StepHabit Capstone Report

A comprehensive technical and product deep dive

Prepared by: Engineering Team

Date: December 16, 2025

Abstract

StepHabit is a full-stack habit-building and productivity platform designed to help users convert long-term goals into consistent daily action. The system integrates habit tracking, task management, intelligent scheduling, and AI-assisted coaching within a unified experience. This report presents a detailed technical and product-level analysis of StepHabit, covering system architecture, data modeling, backend services, frontend workflows, and implementation decisions. The document serves both as a formal capstone submission and as a long-term reference for future contributors and system extensions.

Contents

1 Introduction

1.1 System Overview

StepHabit is a secure, AI-assisted productivity and habit-building platform designed to help users transform long-term goals into consistent daily routines. Built with a strong emphasis on structure, motivation, and sustainability, the system enables users to create, manage, and track habits, tasks, and schedules within a unified environment. A secure registration and authentication process ensures account integrity, while personalized dashboards allow users to monitor progress, streaks, and achievements over time.

Once authenticated, users can define habits with daily goals, manage tasks with durations and deadlines, and organize their time through calendar-based planning and busy-block scheduling. The platform provides visual feedback through planners, charts, and progress logs, helping users clearly understand how their daily actions align with broader objectives. Notifications, reminders, and achievement milestones reinforce engagement and encourage consistency.

Beyond traditional productivity tools, StepHabit integrates artificial intelligence to provide personalized coaching and guidance. An AI-driven assistant analyzes user input and context to generate structured habit plans, refine habit ideas, and offer actionable suggestions. By maintaining assistant memory tied to individual users, the system delivers increasingly relevant feedback and supports reflective, goal-oriented decision making over time.

User data is stored securely and is accessible only to authenticated users, with all core features scoped to individual accounts to ensure privacy. Social features such as friendships, messaging, and group challenges are implemented within controlled boundaries, allowing users to engage in accountability-driven interactions without compromising personal data. Calendar integrations further enhance the experience by preventing scheduling conflicts between habits, tasks, and external commitments.

In summary, StepHabit combines secure account management, structured planning, social accountability, and AI-powered coaching to deliver a comprehensive and intelligent productivity experience. The platform demonstrates how modern web technologies and artificial intelligence can be thoughtfully integrated to support sustainable habit formation and long-term personal growth.

1.2 Objectives

The primary objective of this project is to design and develop a secure, AI-assisted productivity and habit-building platform that enables users to translate long-term goals into consistent daily actions. The system aims to provide a structured yet flexible environment in which users can create, manage, and track habits, tasks, and schedules, while receiving intelligent guidance to improve planning and execution. Through AI-powered coaching, the platform supports users in refining habit ideas, breaking goals into actionable steps, and maintaining motivation over time.

In addition, the project seeks to deliver an intuitive and user-friendly interface, secure account management through authenticated access, and personalized dashboards that visu-

alize progress, streaks, and achievements. By combining scheduling tools, progress tracking, and social accountability mechanisms, StepHabit encourages sustainable behavior change rather than short-term productivity bursts.

Secondary objectives include building a scalable full-stack web application using modern frameworks, ensuring data security and user privacy, and establishing a robust architectural foundation that can support future extensions such as mobile clients, advanced analytics, or enhanced AI-driven personalization.

1.3 Scope

The primary objective of this project is to design and develop a secure, AI-assisted productivity and habit-building platform that enables users to translate long-term goals into consistent daily actions. The system aims to provide a structured yet flexible environment in which users can create, manage, and track habits, tasks, and schedules, while receiving intelligent guidance to improve planning and execution. Through AI-powered coaching, the platform supports users in refining habit ideas, breaking goals into actionable steps, and maintaining motivation over time.

In addition, the project seeks to deliver an intuitive and user-friendly interface, secure account management through authenticated access, and personalized dashboards that visualize progress, streaks, and achievements. By combining scheduling tools, progress tracking, and social accountability mechanisms, StepHabit encourages sustainable behavior change rather than short-term productivity bursts.

