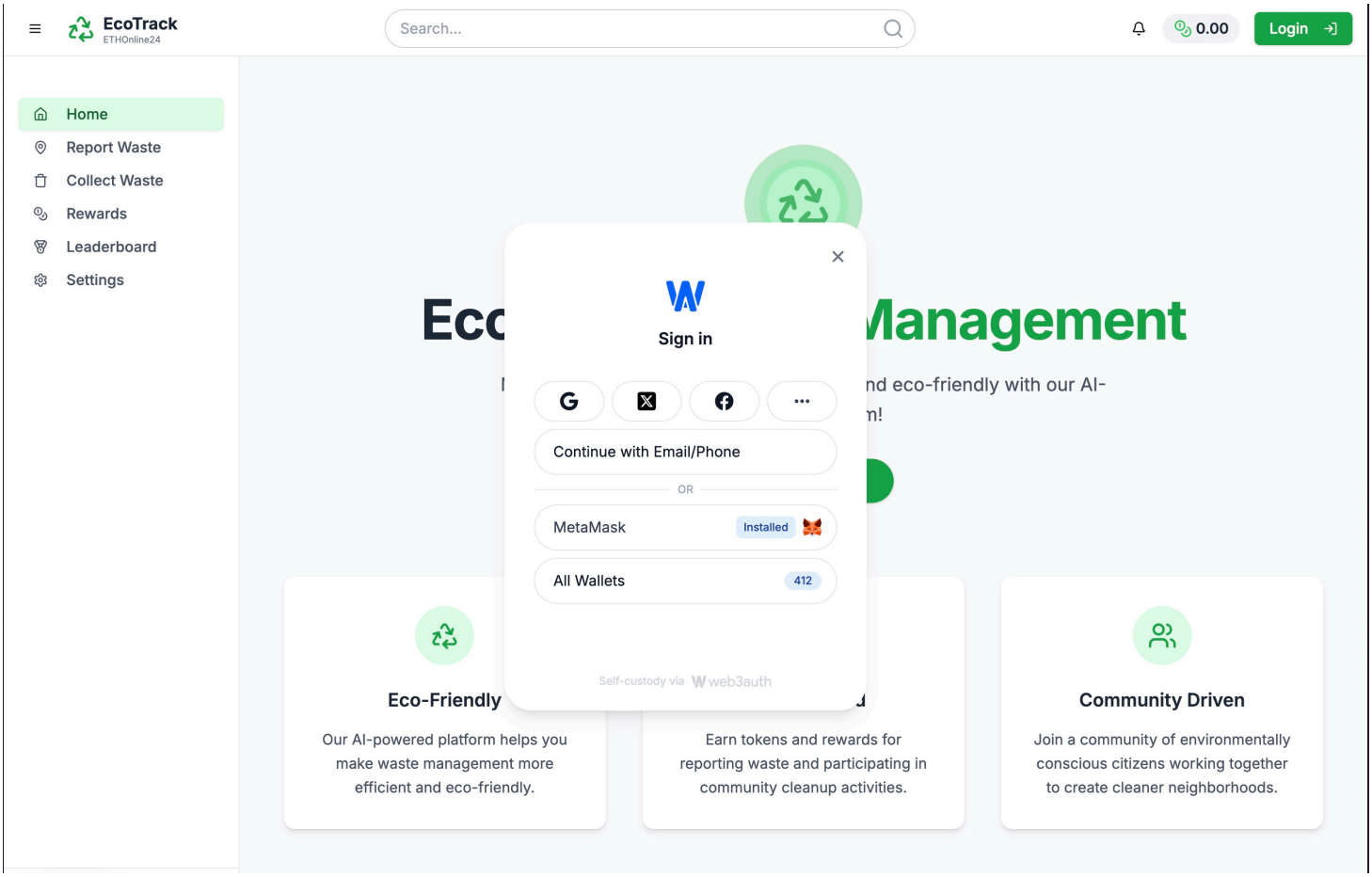
- On app load, the authentication controller checks if the user is already logged in via Web3Auth.
- If the user is authenticated, their profile is fetched (or created if new) and stored in the app state.
- If not authenticated, the user is prompted to log in via Web3Auth’s modal, supporting multiple OAuth providers and wallets.
- After login, the app ensures a user profile exists in the database (creating one if needed).
- The user is then routed to the main dashboard.
- All authentication errors (e.g., network mismatch, failed login) are caught and displayed to the user via toast notifications.
- The authentication state is monitored, and users are redirected to login if their session expires.
- This approach ensures that every user has a unique, secure account, and onboarding is as frictionless as possible, leveraging modern Web3 standards.

Implementation Challenges

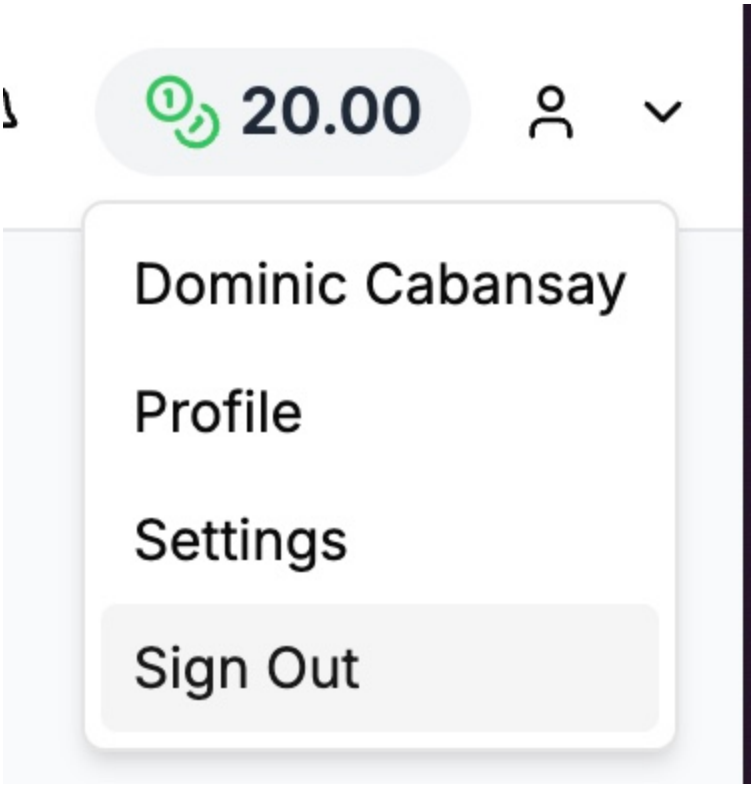
For user authentication, the key issue we faced was network mismatch and configuration. Enusring the Web3Auth client ID and network configuartion match the project's setting was quite difficult and we kept facing usses where the wallet was not being initialised, leading to failed logins. Additionally, handling session persistence and automatic re-authentication required careful integration with Web3Auth's SDK and local storage.

Diagrams




EcoTrack
ETHOnline24
Search...
0.00
Login
Home
Report Waste
Collect Waste
Rewards
Leaderboard
Settings
Sign in
Continue with Email/Phone
OR
MetaMask
Installed
All Wallets
412
Self-custody via web3auth
Eco-Friendly
Our AI-powered platform helps you make waste management more efficient and eco-friendly.
Earn tokens and rewards for reporting waste and participating in community cleanup activities.
Community Driven
Join a community of environmentally conscious citizens working together to create cleaner neighborhoods.

Login Modal


20.00
Dominic Cabansay
Profile
Settings
Sign Out

Sign Out

2. Waste Reporting and Image Upload

Description

EcoTrack allows users to report waste incidents by uploading images and specifying where the waste was found. The reporting process is designed to be straightforward:

- Users can upload a photo of the waste (from their device or by taking a new photo).
- Users provide the location where the waste was found, either by entering an address.
- Each report includes the image, location, and the waste type (after AI verification), and is stored in the database with a status (e.g., pending, collected).
- Upon successful submission, users receive confirmation and may be awarded points as part of the rewards system.

