Software Design Specifications

for

Eco Habits Tracker

Prepared by: Team 28 (LFC4)
Dheeru(SE22UARI046),
Dhanush(SE22UARI044),
Likhith(SE22UARI082),
Manikanta(SE22UARI042)

Document Information

Title: **Software Design Specification For Eco Habits Tracker**	
	Document Version No: 1.0
Project Manager: **Dheeru.D**	
	Document Version Date: 09-04-2025
Prepared By: **Team 28**	**Preparation Date: 09-04-2025**

Version History

Ver. No.	Ver. Date	Revised By	Description	Filename

Table of Contents

1 INTRODUCTION 4
1.1 Purpose 4
1.2 Scope 4
1.3 Definitions, Acronyms, and Abbreviations 4
1.4 References 4

2 USE CASE VIEW 4
2.1 Use Case 4

3 DESIGN OVERVIEW 4
3.1 Design Goals and Constraints 5
3.2 Design Assumptions 5
3.3 Significant Design Packages 5
3.4 Dependent External Interfaces 5
3.5 Implemented Application External Interfaces 5

4 LOGICAL VIEW 5
4.1 Design Model 6
4.2 Use Case Realization 6

5 DATA VIEW 6
5.1 Domain Model 6
5.2 D ata Model (persistent data view) 6
5.2.1 Data Dictionary 6

6 EXCEPTION HANDLING 6

7 CONFIGURABLE PARAMETERS 6

8 QUALITY OF SERVICE 7
8.1 Availability 7
8.2 Security and Authorization 7
8.3 Load and Performance Implications 7
8.4 Monitoring and Control 7

1 Introduction

The Eco Habits Tracker is a cross-platform application designed to encourage users to adopt environmentally sustainable habits through personalized habit tracking, gamification, analytics, and community interaction. The system provides real-time insights and rewards, helping users improve their ecological footprint.
This Software Design Specification (SDS) document defines the architectural structure, component design, data flow, and interaction logic of the application, based on the requirements set forth in the SRS.

1.1 Purpose

The purpose of this SDS is to serve as a bridge between the Software Requirements Specification and the implementation phase. It defines how system functionality will be realized through modular, scalable, and secure software components. This document will guide developers, testers, and maintainers.

1.2 Scope

Eco Habits Tracker supports:

- User onboarding and login
- Habit logging and tracking
- Goal setting and progress analytics
- Social challenges and leaderboards
- Real-time notifications
- Achievement unlocking

1.3 Definitions, Acronyms, and Abbreviations

- SDS: Software Design Specification
- UI: User Interface
- API: Application Programming Interface
- JWT: JSON Web Token
- AES: Advanced Encryption Standard
- GDPR: General Data Protection Regulation
- DBMS: Database Management System
- ORM: Object Relational Mapper
- MFA: Multi-Factor Authentication

1.4 References

- IEEE Std 1016-2009
- Team 28 SRS Document
- React Native, Django, PostgreSQL Documentation

- UN Sustainable Development Goals

2 Use Case View

2.1 Use Case

Use Case 1: User Login

- **Actors**: User, System
- **Preconditions**: User must be registered
- **Basic Flow**:
    1. User enters credentials
    2. Backend verifies via OAuth
    3. On success, session token is returned
- **Exceptions**:
    1. Invalid credentials → error displayed
    2. Server timeout → retry prompt

