Software Design Document

Habit Tracker

Version: 1.0

Prepared by:

Group Name: AAAAS

Name	Student ID	Email
Srachet Rai	BT22GCS155	srachet.rai22@st.niituniversity.in
Abhi Bhardwaj	BT22GCS108	abhi.bhardwaj22@st.niituniversity.in
Anmol Ranjan	BT22GCS082	anmol.ranjan22@st.niituniversity.in
Adarsh Bajpai	BT22GCS055	adarsh.bajpai22@st.niituniversity.in
Avijith Manikandan	BT22GCS140	avijith.manikandan22@st.niituniversity.in

Instructor: Mr. Manish Hurkat
Course: Capstone 1
Date: 02/02/2025

Table of Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Initial Draft v1	Srachet Rai	Started the initial draft of the SDD document, outlining the major headings and subheadings to be discussed	02/02/2024
Initial Draft v2	Srachet Rai, Adarsh Bajpai	Added database schema and wireframe	15/02/2024
Mid Submission 1	Srachet Rai, Abhi Bhardwaj	Finalizing the document for mid-submission	20/02/2024

Introduction

➢ Document Purpose

The following document provides a detailed specification of the habit tracker application, outlining its functionality, design constraints, and specific requirements. It may serve as a guide for developers, designers, stakeholders, and testers throughout the development process.

➢ Product Scope

The habit tracker is a web-based and mobile-friendly application that helps users track, and analyze their daily habits, goals, and tasks. It integrates an AI-powered chatbot for personalized recommendations and insight, supports speech-to-text input and ensures secure data storage using MongoDB. Key features include -

- Habit and Goal Tracking (with informative dashboard)
- Pomodoro Timer
- AI Assistant (LLM API) and Progress Map
- In application Note-Taking
- Speech-to-Text Functionality
- User Profile and Personalization
- Motivational Quotes and load time tips

➢ Intended Audience and Document Overview

Developers: Require technical specifications and instructions for implementation.
Project Managers: Consider the project's objectives, schedule, and essential specifications.
Marketing Staff: attracted by the features, advantages, and positioning of the app in the market.
Users: Need guidance on how to utilize the app and get its advantages.
Testers: Require acceptance criteria, use cases, and testing protocols.
Documentation Writers: To build user manuals and support materials, comprehensive information is required.
System Administrators: System requirements are necessary for setup and maintenance.
Designers: To create interfaces and experiences, information about user requirements is required.

➢ Definitions, Acronyms, and Abbreviations

Here is a short list of abbreviations and acronyms used in the SRS document, sorted alphabetically:

API: Application Programming Interface

DP-DP: Data protection and digital privacy law

LLM: Large Language Model

SRS: Software Requirements Specification

URL: Uniform Resource Locator

MERN: MongoDB, Express,js, React, Node.js

UI/UX: User Interface/ User Experience

➢ Document Conventions

In general, this document follows the IEEE formatting requirements. Use Arial font size 11, or 12 throughout the document for text. Use italics for comments. Document text should be single-spaced and maintain the 1" margins found in this template. For Section and Subsection titles please follow the template.

Formatting Conventions:

- Font: To ensure readability on a variety of devices and formats, the main text is written in Arial at an 11-point size.
- Headings and Subheadings: To make it easy to identify the important sections, headings (such as section titles) are bolded and set in a 14-point font size.
- Paragraphs: To improve readability, a space is inserted between each paragraph and the body content is left-aligned with a single line spacing.
- Lists: To convey information in an ordered and succinct manner, bullet points are utilized for unordered lists.
- Sequential instructions are easier to follow when presented in ordered lists, which are utilized for organized tasks or procedures.

➢ References and Acknowledgments

- https://api-docs.deepseek.com/guides/reasoning_model
- https://platform.openai.com/docs/api-reference/introduction
- https://cloud.google.com/vision/docs/ocr
- https://cloud.google.com/speech-to-text/docs

System Overview

The **Habit Tracker** is an intelligent habit-tracking system that allows users to create, manage, and analyze their daily habits. It provides personalized AI-powered recommendations and integrates digital well-being features such as screen time monitoring and focus mode. The system ensures secure data storage using MongoDB and supports offline functionality via local storage.

System Architecture

3.1 Architectural Design

The Habit Tracker follows a **MERN (MongoDB, Express.js, React, Node.js) stack architecture** for a scalable and efficient backend. The system consists of the following modules:

- **Frontend (React.js):** User interface and interactive components.
- **Backend (Node.js & Express.js):** API handling, authentication, and AI integration.
- **Database (MongoDB):** Stores user habits, progress, and settings.
- **AI Module:** Provides habit recommendations via an LLM API.