Secondary objectives include building a scalable full-stack web application using modern frameworks, ensuring data security and user privacy, and establishing a robust architectural foundation that can support future extensions such as mobile clients, advanced analytics, or enhanced AI-driven personalization.

1.4 Scope of the Project

1.5 Structure of the Report

2 System Architecture

2.1 Overview of System Components

The StepHabit platform is designed using a modular system architecture, in which each major component is responsible for a clearly defined set of responsibilities. This architectural approach improves clarity, maintainability, and scalability, while allowing individual components to evolve independently. Together, these components form a cohesive, secure, and intelligent productivity system that supports habit formation, task management, and goal-oriented behavior.

The frontend application serves as the primary interaction layer between users and the system. It provides interfaces for user registration and authentication, habit and task creation, calendar-based planning, and progress tracking. Through dashboards, planners, and visual analytics, users can monitor habits, streaks, achievements, and upcoming tasks. Social features such as messaging, group challenges, and notifications are also accessible through the frontend, enabling accountability-driven engagement in a user-friendly and responsive interface.

The backend server implements the core business logic and coordinates communication between the frontend, the database, and external services. It manages authentication and authorization, enforces data ownership and privacy rules, and processes all user actions, including habit updates, task scheduling, progress logging, notifications, and community interactions. The backend also exposes RESTful APIs that provide a stable and extensible interface for current and future clients.

The database layer is responsible for persistent data storage and integrity. It stores user profiles, habits, tasks, schedules, progress records, achievements, notifications, assistant memory, and social relationships. The relational schema is designed to support clear ownership boundaries between users and their data, while enabling efficient queries for analytics, dashboards, and planner views.

An AI-powered coaching component enhances the platform by providing intelligent guidance and personalization. This service analyzes user-provided habit ideas and contextual data to generate structured habit plans, refine goal descriptions, and offer actionable suggestions. Assistant memory is maintained on a per-user basis, allowing the system to deliver increasingly relevant and consistent coaching over time. AI outputs are evaluated and stored where appropriate to support reflection and future interactions.

Together, these components form a robust and extensible system that integrates structured planning, progress visibility, social accountability, and AI-driven guidance. StepHabit combines modern web technologies with intelligent feedback mechanisms to deliver a productivity platform that supports sustainable habit formation and long-term personal growth.

2.2 Technology Stack and Frameworks

The StepHabit platform utilizes a modern, modular technology stack designed to ensure performance, scalability, maintainability, and ease of future extension across all layers of the system. Each technology was selected to support rapid development while maintaining architectural clarity and long-term sustainability.

Frontend React (Vite): A modern JavaScript library used to build responsive and interactive user interfaces. Vite is employed as the build tool to enable fast development cycles and optimized production bundles. CoreUI: A component-based UI framework used to implement consistent layouts, navigation structures, dashboards, and data visualization elements across the application.

Backend Node.js with Express: Node.js serves as the runtime environment for the backend, while Express provides a lightweight and flexible framework for handling HTTP requests, routing, and middleware composition. Sequelize ORM: Used to manage relational data interactions with PostgreSQL, providing model definitions, associations, and schema synchronization. JWT (JSON Web Tokens): Employed for secure authentication and authorization, ensuring that protected API endpoints are accessible only to authenticated users. Bcrypt: Used for secure password hashing and credential protection. Socket-Based Messaging (Planned): The architecture supports real-time features such as notifications and messaging, allowing future integration of WebSocket-based communication if required.

Database PostgreSQL: A robust open-source relational database used to store structured data, including user profiles, habits, tasks, schedules, progress logs, achievements, notifications, and social relationships. PostgreSQL's reliability and relational capabilities support data integrity and complex querying needs.

AI Integration LangChain with Large Language Models (Anthropic): Used to implement AI-assisted habit coaching, including habit plan generation, idea refinement, and contextual guidance. LangChain provides an abstraction layer that simplifies prompt management, model configuration, and future AI provider replacement.

Environment-Based Configuration: Sensitive credentials and environment-specific variables are managed through environment files, supporting secure and flexible configuration across development and production environments.