Use Case 2: Habit Logging

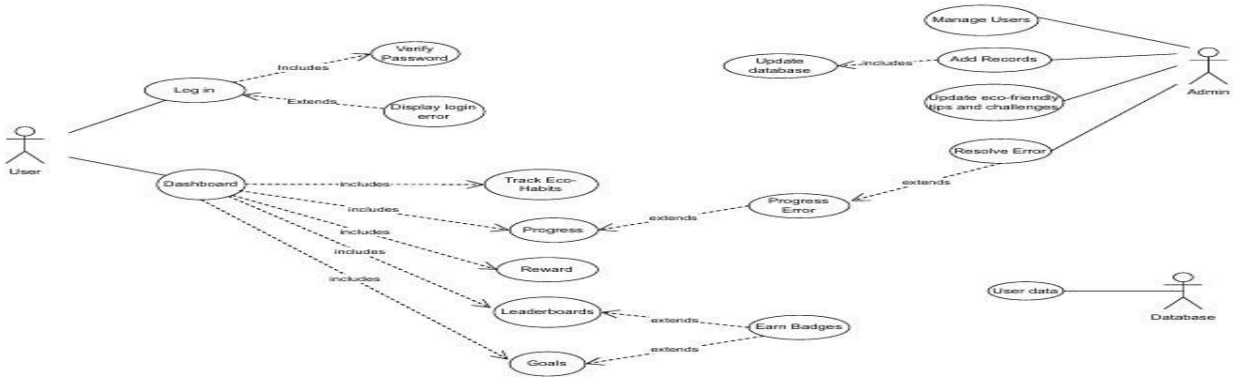
- **Actors**: User, Database
- **Flow**:
    1. User selects habit
    2. Logs progress via UI
    3. Backend updates streak, sends notification if milestone is reached