(A system architecture diagram is provided in **Appendix A**.)

3.2 Decomposition Description

The system follows an **MVC (Model-View-Controller) pattern**:

- **Model:** Manages database interactions (MongoDB).
- **View:** UI components built with React.js.
- **Controller:** Handles API requests and user interactions.

3.3 Design Rationale

We selected the **MERN stack** for its flexibility, scalability, and real-time data handling. The system was designed to be **mobile-friendly** and **responsive** for better user accessibility.

Data Design

4.1 Data Description

The system stores user data, including:

- **User Profiles** (Name, Email, Preferences)
- **Habit Logs** (Date, Completion Status, Notes)
- **AI Recommendations**

4.2 Database Schema (Appendix D)

Below is the entity-relationship (ER) design for this project. This explanation breaks down each entity with its attributes and keys and then lists the relationships between entities.

1. Entity Details

Table: USERS

Attribute	Data Type	Key	Description
_id	string	Primary Key	Unique identifier for the user
name	string	–	User's full name
email	string	Unique	User’s email address
passwordHash	string	–	Hashed password for security
createdAt	date	–	Account creation timestamp
updatedAt	date	–	Last update timestamp

Table: ZEN_GARDENS

Attribute	Data Type	Key	Description
_id	string	Primary Key	Unique identifier for the Zen Garden
userId	string	Foreign Key	References the owning user (_id in USERS)
health	string	–	Garden status: “Thriving” or “Weedy”

Table: GOALS

Attribute	Data Type	Key	Description
_id	string	Primary Key	Unique identifier for the goal
gardenId	string	Foreign Key	References the Zen Garden (_id in ZEN_GARDENS)
title	string	–	Title or description of the goal
createdAt	date	–	When the goal was created

Table: HABIT_PATCHES

Attribute	Data Type	Key	Description
_id	string	Primary Key	Unique identifier for the habit patch
goalId	string	Foreign Key	References the goal (_id in GOALS)
flowerType	string	–	Represents status: “Healthy”, “Fading”, “Withered”, or “Bonus”

Table: HABITS

Attribute	Data Type	Key	Description
_id	string	Primary Key	Unique identifier for the habit
patchId	string	Foreign Key	References the habit patch (_id in HABIT_PATCHES)
userId	string	Foreign Key	References the owning user (_id in USERS)
title	string	–	Habit title or name
description	string	–	Detailed description of the habit
category	string	–	Category of the habit (e.g., Morning Routine)
frequency	number	–	How often the habit should be performed
type	string	–	“Daily”, “Weekly”, or “Monthly”
status	string	–	“StrictlyFollowed”, “PartiallyFollowed”, or “Ignored”
streak	number	–	Current streak count
createdAt	date	–	When the habit was created
updatedAt	date	–	Last updated timestamp

Table: COMPLETION_LOGS

Attribute	Data Type	Key	Description
_id	string	Primary Key	Unique identifier for the completion log
habitId	string	Foreign Key	References the habit (_id in HABITS)
userId	string	Foreign Key	References the user (_id in USERS)
logDate	date	–	Date when the habit was logged
isCompleted	boolean	–	Indicates if the habit was completed
notes	string	–	Additional notes regarding the completion

Table: ANALYTICS

Attribute	Data Type	Key	Description
_id	string	Primary Key	Unique identifier for the analytics record
userId	string	Foreign Key	References the user (_id in USERS)
totalHabits	number	–	Total number of habits for the user
completedHabits	number	–	Number of habits completed
streaks	number	–	Cumulative streak information
completionRate	number	–	Percentage or rate of completion
lastUpdated	date	–	Timestamp for last update on analytics

Table: REWARDS

Attribute	Data Type	Key	Description
_id	string	Primary Key	Unique identifier for the reward record
userId	string	Foreign Key	References the user (_id in USERS)
rewardPoints	number	–	Points earned by the user

Table: THEMES

Attribute	Data Type	Key	Description
_id	string	Primary Key	Unique identifier for the theme
rewardId	string	Foreign Key	References the reward (_id in REWARDS)
themeName	string	–	Name of the theme

Table: DECORATIONS

Attribute	Data Type	Key	Description
_id	string	Primary Key	Unique identifier for the decoration
rewardId	string	Foreign Key	References the reward (_id in REWARDS)
itemName	string	–	Name of the decoration item

2. Relationship Details

The following table summarizes the relationships between the entities:

Relationship	Source Entity	Target Entity	Cardinality	Description
owns	USERS	ZEN_GARDENS	1 : 0..*	A user can own one or more Zen Gardens.
contains	ZEN_GARDENS	GOALS	1 : 0..*	A Zen Garden can contain multiple Goals.
has	GOALS	HABIT_PATCHES	1 : 0..*	Each Goal may have one or more Habit Patches.
represents	HABIT_PATCHES	HABITS	1 : 0..*	Each Habit Patch represents one or more Habits.

tracked_by	HABITS	COMPLETION_LOGS	1 : 0..*	Each Habit can have multiple Completion Logs (each log is an instance of tracking).
logs (direct relationship)	USERS	COMPLETION_LOGS	1 : 0..*	Users can also be directly associated with their Completion Logs.
has	USERS	ANALYTICS	1 : 0..*	Each user has associated Analytics data.
earns	USERS	REWARDS	1 : 0..*	A user earns Reward points.
unlocks	REWARDS	THEMES	1 : 0..*	A reward record can unlock multiple Themes.
unlocks	REWARDS	DECORATIONS	1 : 0..*	A reward record can unlock multiple Decorations.

Note: The cardinality “1 : 0..*” indicates that one record in the source entity can be related to zero or many records (1 to many) in the target entity.

Component Design

Each component follows modular development principles. Key components include:

- **User Authentication Module**
- **Habit Management Module**
- **Analytics Dashboard**
- **AI-Powered Habit Recommendation System**

The flowchart is detailed in **Appendix B**.

Human Interface Design

6.1 Overview of User Interface

The **Habit Tracker** provides an intuitive UI with features such as:

- **Dashboard**: Displays habits, goals, and analytics.
- **Habit Editor**: Allows users to add and modify habits.
- **AI Chatbot**: Suggests personalized habit improvements.

6.2 Screen Images

Mockups and UI designs are provided in **Appendix C**.

6.3 Screen Objects and Actions

UI Element / Button	Function / Interaction	Affected Database Entities	Notes / Workflow
Sign Up	Creates a new user account	USERS	Collects details (name, email, password) and inserts a new record into the USERS table.
Log In	Authenticates an existing user	USERS (read)	Validates credentials and starts a session.
Logout	Ends the current user session	N/A	Terminates the active session.
My Zen Garden	Displays the user's personalized Zen Garden view	ZEN_GARDENS, GOALS, HABIT_PATCHES, HABITS	Shows the overall garden status (health), active goals, habit patches, and associated habits; acts as a dashboard for daily interactions.
Add Goal	Allows a user to create a new goal	GOALS	Presents a form to enter goal details (title, creation date) and associates the goal with the user's Zen Garden.
Edit Goal	Enables a user to update an existing goal	GOALS	Updates goal details such as title or creation date.
Add Habit Patch	Lets a user create a new habit patch under a selected goal	HABIT_PATCHES	Users choose a flower type (Healthy, Fading, Withered, or Bonus) to represent the patch; links to a specific goal.
Add Habit	Allows a user to create a new habit within a habit patch	HABITS	Users input habit details (title, description, frequency, type [Daily/Weekly/Monthly], status, and initial streak) and associate it with a habit patch.
Edit Habit	Enables editing of an existing habit's details	HABITS	Updates fields such as title, description, frequency, type, or status.

Log Completion	Marks a habit as completed for a given time period and records a completion log	COMPLETION_LOGS; updates HABITS (streak)	Adds a new record in COMPLETION_LOGS (with log date, completion status, and notes) and may update the habit's streak count.
View Habit History	Displays all completion logs for a selected habit	COMPLETION_LOGS	Presents a list or calendar view of past completion entries for tracking progress.
View Analytics	Shows a dashboard summarizing the user's habit tracking progress	ANALYTICS	Displays metrics such as total habits, completed habits, streaks, and completion rate; helps users gauge their performance over time.
Claim / View Rewards	Provides access to view and redeem reward points	REWARDS; indirectly THEMES and DECORATIONS	Shows current reward points and offers options to unlock new themes or decorations for customizing the Zen Garden.
Customize Garden	Allows users to apply unlocked themes and decorations to personalize their Zen Garden	ZEN_GARDENS (updated via reward redemption); THEMES; DECORATIONS	Lets users visually customize the garden using rewards; triggers updates to the garden’s appearance based on selected themes or decorations.

Requirements Matrix

This table maps system components to functional requirements.