All image uploads and report submissions are handled via the Next.js frontend, with backend logic in `/utils/db/actions.ts` and API routes for database integration.

Implementation Philosophy

The waste reporting and image upload feature is designed for simplicity and accessibility:

- Users only need to provide an image and location, lowering the barrier to reporting.
- The UI guides users through uploading an image and specifying the location in a few easy steps.
- All reports are validated before submission, ensuring required fields (image, location) are present.
- Users receive immediate feedback on the status of their report submission via toast notifications.
- The reporting logic is integrated with the rewards and notification systems, so users are incentivized and kept informed.

The frontend checks for user authentication before allowing report submission. If the user is not logged in, they are prompted to authenticate first. The backend ensures that each report is associated with a valid user and that all required data is stored securely.

Implementation Challenges

As this section was one of the first features implemented in EcoTrack, we encountered a variety of setbacks and technical challenges throughout the development process.

The first major challenge was designing a suitable data structure for storing waste reports and handling image uploads. Since EcoTrack uses Neon as its database, we needed to ensure that each report could efficiently reference its associated image, user, and location data. As we did not use any cloud storage service, images were stored directly as base64-encoded strings within the database itself. This approach simplified the architecture but required careful consideration of database performance and storage limits.

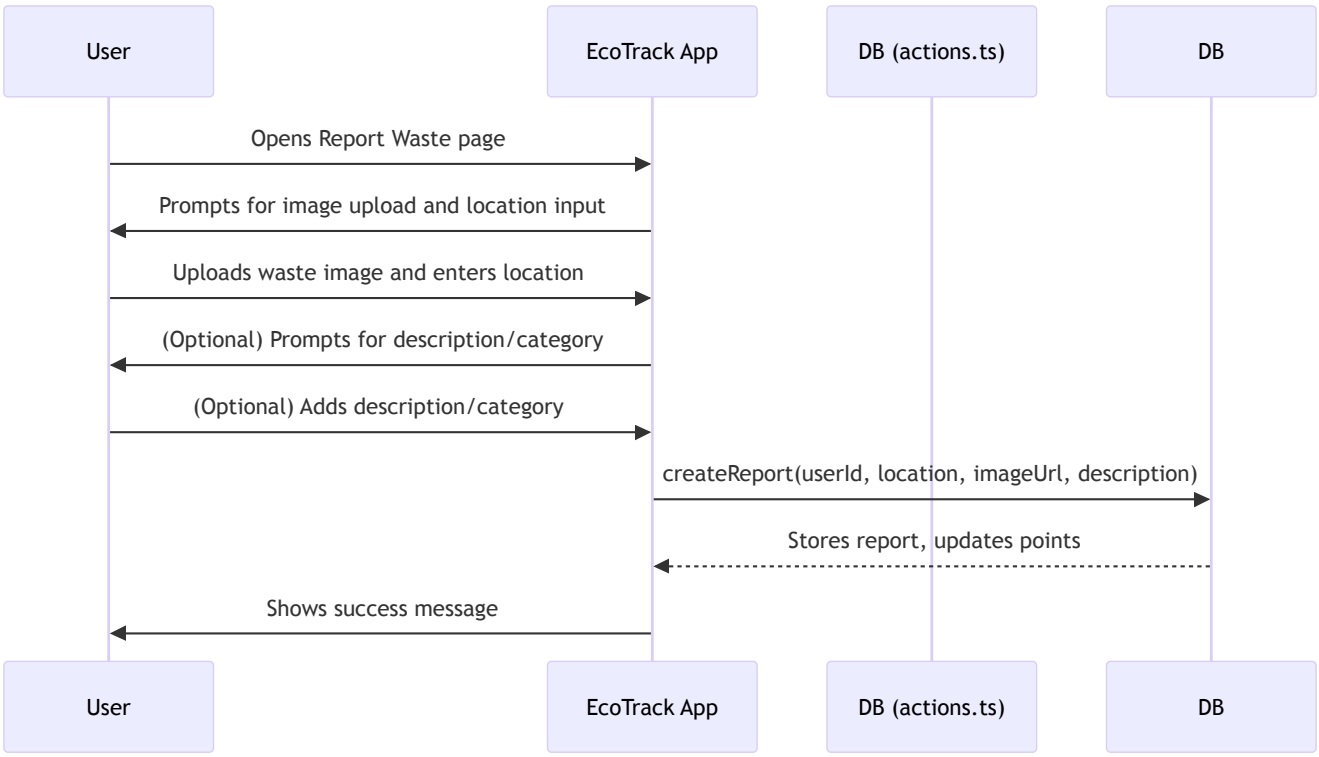
Another significant issue was handling the asynchronous nature of image uploads and database writes. There were cases where the image upload would succeed but the database write would fail (or vice versa), leading to orphaned images or incomplete reports. To address this, we implemented atomic operations and transaction-like logic in our backend API. Only after a successful image upload would the report be created in the database, and any failures would trigger cleanup routines to remove unused images.

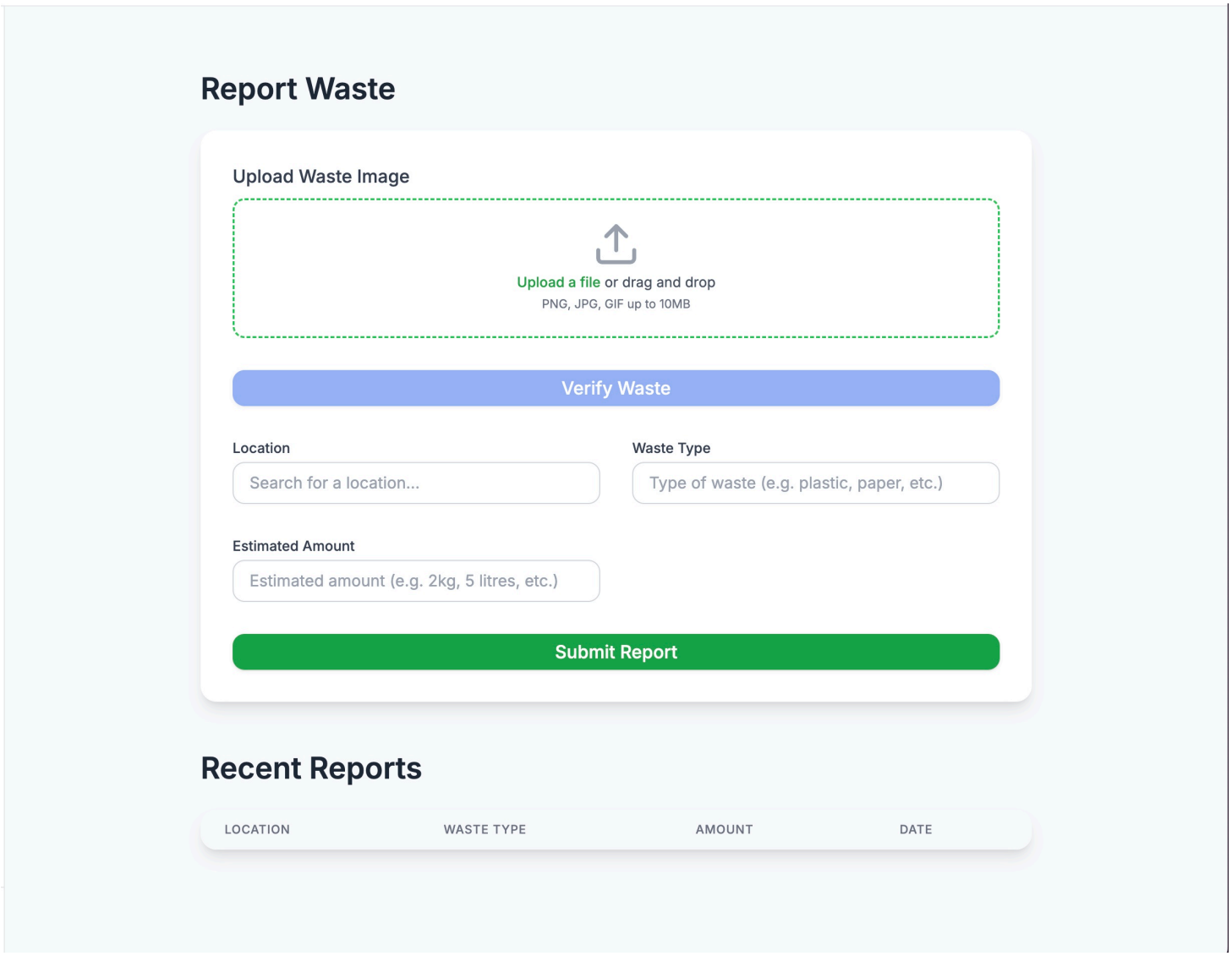
Managing location data also presented challenges. Some users denied location permissions, while others entered ambiguous or incomplete addresses. To overcome this, we provided multiple options for location input: users could use GPS or manually enter an address.