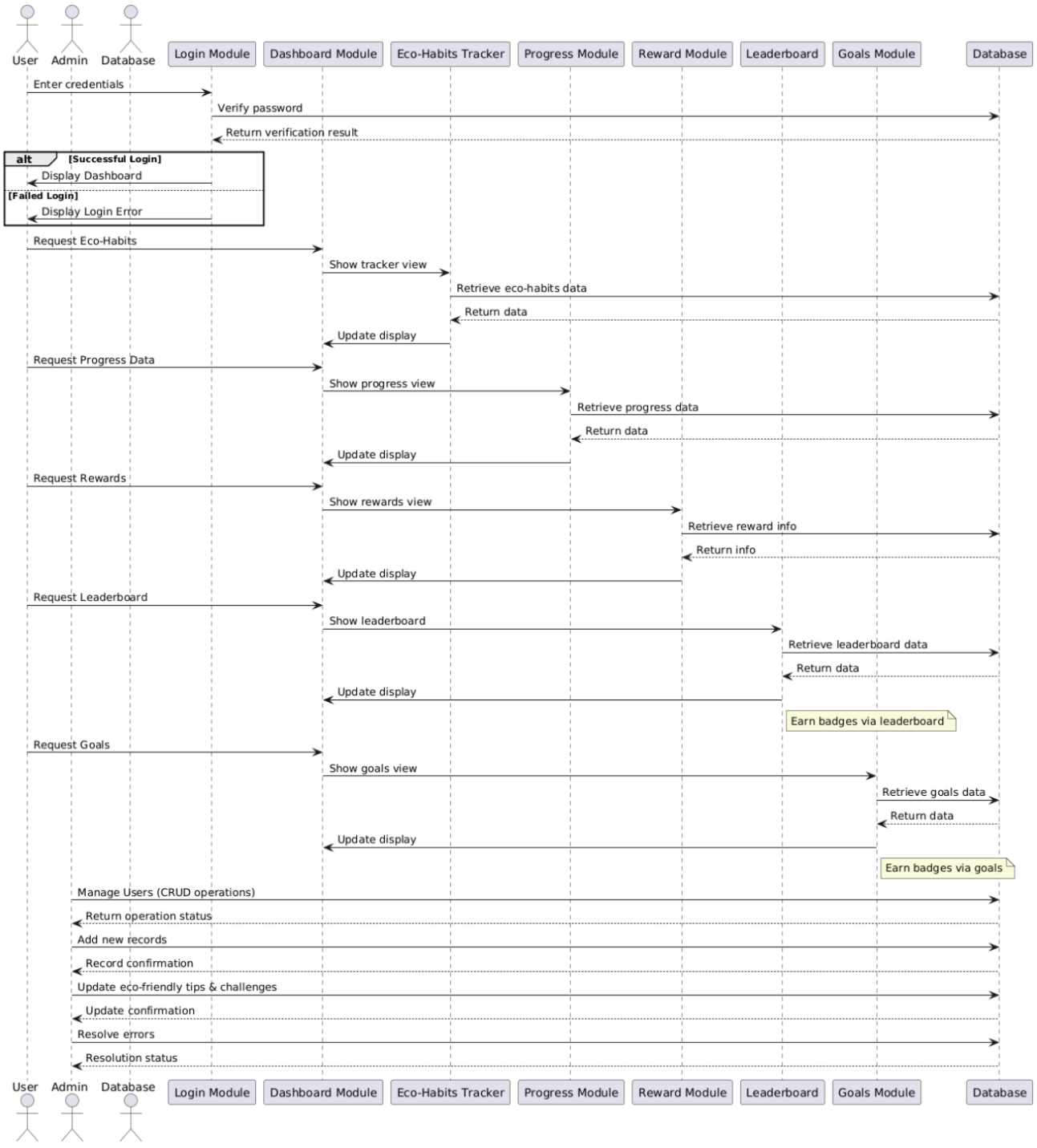
Use Case 3: Goal Tracking & Analytics

- **Actors**: User, Analytics Engine
- **Flow**:
    1. User sets a goal
    2. Backend calculates and visualizes goal status

Use Case 4: Achievement Unlocking

- **Actors**: User, Achievement Module
- **Flow**:
    1. User completes a challenge
    2. Backend checks rule
    3. New badge is unlocked and notified





3 Design Overview

3.1 Design Goals and Constraints

Design Goals

- To create a cross-platform mobile and web application that promotes eco-friendly habits.
- To implement gamification, analytics, and real-time notifications to encourage continuous user engagement.
- To ensure security and privacy of user data in compliance with GDPR and local regulations.
- To maintain high availability and responsiveness under heavy user loads.

Constraints

- Must support Android 8+ and iOS 12+ for mobile deployment.
- Frontend must use **React Native** (for mobile) and **React.js** (for web).
- Backend must use **Django REST Framework** with **PostgreSQL** as the primary database.
- The system should handle at least 10,000 concurrent users with real-time habit logging and updates.
- Limited budget and development team size (4 developers).
- Target delivery timeline: 8 weeks.

3.2 Design Assumptions

- Users have access to stable internet for real-time data sync and notifications.
- Users possess basic digital literacy to interact with a mobile/web-based app.
- External services (Firebase, OAuth 2.0, sustainability tips API) are reliably available.
- Mobile devices have basic storage and notification capabilities.
- No legacy systems or existing codebase constraints are present.
- All users will operate within the IST time zone (initial deployment scope is India)

3.3 Significant Design Packages

Frontend

- App Layer: Implements navigation using react-router-dom and React Navigation for routing and layout transitions.
- Component Layer: Consists of reusable UI components (buttons, cards, modals) styled using Tailwind CSS and Styled Components.
- Service Layer: Handles communication with the backend through Axios.
- State Management: Uses Redux Toolkit for global state and asynchronous side-effects.

Backend

- Routing Layer: Defined using Django REST URLs; endpoints follow RESTful naming conventions.
- Controller Layer: Views handle request validation, token checking, and responses.
- Service Layer: Contains business logic like streak evaluation, habit validation, goal tracking, and reward generation.
- Model Layer: Django ORM models representing users, habits, goals, notifications, and achievements.
- Security Layer: Handles OAuth 2.0, JWT-based session management, and role-based access control.

Shared/External

- Build Tool: Vite and Webpack for optimized frontend builds.
- Monitoring and Logging: Integrated with Prometheus, Grafana, and Django Logging.
- Database ORM: PostgreSQL with Django ORM; schema migrations via Alembic or Django Migrate.
- Cloud Services: AWS/Firebase used for hosting, notifications, and backups.

3.4 Dependent External Interfaces

The following external services are integrated into the Eco Habits Tracker system. These services are critical to enable secure authentication, push notifications, external sustainability content, and environmental impact data analytics.