System Component	Functional Requirement	UI Requirements	Affected Database Entities	Description	Priority

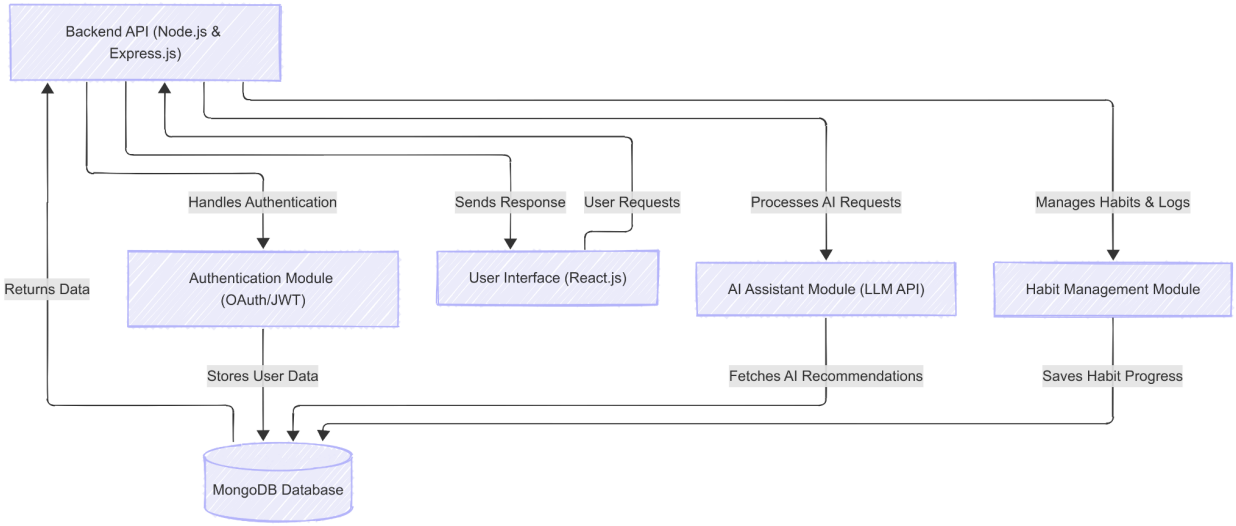
User Authentication	**Sign Up**	**Sign Up Button**	USERS	Allow new users to register by validating inputs, ensuring unique emails, hashing passwords, and creating a new record in the USERS table.	High
	Log In	**Log In Button**	USERS	Authenticate users by checking credentials against the USERS table. Successful login initiates a secure session; failure displays a clear error message.	High
	Log Out	**Log Out Button**	N/A	Terminate the user session and redirect to the login or home screen.	High
Zen Garden Dashboard	**Display User's Zen Garden**	**Zen Garden Dashboard**	ZEN_GARDENS, GOALS, HABIT_PATCHES, HABITS	Render a personalized dashboard that aggregates and visually presents the state of the user's Zen Garden and its associated data (goals, patches, habits).	High
Goal Management	**Add Goal**	**Add Goal Button**	GOALS, ZEN_GARDENS	Allow users to create a new goal linked to their Zen Garden. On submission, a new goal record is inserted into the GOALS table with the appropriate garden association.	High
	Edit Goal	**Edit Goal Button**	GOALS	Permit users to update goal details. The updated information is saved to the GOALS table and reflected immediately on the dashboard.	Medium

Habit Patch Management	**Add Habit Patch**	**Add Habit Patch**	HABIT_PATCHES, GOALS	Allow users to add a habit patch under a selected goal by choosing a flower type. The new habit patch is recorded in HABIT_PATCHES and linked to the chosen goal via goalId.	High
Habit Management	**Add Habit**	**Add Habit Button**	HABITS, HABIT_PATCHES, USERS	Let users create a new habit within a habit patch. This includes entering all habit details, which are then saved in the HABITS table with proper linkage to the habit patch and user.	High
	Edit Habit	**Edit Habit Button**	HABITS	Allow users to modify an existing habit's details (such as title, description, frequency, or status). Updates are saved to the HABITS table and reflected in the habit display.	Medium
Habit Tracking	**Log Completion**	**Log Completion Button**	COMPLETION_LOGS, HABITS	Enable users to mark a habit as completed. When logged, a new record is inserted into COMPLETION_LOGS, and the habit's streak in the HABITS table is updated accordingly.	High
	View Habit History	**View Habit History Button**	COMPLETION_LOGS	Provide a detailed history view of habit completions, allowing users to track their progress over time by fetching and displaying records from the COMPLETION_LOGS table.	Medium