State management on the frontend was another hurdle, especially when users navigated away from the reporting page or experienced network interruptions. We initially faced issues where partially completed reports would be lost, or duplicate submissions would occur if users retried after a failure. To solve this, we implemented temporary local state caching and ensured that all required fields were validated before allowing submission.

Finally, authentication and access control required careful handling. We needed to ensure that only authenticated users could submit reports, but also wanted to maintain a smooth user experience. Our solution was to check authentication status before allowing access to the reporting page, and to redirect unauthenticated users to the login flow, preserving their progress so they could resume reporting after logging in.

Diagrams


User
EcoTrack App
DB (actions.ts)
DB
Opens Report Waste page
Prompts for image upload and location input
Uploads waste image and enters location
(Optional) Prompts for description/category
(Optional) Adds description/category
createReport(userId, location, imageUrl, description)
Stores report, updates points
Shows success message
User
EcoTrack App
DB (actions.ts)
DB



Report Waste Page

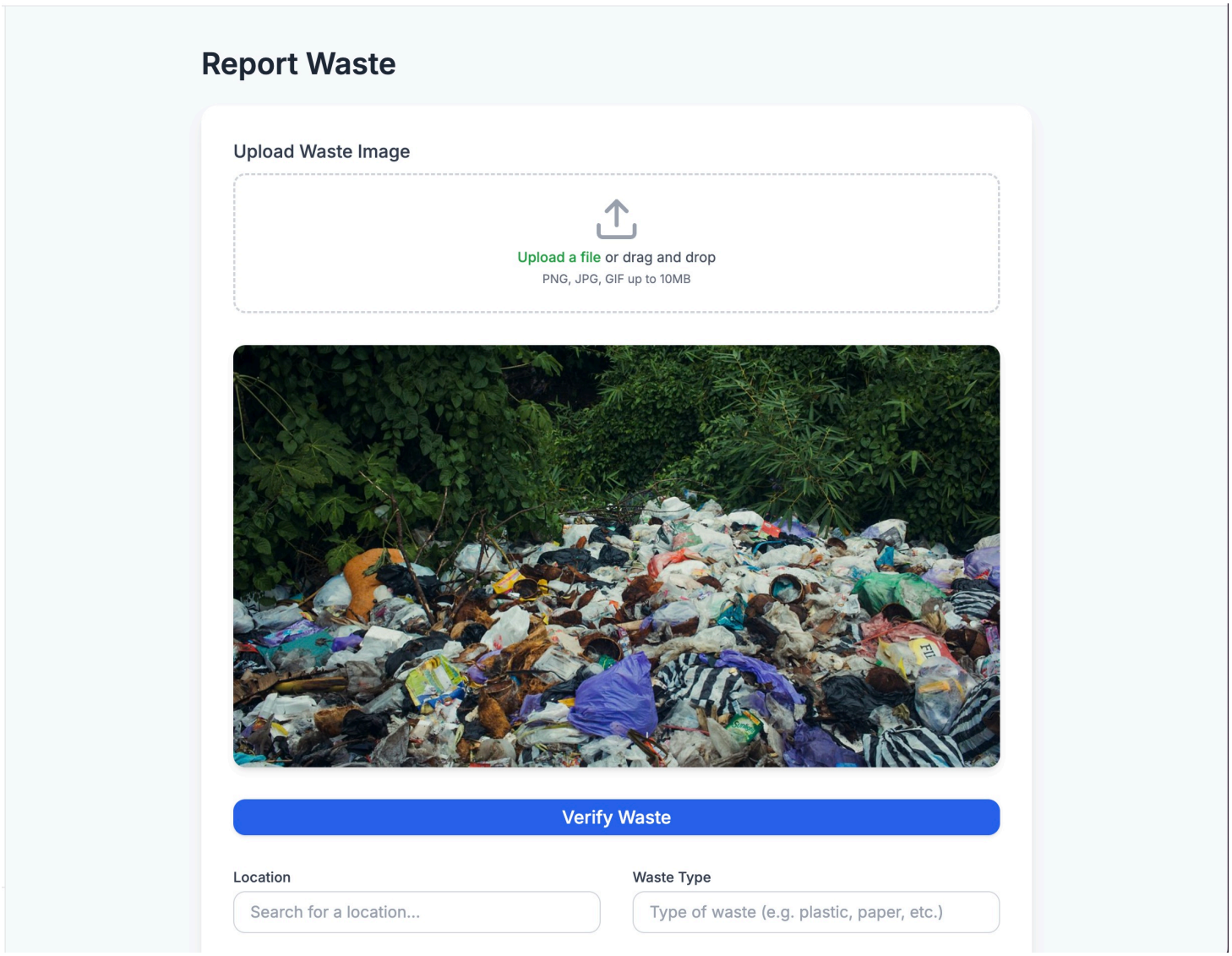
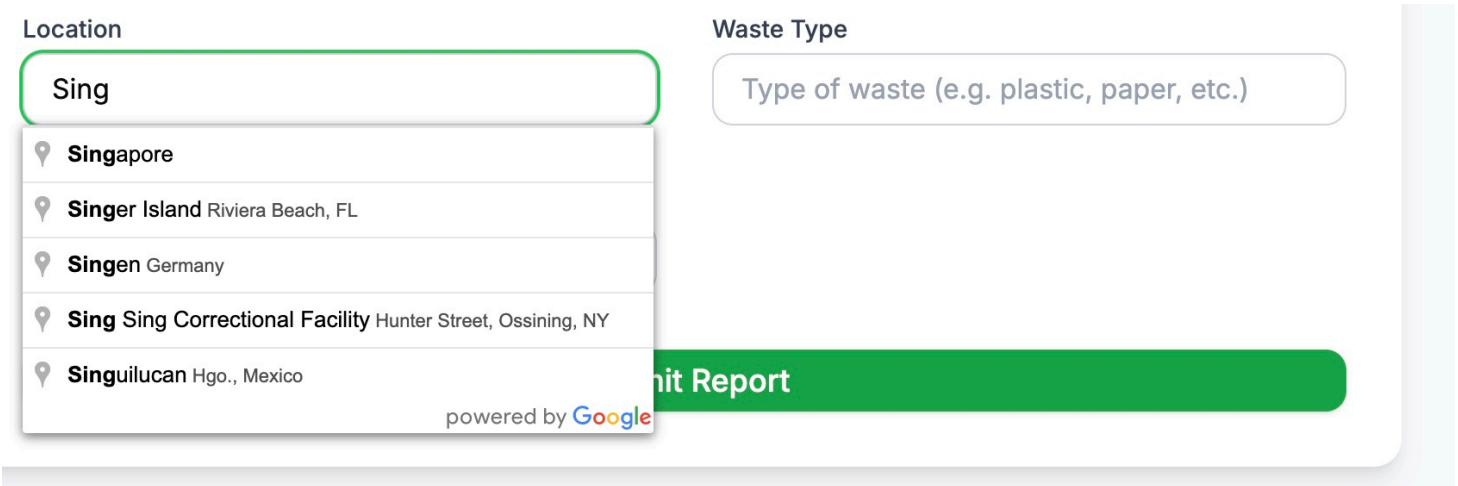


Image Upload



Location Input

3. AI Verification of Waste Type

Description

EcoTrack leverages Google Gemini AI to automatically verify the type and estimated amount of waste from user-uploaded images. This feature streamlines the reporting process, reduces manual input, and improves data accuracy by providing AI-generated waste classifications and quantity estimates.