External Application	Module Using the Interface	Interface Name	Description and Interface Usage
Firebase	Notification	Push API	Sends habit reminders and achievement unlock alerts
OAuth 2.0	Auth Module	Login API	Verifies and authorizes users securely
GreenAPI	Analytics	Impact API	Fetches environmental impact metrics based on logged habits

3.5 Implemented Application External Interfaces (and SOA web services)

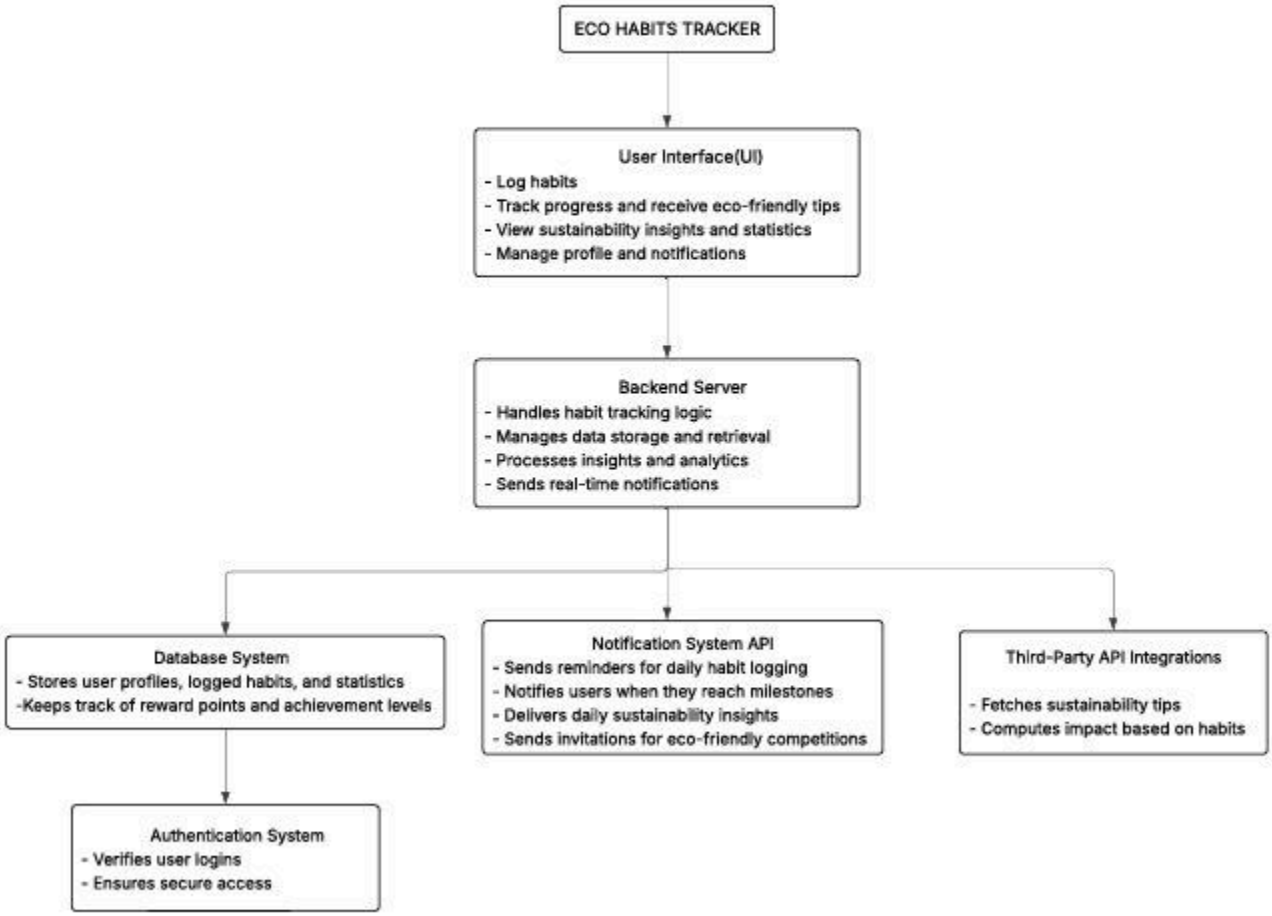
The table below lists the implementation of public interfaces this design makes available for other applications.