2.3 Development and DevOps Tools

The development of the StepHabit platform relied on a set of modern development and DevOps tools to support efficient implementation, testing, collaboration, and deployment. These tools contributed to code quality, system reliability, and developer productivity throughout the project lifecycle.

Visual Studio Code (VS Code): Used as the primary integrated development environment for both frontend and backend development. VS Code provides rich language support, debugging tools, and extensions that streamline full-stack development.

Postman: Employed extensively during backend development for testing RESTful API endpoints, validating request and response payloads, and debugging authentication and business logic workflows.

Git and GitHub: Used for version control, source code management, and collaboration. Git enables systematic tracking of code changes, while GitHub supports repository hosting, issue tracking, and pull-request-based development workflows.

Docker and Docker Desktop: Used to containerize the application's backend services, frontend client, and database, ensuring consistent environments across local development and deployment. Docker Compose simplifies orchestration of multi-container setups.

Navicat: Utilized as a database administration and visualization tool for managing the

PostgreSQL database, inspecting tables and relationships, and validating data integrity during development and testing.

ESLint and Prettier: Applied to enforce consistent coding standards, detect potential issues early, and maintain readability and maintainable code across the project.

Email and Notification Monitoring Tools: Used to test and monitor email-based features such as account verification and notification delivery, ensuring reliability and correctness of user-facing communication workflows.

API Documentation Tools (Swagger UI): Used to document and interactively test API endpoints, improving transparency of the API surface and simplifying development, debugging, and future system integration.

3 Database Design

3.1 Database Choice

PostgreSQL was chosen for this project because of its excellent support for relational data, adherence to ACID principles, and support for TypeORM integration. PostgreSQL's support for complex querying and indexing provides an effective solution to managing the complex relationships between users, posts, and interactions (likes, comments).

3.2 Entity Relationship Diagram (ERD)

ERD

3.3 Database Tables

Users Table

- **Table Name:** users
- **Fields:**
  - id (SERIAL, Primary Key)
  - name (VARCHAR(100), Not Null)
  - email (VARCHAR(150), Unique, Not Null)
  - password (VARCHAR(200), Not Null)
  - age (INT, Nullable)
  - gender (VARCHAR(20), Nullable)
  - bio (TEXT, Nullable)
  - avatar (VARCHAR(255), Nullable)
  - primary_goal (VARCHAR(150), Nullable)
  - focus_area (VARCHAR(120), Nullable)
  - experience_level (VARCHAR(60), Nullable)
  - daily_commitment (VARCHAR(60), Nullable)
  - support_preference (VARCHAR(120), Nullable)
  - motivation_statement (TEXT, Nullable)
  - created_at (TIMESTAMP, Default: CURRENT_TIMESTAMP)
- **Relationships:**
  - One-to-one with user_settings
  - One-to-many with habits, tasks, busy_schedules, progress, notifications
  - One-to-many with assistant_memories, calendar_integrations, calendar_events
  - Many-to-many with achievements (via user_achievements)

  - Many-to-many with group_challenges (via user_group_challenges)
  - Self-referential relationship via friends

User Settings Table

- **Table Name:** user_settings
- **Fields:**
  - id (SERIAL, Primary Key)
  - user_id (INT, FK → users.id, Unique, Not Null)
  - timezone (VARCHAR(80), Default: UTC)
  - daily_reminder_time (VARCHAR(10), Nullable)
  - weekly_summary_day (VARCHAR(16), Default: Sunday)
  - email_notifications (BOOLEAN, Default: TRUE)
  - push_notifications (BOOLEAN, Default: FALSE)
  - share_activity (BOOLEAN, Default: TRUE)
  - theme (VARCHAR(20), Default: light)
  - ai_tone (VARCHAR(30), Default: balanced)
  - support_style (VARCHAR(30), Default: celebrate)
  - google_calendar (BOOLEAN, Default: FALSE)
  - apple_calendar (BOOLEAN, Default: FALSE)
  - fitness_sync (BOOLEAN, Default: FALSE)
  - created_at (TIMESTAMP, Default: CURRENT_TIMESTAMP)
- **Relationships:**
  - One-to-one with users