- Users upload a photo of the waste.
- Upon clicking "Verify Waste," the image is sent to Gemini AI for analysis.
- Gemini AI returns a JSON object containing the predicted waste type (e.g., plastic, paper), estimated quantity (e.g., 2kg, 5 litres), and a confidence score.
- The results are displayed to the user for review before final submission.
- The verified data is then included in the waste report stored in the database.

Implementation Philosophy

The AI verification feature is designed to minimize user effort and maximize data reliability:

- **Prompt Engineering:**
  The prompt sent to Gemini AI is crafted to elicit structured, actionable responses. It instructs the AI to act as a waste management expert and to analyze the image for three key outputs: waste type, approximate quantity, and confidence level. The prompt explicitly requests a JSON response in the following format:

```
{
  "wasteType": "type of waste",
  "quantity": "approximate quantity",
  "confidence": confidence level (0–100)
}
```

This ensures the AI's output is easily parsed and directly usable in the app.

- The uploaded image is converted to a base64 string and sent as inline data to Gemini AI, ensuring compatibility and security.
- The verification process is triggered by a single button click. Users receive real-time feedback on the verification status (verifying, success, or failure) and can review the AI's results before

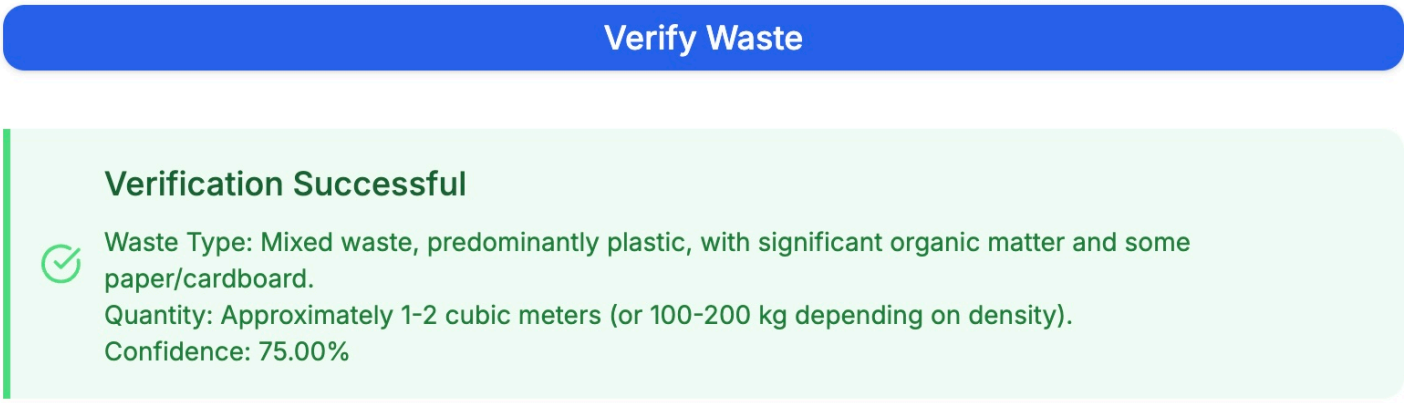
submitting their report.

- Only verified results (with all required fields present) are accepted for submission. If the AI response is malformed or missing data, the user is prompted to retry.
- The AI verification is tightly integrated with the reporting workflow, ensuring that only AI-verified data is stored in the database.

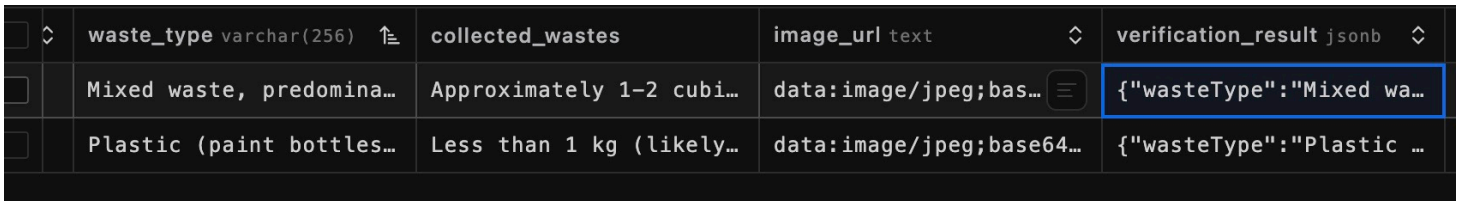
Implementation Challenges

Gemini AI sometimes returned responses with extra formatting (such as Markdown code blocks) or incomplete JSON, which caused parsing errors and disrupted the user flow. We tackled this by iteratively refining the prompt to be as explicit as possible, instructing the AI to respond only with a JSON object and nothing else. On the application side, we extracted only the JSON content. This included handling edge cases where the AI might return code fences or additional commentary.

Diagrams



Successful Waste Verification



waste_type varchar(256)	collected_wastes	image_url text	verification_result jsonb
Mixed waste, predomina…	Approximately 1–2 cubi…	data:image/jpeg;bas…	{"wasteType":"Mixed wa…
Plastic (paint bottles…	Less than 1 kg (likely…	data:image/jpeg;base64…	{"wasteType":"Plastic …