Interface Name	Implementing the Interface	Functionality / Description
/api/auth/login	Auth Controller	Authenticates users and issues JWT tokens
/api/habits	Habit Module	Tracks, updates, deletes user habits
/api/goals	Goal Tracker	CRUD for personal and community goals
/api/analytics	Analytics Module	Computes progress, impact, charts

4 Logical View

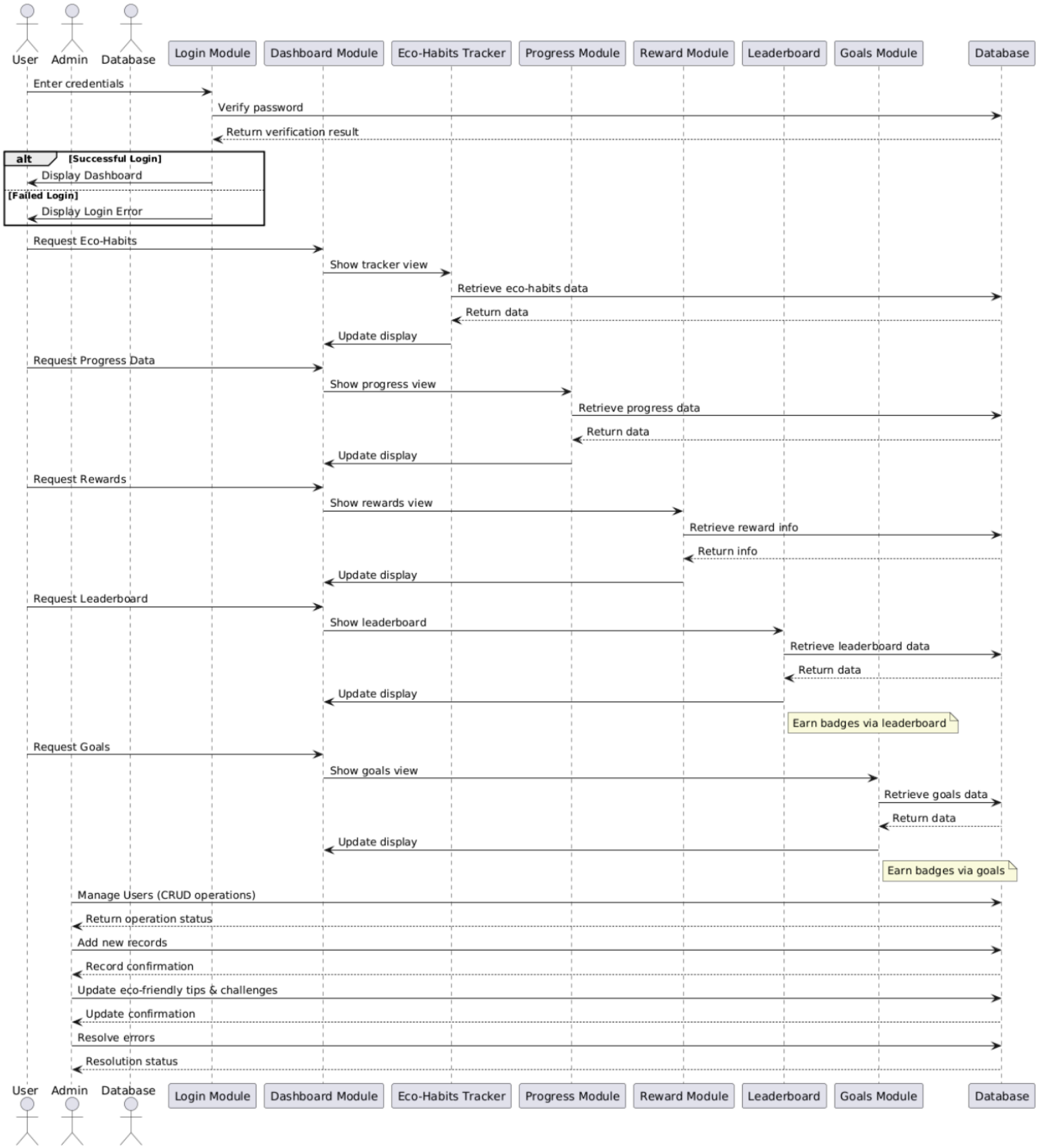
4.1 Design Model

- **User**: Login, profile, streak, goals
- **Habit**: name, frequency, category, creation date
- **Tracker**: streak, timestamps, user ref
- **Goal**: description, target, progress
- **Achievement**: title, unlockedAt, linked goals



4.2 Use Case Realization

[For each use case defined in section 2, provide detailed descriptions of how that use case scenario is realized within the design. At the highest level, show the interactions between modules using sequence or activity diagrams that implement each key use case (i.e. the business transaction implementation). For each interaction in the highest level, expand to a lower level sequence or activity diagram that shows how each class within the module collaborates to implement the required behaviour.]


User
Admin
Database
Login Module
Dashboard Module
Eco-Habits Tracker
Progress Module
Reward Module
Leaderboard
Goals Module
Database
Enter credentials
Verify password
Return verification result
alt
[Successful Login]
Display Dashboard
[Failed Login]
Display Login Error
Request Eco-Habits
Show tracker view
Retrieve eco-habits data
Return data
Update display
Request Progress Data
Show progress view
Retrieve progress data
Return data
Update display
Request Rewards
Show rewards view
Retrieve reward info
Return info
Update display
Request Leaderboard
Show leaderboard
Retrieve leaderboard data
Return data
Update display
Earn badges via leaderboard
Request Goals
Show goals view
Retrieve goals data
Return data
Update display
Earn badges via goals
Manage Users (CRUD operations)
Return operation status
Add new records
Record confirmation
Update eco-friendly tips & challenges
Update confirmation
Resolve errors
Resolution status

5 Data View

5.1 Domain Model