Registration Verifications Table

- **Table Name:** registration_verifications
- **Fields:**
  - id (SERIAL, Primary Key)
  - email (VARCHAR(150), Unique, Not Null)
  - code_hash (VARCHAR(200), Not Null)
  - payload (JSONB, Not Null)
  - expires_at (TIMESTAMP, Not Null)

Password Resets Table

- **Table Name:** password_resets

- **Fields:**
  - id (SERIAL, Primary Key)
  - email (VARCHAR(150), Unique, Not Null)
  - code_hash (VARCHAR(200), Not Null)
  - expires_at (TIMESTAMP, Not Null)

Habits Table

- **Table Name:** habits
- **Fields:**
  - id (SERIAL, Primary Key)
  - user_id (INT, FK → users.id, Not Null)
  - title (VARCHAR(100), Not Null)
  - description (TEXT, Nullable)
  - category (VARCHAR(50), Nullable)
  - target_reps (INT, Nullable)
  - is_daily_goal (BOOLEAN, Default: FALSE)
  - created_at (TIMESTAMP, Default: CURRENT_TIMESTAMP)
- **Relationships:**
  - Many-to-one with users
  - One-to-many with schedules and progress

Schedules Table

- **Table Name:** schedules
- **Fields:**
  - id (SERIAL, Primary Key)
  - habit_id (INT, FK → habits.id, Not Null)
  - user_id (INT, FK → users.id, Not Null)
  - day (DATE, Not Null)
  - starttime (TIME, Not Null)
  - endtime (TIME, Nullable)
  - enddate (DATE, Nullable)
  - repeat (VARCHAR(50), Default: daily)
  - customdays (VARCHAR(100), Nullable)
  - notes (TEXT, Nullable)

  - created_at (TIMESTAMP)
  - updated_at (TIMESTAMP)

Tasks Table

- **Table Name:** tasks
- **Fields:**
  - id (SERIAL, Primary Key)
  - user_id (INT, FK → users.id, Not Null)
  - name (VARCHAR(255), Not Null)
  - duration_minutes (INT, Default: 60)
  - min_duration_minutes (INT, Nullable)
  - max_duration_minutes (INT, Nullable)
  - split_up (BOOLEAN, Default: FALSE)
  - hours_label (VARCHAR(120), Nullable)
  - schedule_after (TIMESTAMP, Nullable)
  - due_date (TIMESTAMP, Nullable)
  - color (VARCHAR(20), Nullable)
  - status (VARCHAR(20), Default: pending)
  - created_at (TIMESTAMP, Default: CURRENT_TIMESTAMP)

Busy Schedules Table

- **Table Name:** busy_schedules
- **Fields:**
  - id (SERIAL, Primary Key)
  - user_id (INT, FK → users.id, Not Null)
  - title (VARCHAR(255), Not Null)
  - day (DATE, Not Null)
  - starttime (TIME, Not Null)
  - endtime (TIME, Nullable)
  - enddate (DATE, Nullable)
  - repeat (VARCHAR(50), Default: daily)
  - customdays (VARCHAR(100), Nullable)
  - notes (TEXT, Nullable)
  - created_at (TIMESTAMP)
  - updated_at (TIMESTAMP)

Progress Table

- **Table Name:** progress
- **Fields:**
  - id (SERIAL, Primary Key)
  - user_id (INT, FK → users.id, Not Null)
  - habit_id (INT, FK → habits.id, Not Null)
  - status (VARCHAR(50), Not Null)
  - reflected_reason (TEXT, Nullable)
  - progress_date (DATE, Default: CURRENT_DATE)
  - created_at (TIMESTAMP)

Achievements Table

- **Table Name:** achievements
- **Fields:**
  - id (SERIAL, Primary Key)
  - title (VARCHAR(100), Not Null)
  - description (TEXT, Not Null)
  - created_at (TIMESTAMP)
- **Relationships:**
  - Many-to-many with users (via user_achievements)

User Achievements Table

- **Table Name:** user_achievements
- **Fields:**
  - id (SERIAL, Primary Key)
  - user_id (INT, FK → users.id, Not Null)
  - achievement_id (INT, FK → achievements.id, Not Null)
  - achieved_at (TIMESTAMP)
- **Relationships:**
  - Many-to-one with users
  - Many-to-one with achievements

3.4 Data Security Measures

The StepHabit platform implements multiple layers of security to protect user data, ensure privacy, and maintain system integrity across authentication, authorization, and data processing workflows.