Analytics Module	**View Analytics**	**View Analytics Button**	ANALYTICS, USERS	Display a dynamic analytics dashboard that aggregates and visualizes key performance metrics derived from user habits and logs stored in the ANALYTICS and USERS tables.	Medium
Rewards & Customization	**Claim / View Rewards**	**Claim/View Rewards Button**	REWARDS, THEMES, DECORATIONS	Allow users to view and redeem reward points for unlocking new themes and decorations. Updates in reward points and unlocked items are saved in the REWARDS, THEMES, and DECORATIONS tables.	Medium
	Customize Zen Garden	**Customize Zen Garden Button**	ZEN_GARDENS, THEMES, DECORATIONS, REWARDS	Enable users to personalize their Zen Garden by applying unlocked customizations. The selected theme or decoration updates the visual representation of the Zen Garden.	Medium

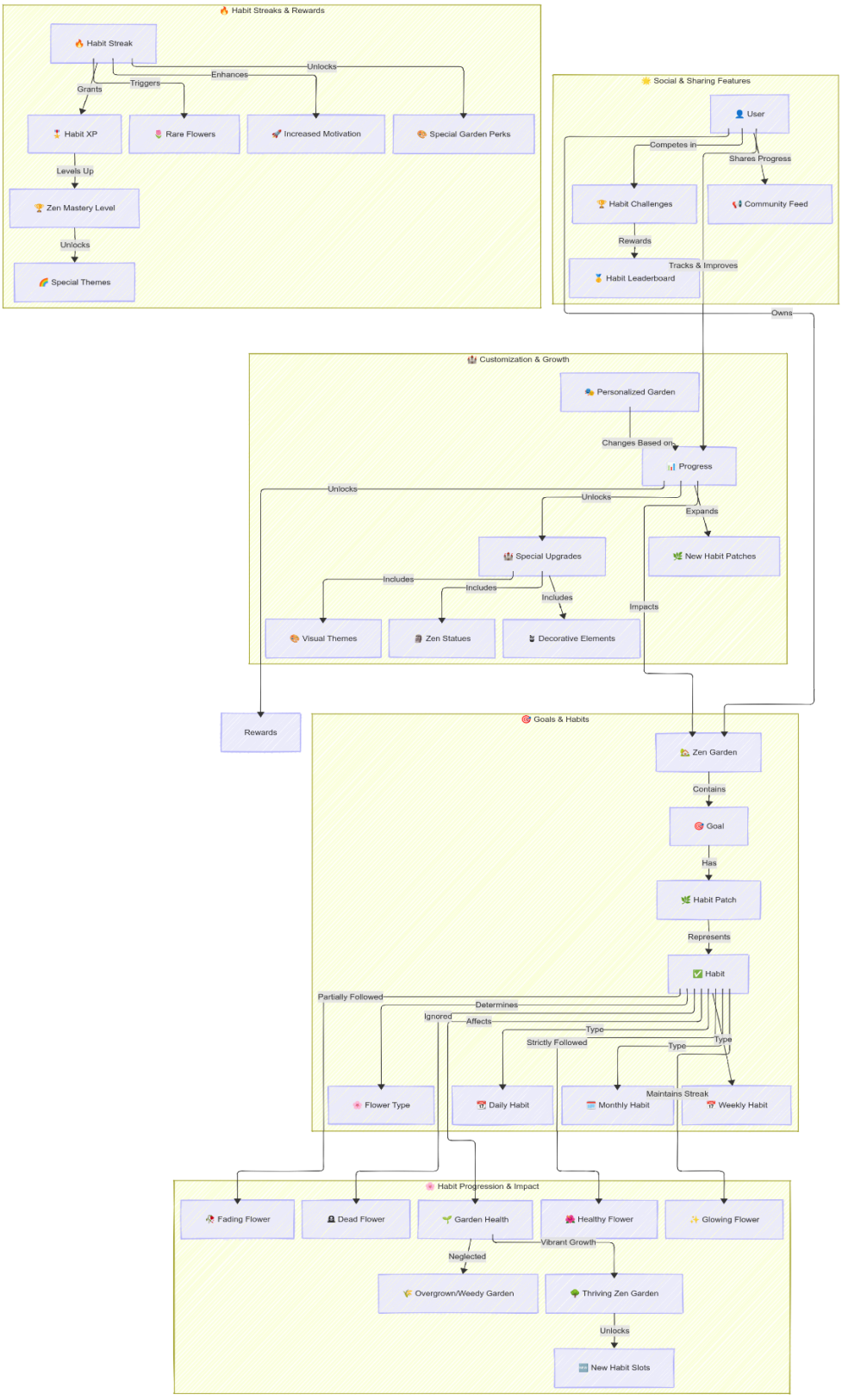
Appendices

Appendix A – System Architecture Diagram

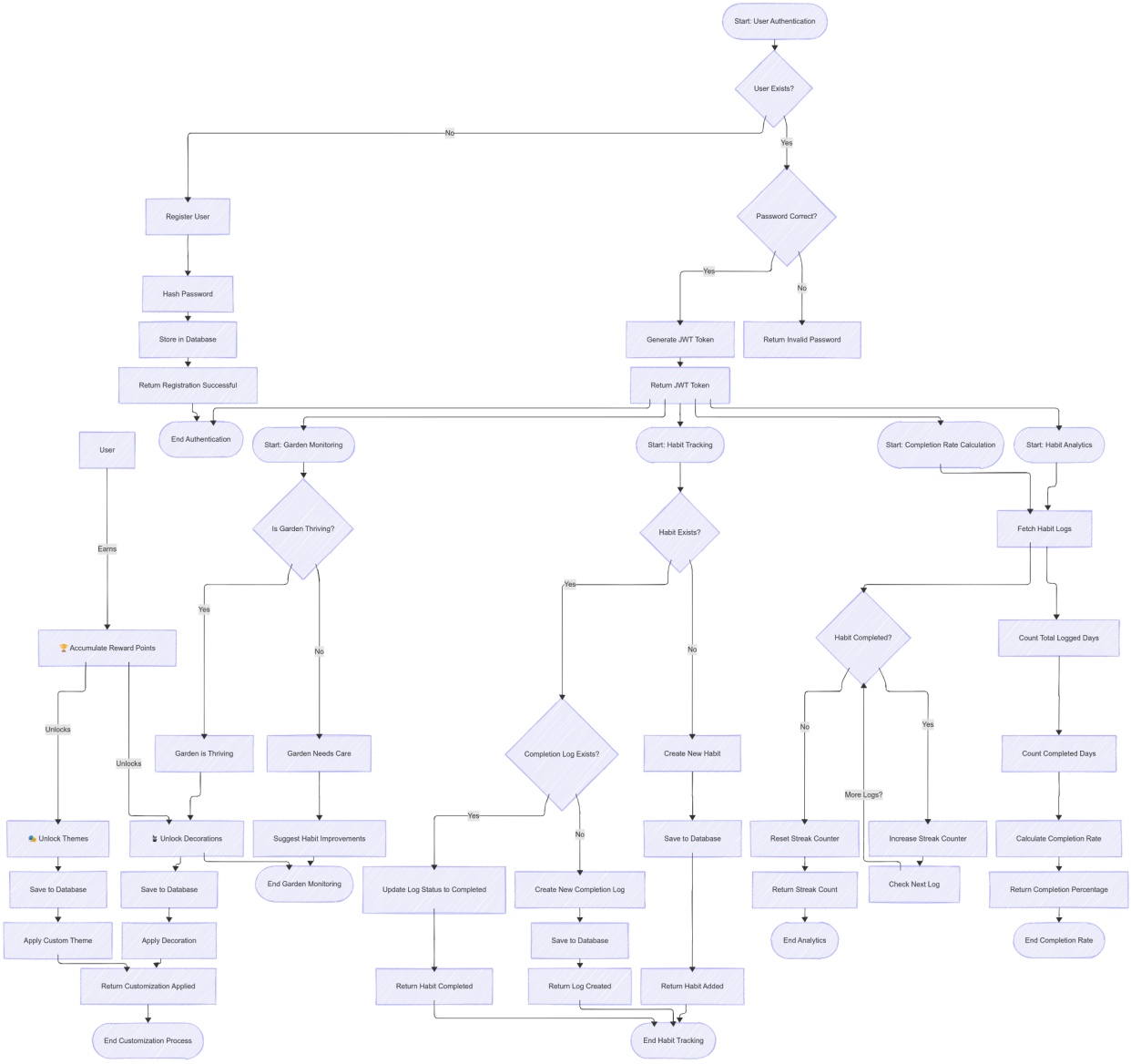


Explanation:

- **Frontend (React.js)** interacts with the **Backend API**.
- **Backend API (Node.js & Express.js)** manages:
  - **Authentication Module** (OAuth/JWT) for user logins.
  - **AI Assistant Module** (LLM API) for habit recommendations.
  - **Habit Management Module** for tracking user habits.
- **Database (MongoDB)** stores **users, habits, logs, and AI recommendations**.