The Eco Habits Tracker system uses **PostgreSQL** as its primary database. It stores structured relational data optimized for quick access, scalability, and security. Tables are designed to support users, habits, goals, progress tracking, rewards, and notifications with clear foreign key relationships.


```
{
 "Users": {
  "user_id": "UUID",
  "username": "eco_user123",
  "email": "user@example.com",
  "password": "hashed_password",
  "role": "User / Admin",
  "join_date": "2025-04-01"
 },

 "Habits": {
  "habit_id": "UUID",
  "user_id": "UUID",      // Foreign key to Users
  "name": "Use Reusable Bags",
  "description": "Avoid plastic bags by using cloth bags",
  "frequency": 5,         // Times per week
  "start_date": "2025-04-01",
  "end_date": "2025-04-30"
 },

 "Trackers": {
  "tracker_id": "UUID",
  "habit_id": "UUID",     // Foreign key to Habits
  "user_id": "UUID",      // Foreign key to Users
  "progress": 80.5,
  "streak": 6
 },

 "Goals": {
  "goal_id": "UUID",
  "user_id": "UUID",      // Foreign key to Users
  "description": "Reduce carbon footprint by 25%",
  "target": 25,
  "progress": 12.5
 },

 "Achievements": {
  "achievement_id": "UUID",
  "user_id": "UUID",
  "title": "Eco Warrior",
  "description": "Logged eco-habits for 30 consecutive days",
  "date_achieved": "2025-04-15"
 },

 "Notifications": {
  "notification_id": "UUID",
  "user_id": "UUID",
```

```
  "message": "Don't forget to log your recycling today!",
  "date_sent": "2025-04-10 08:00:00"
 }
}
```


5.2 Data Model (persistent data view)

Collection	Relationships	Purpose
users	**Referenced by: habits, trackers, goals, achievements, notifications**	**Stores user credentials, profiles, and roles.**
habits	**References: users** **Referenced by: trackers**	**Stores eco-friendly habits created by users.**
trackers	**References: users, habits**	**Tracks user progress and streaks on specific habits**
goals	**References: users**	**Represents sustainability targets set by users.**
achievements	**References: users**	**Stores unlocked badges and rewards tied to user progress**
notifications	**References: users**	**Stores personalized messages and habit reminders.**