Password Hashing. User passwords are securely hashed using the `bcrypt` algorithm before storage. This approach ensures that raw credentials are never persisted in the database and protects user accounts in the event of a data breach.

JWT-Based Authentication. Authentication is managed using JSON Web Tokens (JWT), which are issued upon successful login and attached to subsequent API requests. Tokens are validated on protected endpoints to ensure that only authenticated users can access sensitive resources.

Two-Step Verification. StepHabit employs a code-based email verification mechanism during registration and account recovery workflows. Verification codes are time-limited and must be validated before account activation or password reset is completed.

Access Control and Authorization. The system enforces strict ownership-based access control. Users can only view and modify resources they own, such as habits, tasks, schedules, and progress logs. Public features are limited to non-sensitive content, while all core functionality requires authentication. This model ensures separation between authenticated users and unauthenticated visitors.

Rate Limiting and Throttling. To protect against brute-force attacks, verification workflows are rate-limited. Each user is allowed a maximum of three verification code attempts within a one-minute window. If this threshold is exceeded, the current verification code becomes invalid and a new code must be issued.

HTTPS and CORS Policies. All API communication is secured using HTTPS to protect data in transit. Cross-Origin Resource Sharing (CORS) policies are configured to restrict access to trusted frontend origins and prevent unauthorized cross-site requests.

Field-Level Validation and Sanitization. All incoming API payloads undergo strict validation and sanitization at the controller level. This prevents common injection attacks and ensures that only well-formed data is processed by the application.

Privacy Controls and Visibility Boundaries. StepHabit enforces strict visibility rules for user-generated content. Habits, tasks, progress logs, and AI-generated insights are private by default and accessible only to their respective owners unless explicitly shared through controlled social features.

Transactional Integrity. The application ensures atomic data operations by grouping dependent database actions within transactional boundaries. This guarantees that multi-step operations—such as creating habits, schedules, progress records, or social relationships—are either fully completed or fully rolled back in the event of an error, preserving database consistency.

4 Backend Endpoints

4.1 Habit Endpoints

Method	Path	Description	Auth Required
GET	/habits	Retrieve all habits for the authenticated user	Yes
GET	/habits/:id	Retrieve a habit by its identifier	Yes
POST	/habits	Create a new habit	Yes
PUT	/habits/:id	Update an existing habit	Yes
DELETE	/habits/:id	Delete a habit and related data	Yes
POST	/habits/ai/plan	Generate AI-based habit plan	Yes
POST	/habits/ai/rewrite	Rewrite habit description using AI	Yes

4.2 Task Endpoints

Method	Path	Description	Auth Required
GET	/tasks	Retrieve all tasks for the user	Yes
POST	/tasks	Create a new task	Yes
PUT	/tasks/:id	Update task details	Yes
PATCH	/tasks/:id/status	Update task status	Yes
DELETE	/tasks/:id	Delete a task	Yes

4.3 Progress Tracking Endpoints

Method	Path	Description	Auth Required
GET	/progress	Retrieve habit progress logs	Yes
POST	/progress	Log habit completion	Yes
DELETE	/progress/:id	Delete a progress record	Yes

4.4 User Authentication Endpoints

Method	Path	Description	Auth Required
POST	/users/register	Register a new user	No
PATCH	/users/register/verify	Complete registration via code	No
POST	/users/login	Login with email and password	No
PATCH	/users/login/verify	Login using verification code	No
GET	/users/me	Get authenticated user profile	Yes
PUT	/users/:id	Update user profile	Yes
DELETE	/users	Delete user account	Yes
POST	/users/logout	Logout authenticated user	Yes