Waste Type stored Neon Database

4. Rewards

Description

Implementation Philosophy

Implementation Challenges

Diagrams

5. Waste Collection

Description

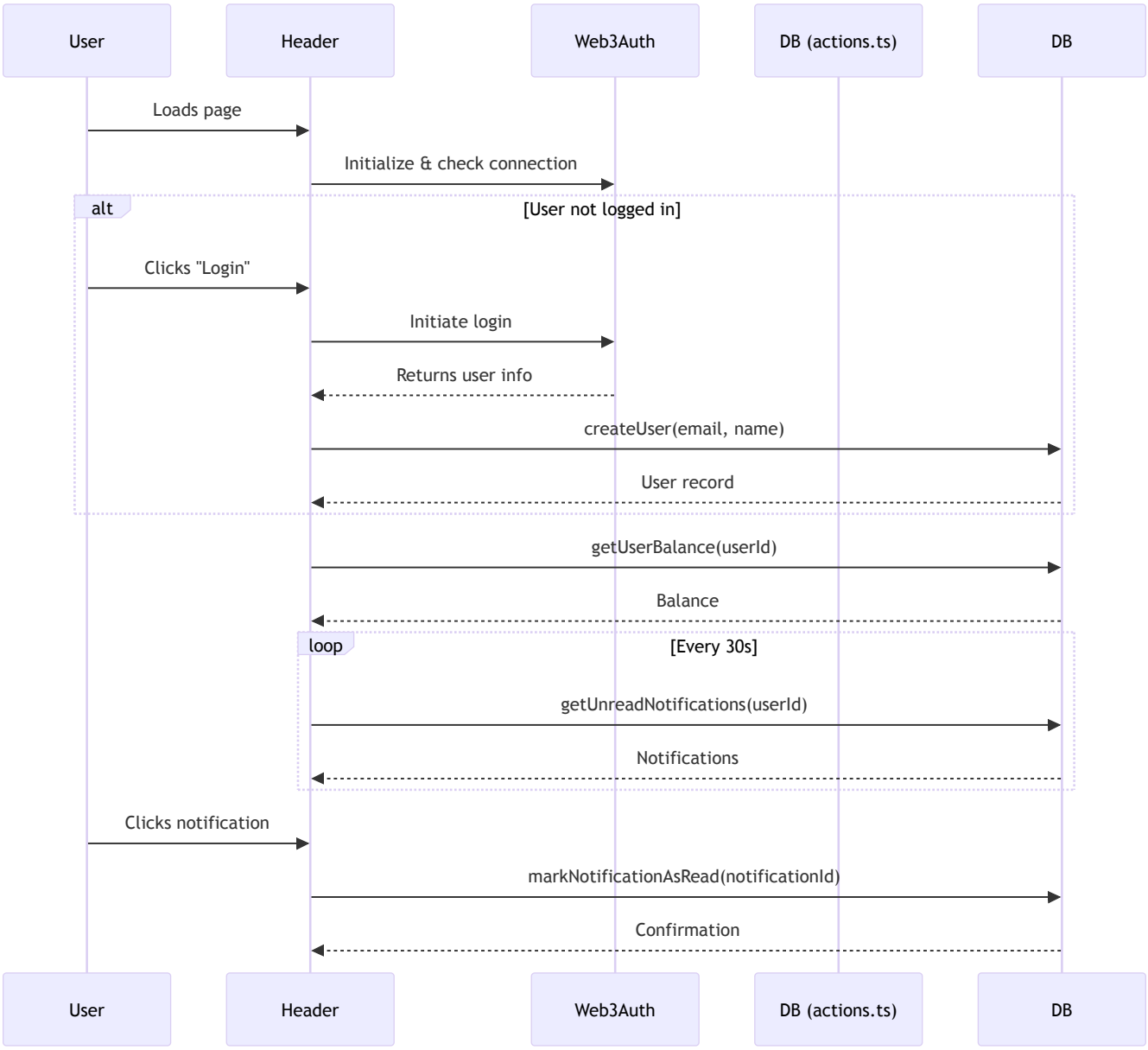
Implementation Philosophy

Implementation Challenges

Diagrams

Unified Modelling Language (UML) Diagrams

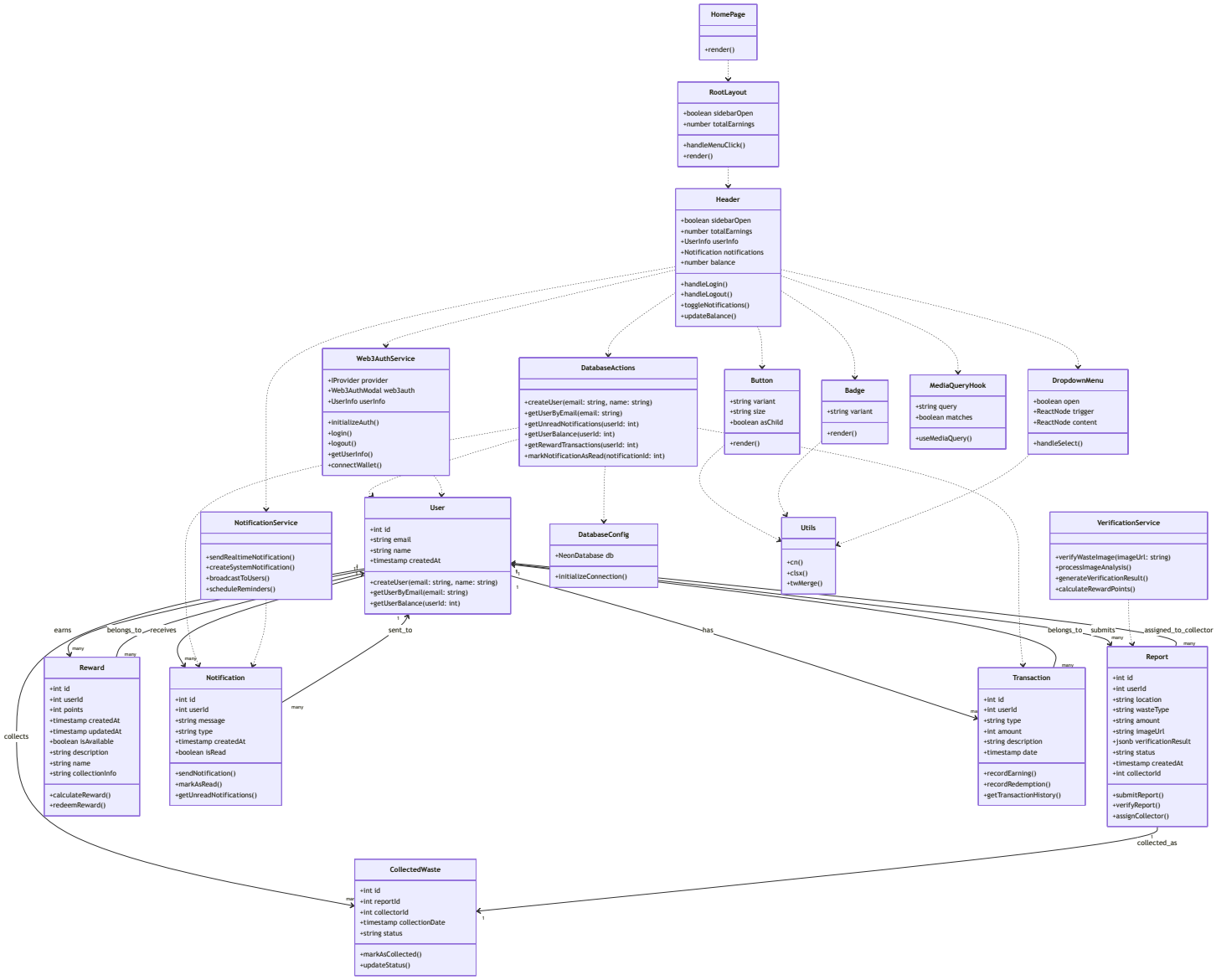
Sequence Diagram



Class Diagram

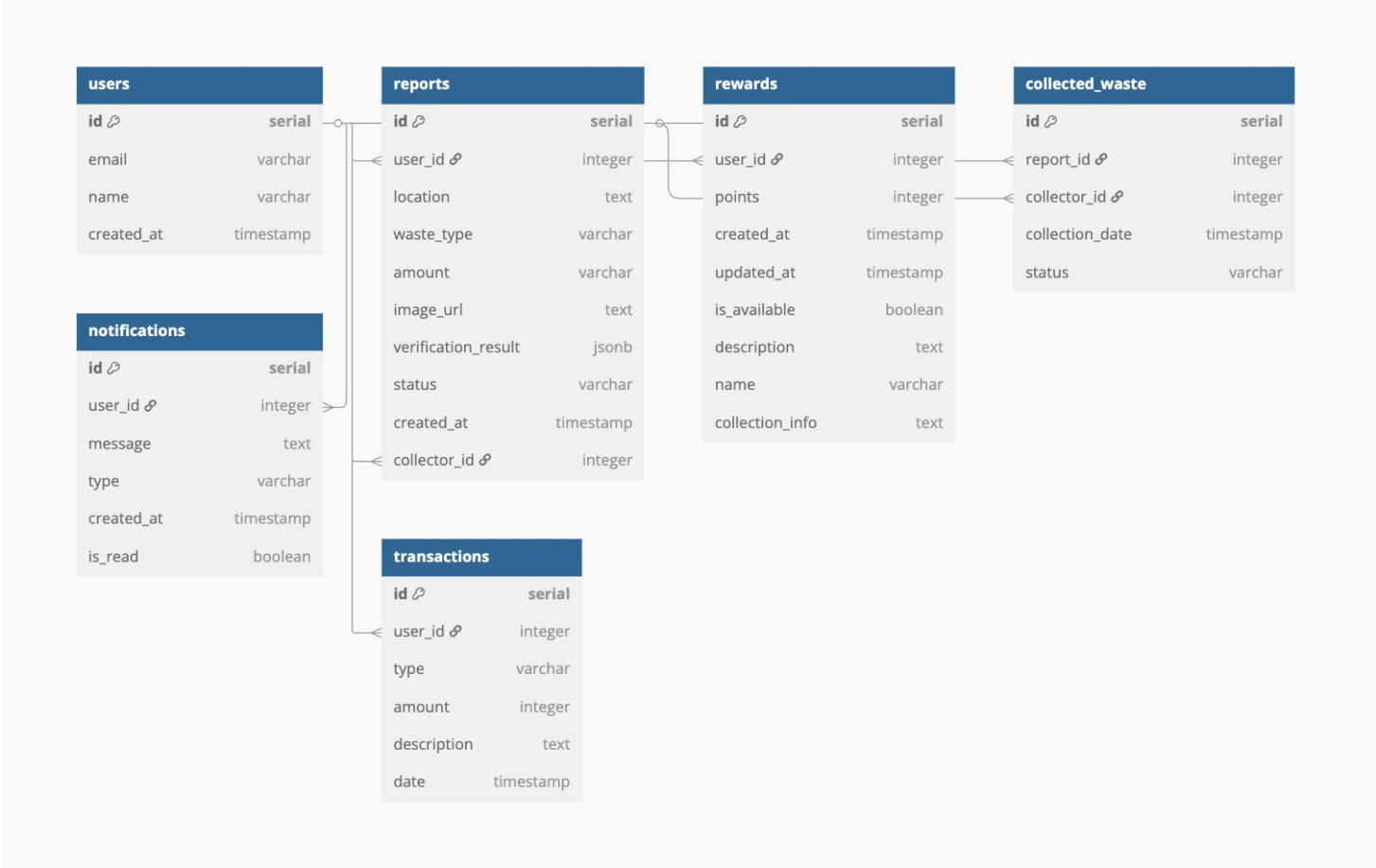
The class diagram below illustrates the core structure of EcoTrack's system design, following the Model-View-Controller (MVC) pattern within a layered architecture. At the heart of the system are the core database entities—User, Report, CollectedWaste, Reward, Notification, and Transaction—which represent the main data models (Model layer) and encapsulate both data and related business logic. The business logic services, such as DatabaseActions, Web3AuthService, VerificationService, and NotificationService, act as controllers and service layers, orchestrating interactions between the

models and handling application workflows like authentication, waste verification, notification delivery, and database operations. The UI components—including Header, Button, Badge, DropdownMenu, RootLayout, and HomePage—constitute the View layer, responsible for rendering the user interface and managing user interactions. Utility classes like MediaQueryHook, DatabaseConfig, and Utils provide supporting functionality for both the UI and backend logic. Relationships between classes are clearly defined: for example, a User can submit multiple Reports, earn Rewards, receive Notifications, and have Transactions, while Reports are linked to CollectedWaste and assigned collectors. Service dependencies and UI component dependencies are also mapped, showing how controllers and views interact with models and utilities. This design enforces separation of concerns, modularity, and scalability, with each layer and component having a well-defined responsibility, thus exemplifying the MVC pattern and supporting maintainable, testable code.



Entity Relationship Diagram (ERD)

EcoTrack's ERD (Entity Relationship Diagram) models a system for tracking waste reports, user activities, and rewards. The core entity is users, who can submit reports about waste, specifying details like location, type, and amount. Each report can be linked to a collector (also a user) and may result in a collected_waste record, tracking collection status and date. Users earn rewards based on their activities, with points and availability status, and all transactions (like earning or redeeming points) are logged in the transactions table. The system also sends notifications to users about relevant events, ensuring engagement and transparency throughout the waste management process.