5.2.1 Data Dictionary

Field Name	Data Type	Description
user_id	UUID	Primary Key
habit_id	UUID	Foreign Key to habit
goal_id	UUID	Foreign Key to goal
name	String	Name of habit or goal

streak	Integer	Consecutive completion count
progress	Float	Completion percent

6 Exception Handling

Eco Habits Tracker implements robust exception handling mechanisms to ensure resilience and a smooth user experience. Below are key exceptions managed in the system:

1. **AuthenticationFailedException**
    - **Triggered When**: A user enters incorrect login credentials.
    - **Handling**: A generic message "Invalid email or password" is displayed.
    - **Logging**: Logs include the email entered, IP address, and timestamp.
    - **Follow-up**: After 5 attempts, temporary lockout and a CAPTCHA prompt are enabled.
2. **InvalidHabitEntryException**
    - **Triggered When**: A user submits an incomplete or invalid habit entry.
    - **Handling**: The system displays field-specific error messages.
    - **Logging**: Logs the user ID, invalid payload, and time.
    - **Follow-up**: Prompts re-entry and suggests valid formats.
3. **UnauthorizedAccessException**
    - **Triggered When**: A user tries to access or manipulate another user's data.
    - **Handling**: The system redirects to an "Access Denied" page and logs the attempt.
    - **Logging**: Includes user ID, resource attempted, and timestamp.
    - **Follow-up**: Repeated attempts alert admins.
4. **GoalNotFoundException**
    - **Triggered When**: A requested goal ID does not exist in the database.
    - **Handling**: Message "No goal found" is returned.
    - **Logging**: Logs include user ID and missing goal ID.
    - **Follow-up**: Option to recreate or return to the dashboard.
5. **ExternalAPIFailureException**
    - **Triggered When**: Sustainability or notification APIs fail.
    - **Handling**: Fallback message shown, system retries up to 3 times.
    - **Logging**: Includes API name, endpoint, status code, and time.
    - **Follow-up**: Alerts are sent to the development team for manual resolution.
6. **DatabaseConnectionException**
    - **Triggered When**: Backend fails to connect to the PostgreSQL database.
    - **Handling**: Displays "Service unavailable. Please try again later."
    - **Logging**: Logs full stack trace with timestamp.
    - **Follow-up**: Auto-alert to admin; application attempts reconnection every 30 seconds.
7. **TokenExpiredException**
    - **Triggered When**: A user's JWT token has expired.
    - **Handling**: System prompts the user to log in again.
    - **Logging**: Records user ID and token expiration timestamp.
    - **Follow-up**: Regenerates token upon successful login.

These structured exception flows ensure user guidance, developer traceability, and system integrity.

7 Configurable Parameters

Configuration Parameter Name	Definition and Usage	Dynamic?
MAX_LOGIN_ATTEMPTS	Lockout after threshold	**Yes**
NOTIFICATION_FREQUENCY	Habit alert frequency	**Yes**
REWARD_THRESHOLD	Points for reward unlock	**Yes**
SESSION_EXPIRY	JWT token validity	**Yes**
DATA_RETENTION_DAYS	Cleanup period	**Yes**

8 Quality of Service

8.1 Availability

- 99.9% uptime through AWS/Firebase
- Auto-healing containers
- DB backups every 12 hours

8.2 Security and Authorization

- Role-based access
- OAuth 2.0 + encrypted JWT
- AES-256 for sensitive fields
- 2FA optional via mobile OTP

8.3 Load and Performance Implications

- API < 500ms response
- Daily usage forecast: 5k active users
- Caching via Redis for leaderboard and tips

8.4 Monitoring and Control

- Prometheus + Grafana for dashboards
- Audit logs tracked per user
- Slack/email alerts for downtime