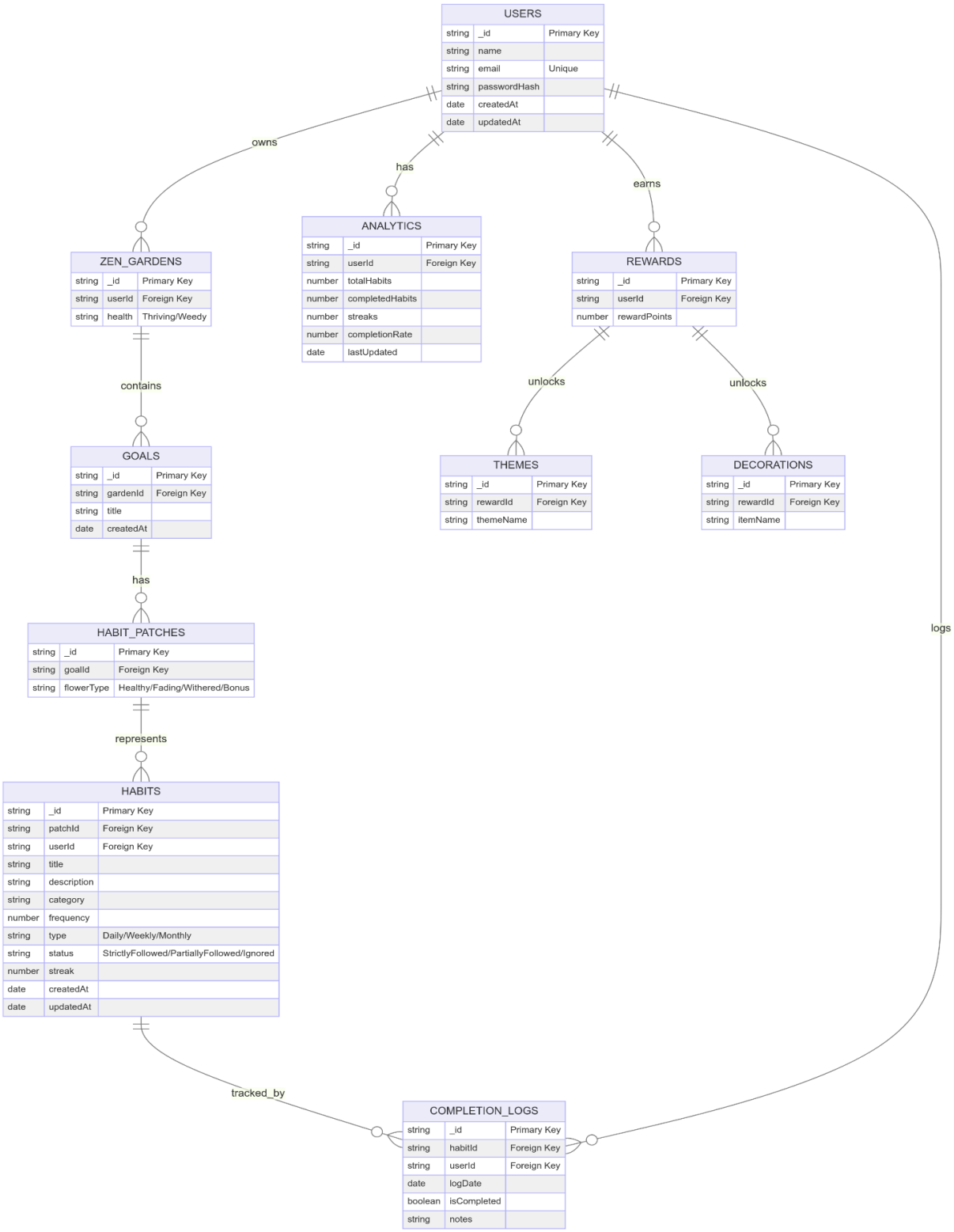

🔥 Habit Streaks & Rewards
🔥 Habit Streak
Grants
Triggers
Enhances
Unlocks
🎖️ Habit XP
🌷 Rare Flowers
🚀 Increased Motivation
🎨 Special Garden Perks
Levels Up
🏆 Zen Mastery Level
Unlocks
🌈 Special Themes
🌟 Social & Sharing Features
👤 User
Competes in
Shares Progress
🏆 Habit Challenges
📢 Community Feed
Rewards
Tracks & Improves
🏅 Habit Leaderboard
Owns
🏰 Customization & Growth
🎭 Personalized Garden
Changes Based on
📊 Progress
Unlocks
Unlocks
Expands
🏰 Special Upgrades
🌿 New Habit Patches
Includes
Includes
Includes
Impacts
🎨 Visual Themes
🗿 Zen Statues
🕯️ Decorative Elements
Rewards
🎯 Goals & Habits
🏡 Zen Garden
Contains
🎯 Goal
Has
🌿 Habit Patch
Represents
✅ Habit
Partially Followed
Determines
Ignored
Affects
Type
Strictly Followed
Type
Type
Type
Maintains Streak
🌸 Flower Type
📆 Daily Habit
🗓️ Monthly Habit
📅 Weekly Habit
🌸 Habit Progression & Impact
🥀 Fading Flower
🪦 Dead Flower
🌱 Garden Health
🌺 Healthy Flower
✨ Glowing Flower
Vibrant Growth
Neglected
🌾 Overgrown/Weedy Garden
🌳 Thriving Zen Garden
Unlocks
🆕 New Habit Slots