POST	/users/forgot-password	Request password reset	No
PATCH	/users/reset-password	Reset password via code	Yes

4.5 Notification Endpoints

Method	**Path**	**Description**	**Auth Required**
GET	/notifications	Retrieve user notifications	Yes
POST	/notifications	Create a notification	Yes
PATCH	/notifications/mark-all-read	Mark all notifications as read	Yes
PATCH	/notifications/:id/read	Mark a notification as read	Yes

4.6 Messaging Social Endpoints

Method	**Path**	**Description**	**Auth Required**
GET	/messages	Retrieve chat messages	Yes
POST	/messages	Send a direct message	Yes
POST	/group-challenges	Create a group challenge	Yes
GET	/group-challenges	List group challenges	Yes
POST	/group-challenges/:id/join	Join a challenge	Yes

4.7 WebSocket Endpoint

WebSocket URL	ws://¡HOST¿:¡PORT¿/notification-message
Namespace	/notification-message
Transport	WebSocket
Authentication	JWT (Authorization Header)

4.8 API Tools and Documentation

To support development, testing, and long-term maintainability, the StepHabit platform provides comprehensive API documentation and testing tools. These tools allow developers and reviewers to explore endpoint behavior, request formats, and response structures in a transparent and interactive manner.

Tool	**Description**	**Access URL**
Swagger UI	Interactive API documentation that exposes all available endpoints, including request parameters, authentication requirements, and response schemas. It enables real-time testing of API calls directly from the browser.	`/api#`

Postman Collection	A curated Postman collection used for manual API testing and debugging during development. It includes preconfigured requests for authentication, habits, tasks, notifications, AI coaching, and social features.	`Postman Workspace`

5 Application and functionality

5.1 User Registration Process

The user registration workflow in StepHabit is designed to ensure account integrity, data validity, and protection against unauthorized access. The process is divided into three structured stages, each incorporating validation rules and security mechanisms to maintain platform reliability.

Account Information Entry

The registration process begins with the user providing the following required information:

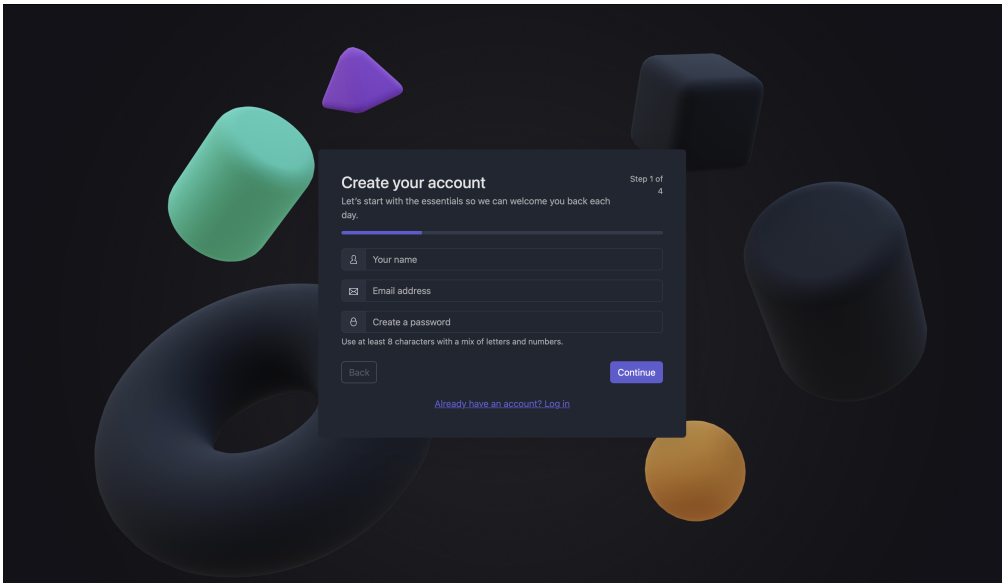

Figure 1: StepHabit registration form for entering account information

- Username
- Email address
- Password

The system enforces the following validation constraints at this stage:

- The username must be unique across the platform.
- The email address must follow a valid email format (e.g., `user@example.com`).
- Duplicate email registrations are not permitted.
- The password must contain a minimum of six characters to ensure basic security.