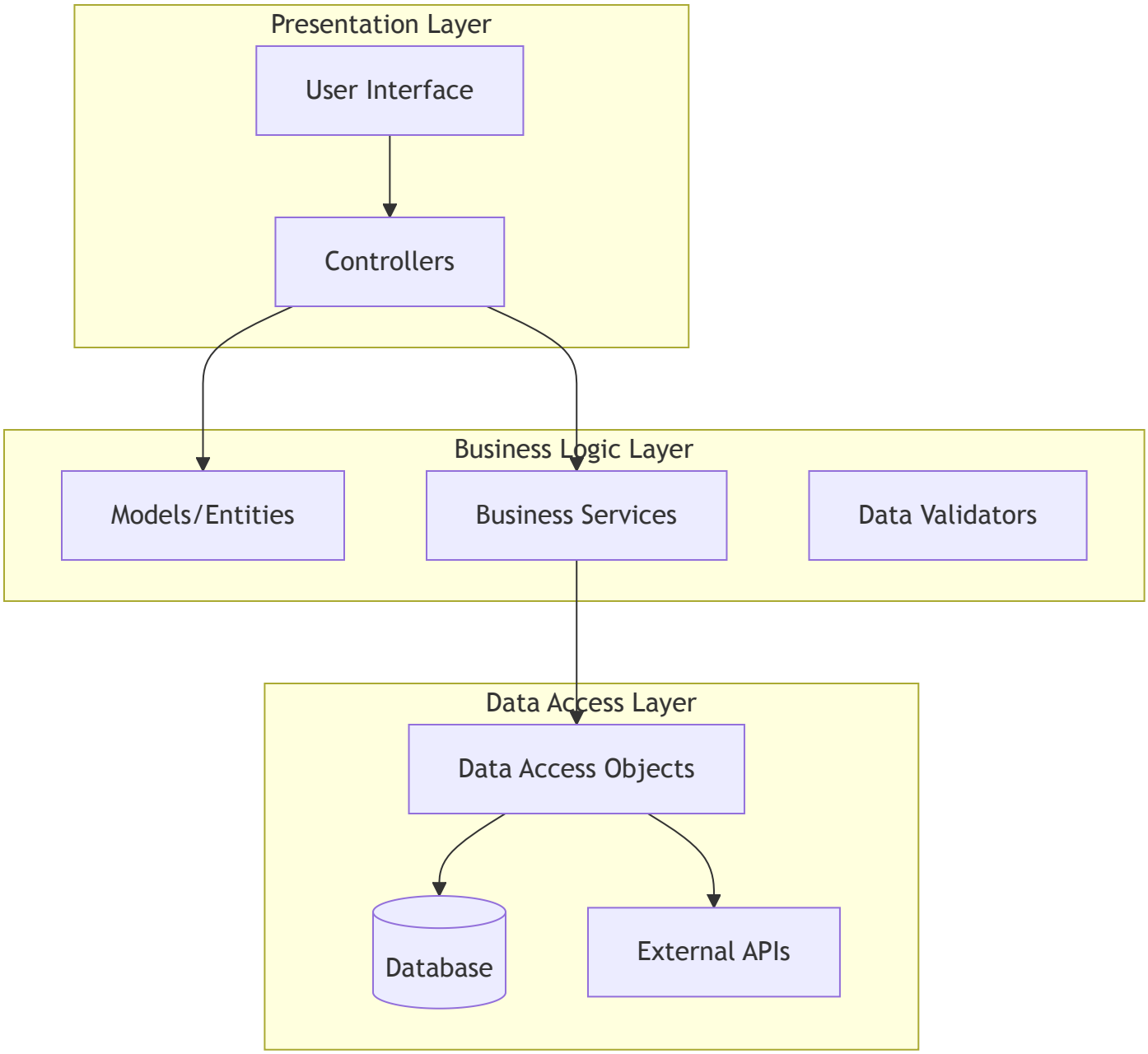


Software Engineering Practices

Primary Architecture: N-tier Architecture with MVC Pattern

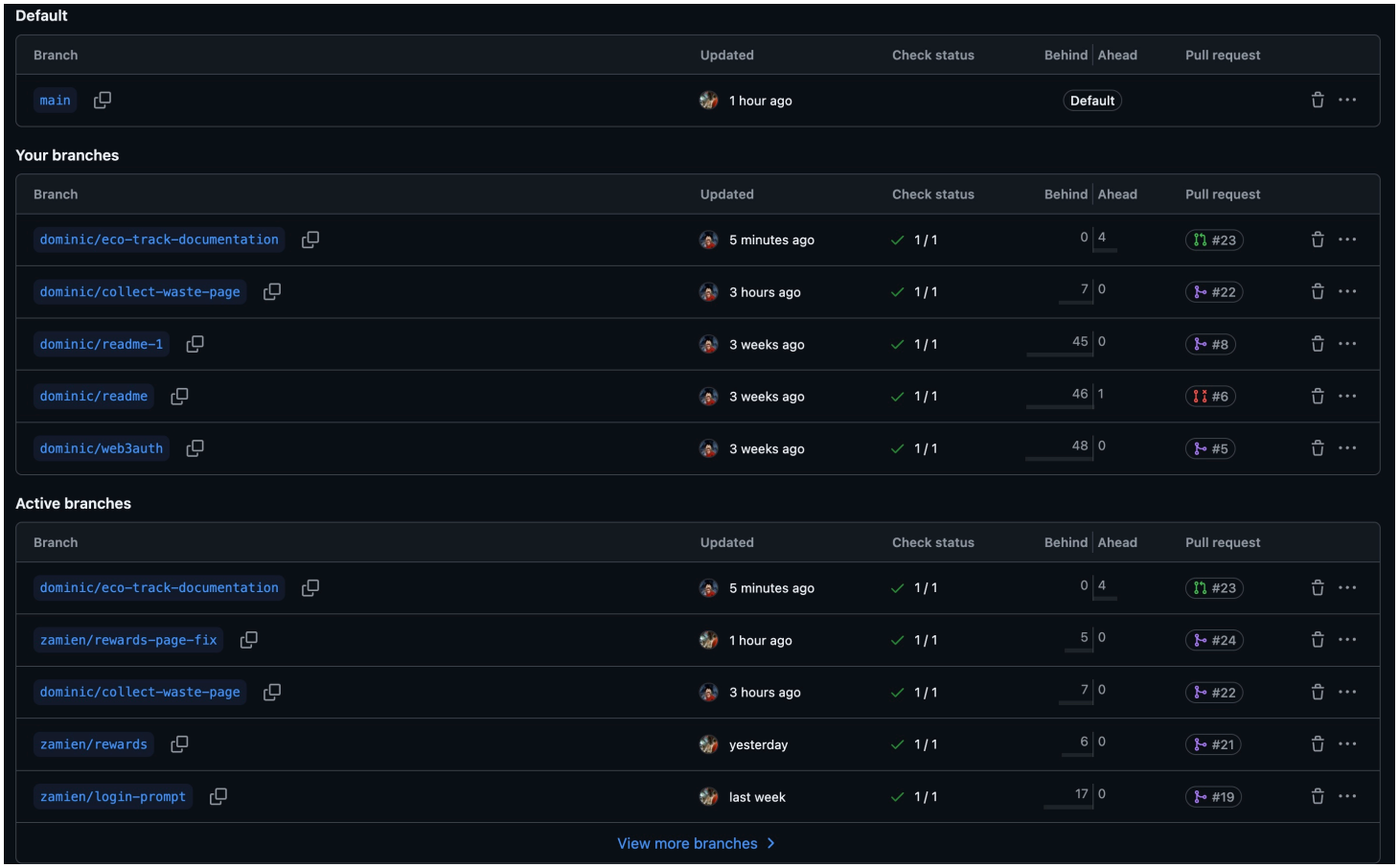
For EcoTrack (an environmental tracking application), we aim to implement a **3-tier layered architecture** combined with the **Model-View-Controller (MVC)** pattern. This is because EcoTrack adopts an N-tier Architecture with the MVC Pattern to achieve a clear separation of concerns, scalability, and maintainability in its environmental tracking application. By dividing the system into

three main layers—Presentation, Business Logic, and Data Access—each layer can focus on a specific responsibility: the Presentation Layer manages user interactions and UI rendering, the Business Logic Layer handles core application rules and data validation, and the Data Access Layer manages communication with databases and external APIs. Integrating the MVC pattern within this structure further organizes the code by separating data models, user interface views, and controllers that handle user input and application flow. This approach not only makes the codebase easier to test and debug but also allows teams to work on different layers independently, supports future enhancements, and ensures that changes in one layer have minimal impact on others, which is essential for a robust, evolving application like EcoTrack.