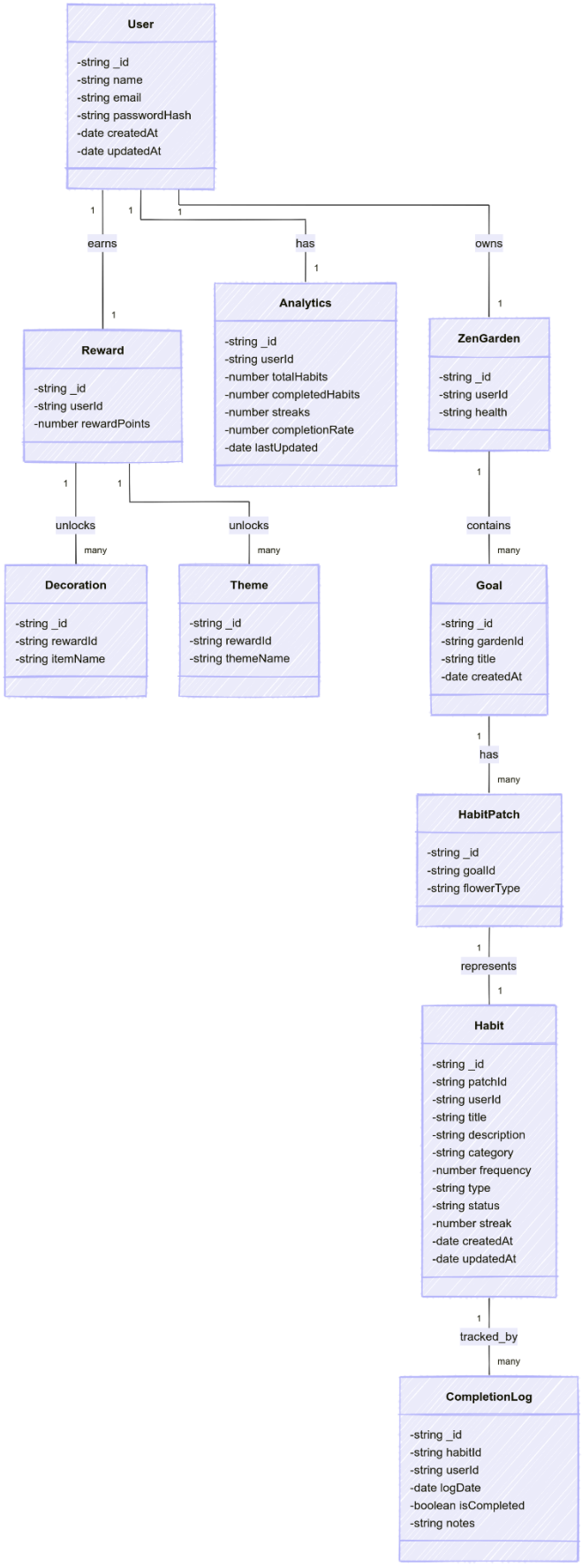
Appendix B – Algorithms & Pseudocode



Appendix C – UI Mockups

Appendix D – Database Schema & Class Diagram




User
-string _id
-string name
-string email
-string passwordHash
-date createdAt
-date updatedAt
1
1
1
earns
has
owns
1
1
1
Reward
-string _id
-string userId
-number rewardPoints
Analytics
-string _id
-string userId
-number totalHabits
-number completedHabits
-number streaks
-number completionRate
-date lastUpdated
ZenGarden
-string _id
-string userId
-string health
1
1
1
1
unlocks
unlocks
contains
many
many
many
Decoration
-string _id
-string rewardId
-string itemName
Theme
-string _id
-string rewardId
-string themeName
Goal
-string _id
-string gardenId
-string title
-date createdAt
1
has
many
HabitPatch
-string _id
-string goalId
-string flowerType
1
represents
1
Habit
-string _id
-string patchId
-string userId
-string title
-string description
-string category
-number frequency
-string type
-string status
-number streak
-date createdAt
-date updatedAt
1
tracked_by
many
CompletionLog
-string _id
-string habitId
-string userId
-date logDate
-boolean isCompleted
-string notes