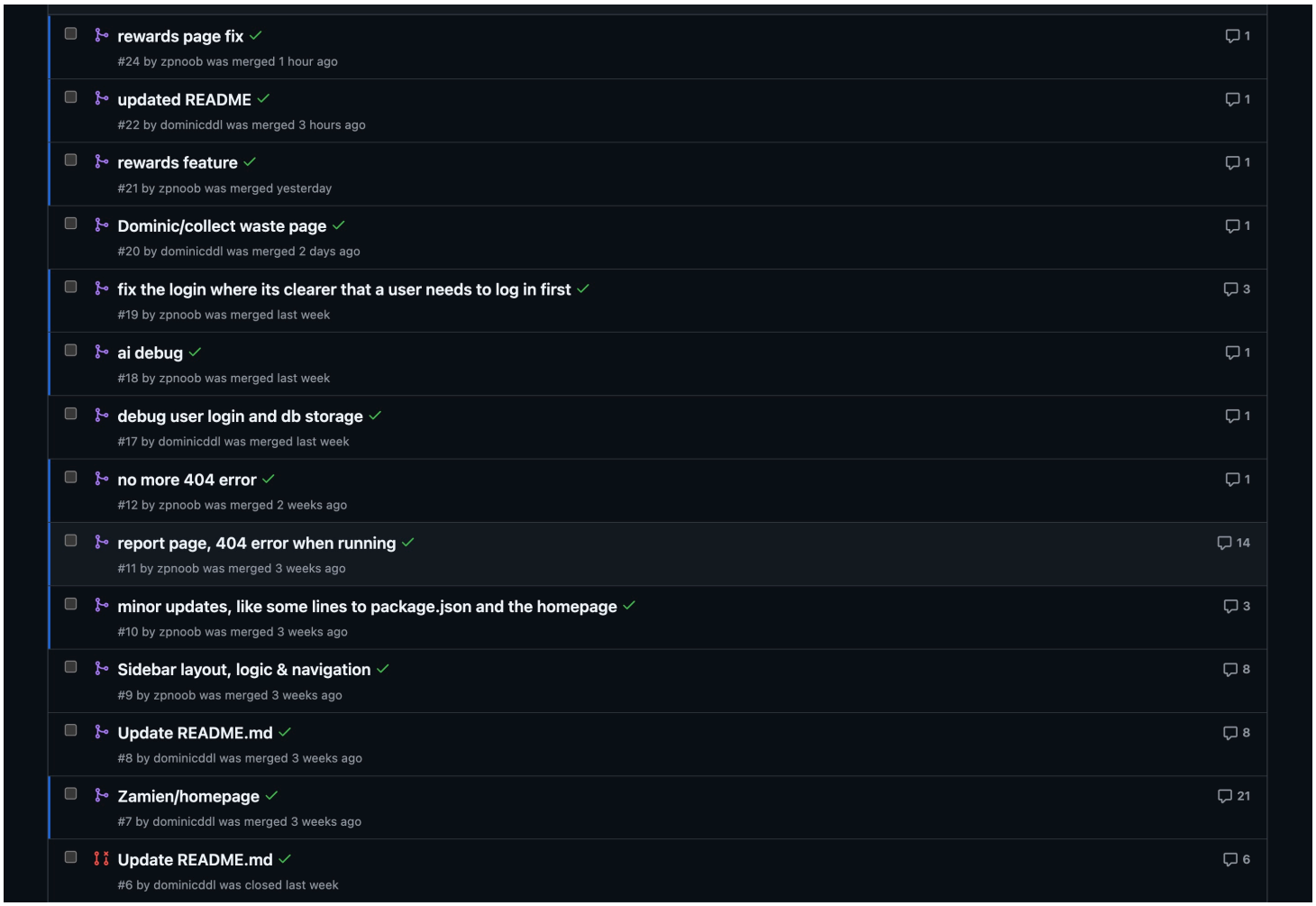
Version Control

Branching



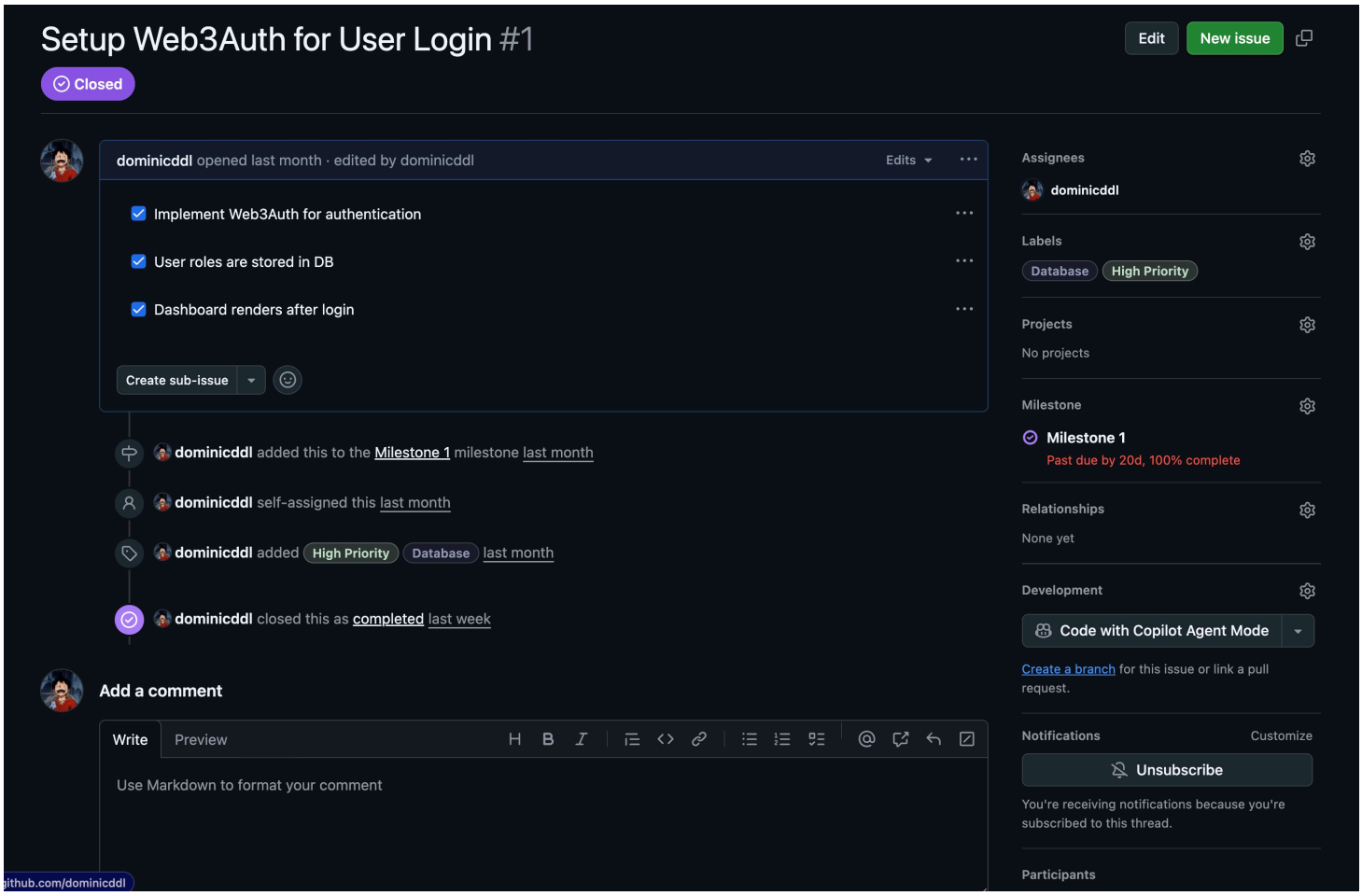
We use Git for version control and for facilitating the management of the code. The remote master branch will always have a working set of code at any point in time. To work on new features, we pull from the master branch and branch off to a feature branch with the feature-developer as the naming convention. Working on separate branches prevents cross-contamination of code which complicates the debugging process. Upon completion of the feature, the feature branch is committed and a pull request is performed to merge the new changes into the master branch.

Pull Requests



The pull request feature of Git is utilised when updating the remote master branch. The developer responsible for the feature creates a pull request with the partner assigned as the code reviewer. This enforces communication and prevents incorrect resolution of merge conflicts which introduces unnecessary bugs. Only after the code is reviewed, approved and merge conflicts are resolved will the pull request be successfully completed.

Issues



Git Issues were used to keep track of any open issues or existing bugs within the app. Tags were also included to help identify the type of issue be it fixes or potential enhancements. Only when the problem has been resolved will the issue be closed.

Quality Control

Automated Testing

User Testing

Timeline and Development Plan

17 EcoTrack Development Timeline & Milestones

Milestone	Due Date	Phase	Deliverables	Status	Features
Milestone 1	**June 2, 2024**	**Technical Proof of Concept**	Minimal Working System	**COMPLETED**	• Web3Auth user authentication • Basic home page & dashboard • Waste reporting with image upload • Database integration (PostgreSQL + Drizzle) • Responsive UI components
Milestone 2	**June 30, 2024**	**Core Prototype**	Working System with Core Features	**NOT COMPLETED**	• AI-powered waste verification • Rewards points system • Real-time push notifications • User balance tracking • Report status management
Milestone 3	**July 28, 2024**	**Extended System**	Full-Featured Application	**NOT COMPLETED**	• Interactive leaderboard & achievements • Admin dashboard & analytics • System

Milestone	Due Date	Phase	Deliverables	Status	Features
					optimization & user testing • Bug fixes & UX improvements • Performance enhancements

🚀 Feature Implementation Progress

Milestone 1 - Technical Proof of Concept

- **Authentication System**: Web3Auth integration with wallet-based login
- **Frontend Foundation**: Next.js 15 with TypeScript and Tailwind CSS
- **Database Layer**: PostgreSQL with Drizzle ORM, normalized schema
- **Core UI Components**: Responsive design with shadcn/ui components
- **Image Upload**: Integrated waste reporting with photo capture

Milestone 2 - Core Prototype

- **AI Verification**: Smart waste classification and validation system
- **Rewards Engine**: Point-based incentive system with balance tracking
- **Notification System**: Real-time updates for user actions and rewards
- **Data Management**: Comprehensive reporting and tracking features
- **User Experience**: Polished interface with loading states and error handling

Milestone 3 - Extended System

- **Gamification**: Leaderboard system with user rankings and achievements
- **Analytics Dashboard**: Admin interface for system monitoring and insights
- **Optimization**: Performance improvements and user feedback integration
- **Testing & Refinement**: Bug fixes, UX enhancements, and system stability
- **Mobile Responsiveness**: Cross-device compatibility and touch optimization

Project Logging

https://docs.google.com/spreadsheets/d/1qt2mJ2I-7t5aVOVSAWLBEEHEN_XpdWWZP9iVdjLWB6Y/edit?gid=0#gid=0