If any validation rule fails, the system returns descriptive error messages and prevents progression to the next step until valid input is provided.

Goal and Focus Area Selection

After the initial account credentials are validated, the user proceeds to the second onboarding step, where StepHabit begins shaping a personalized experience. This step is designed to capture the user's motivation and preferred focus areas, which directly influence habit recommendations, dashboard configuration, and AI-assisted coaching behavior.

The user is first asked to identify what brought them to the platform. Available options include building consistency, boosting energy, improving focus and clarity, or achieving balance and wellbeing. The selected option represents the user's primary motivation and is stored as part of the user profile.

Next, the user selects an initial focus area, such as mindfulness, fitness, productivity, or self-care. This selection determines the category of habits and templates that will be prioritized during early usage of the platform.

All selections made during this step are persisted in the database and later used by the scheduling engine and AI coaching services to personalize recommendations.

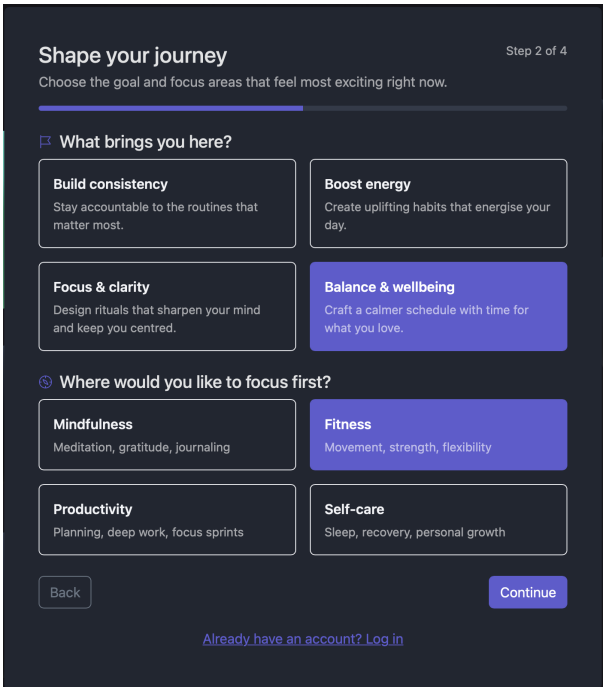

Figure 2: Selection of primary motivation and personal goal

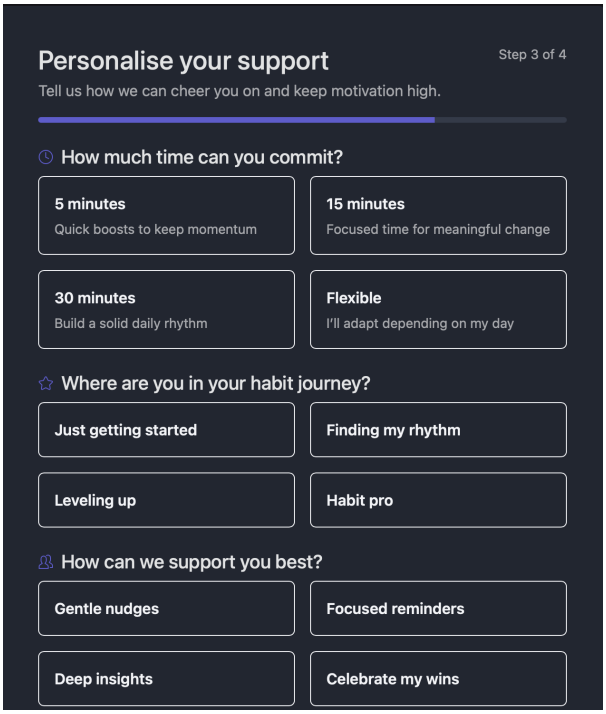

Figure 3: Selection of initial focus area

Commitment Level and Support Preferences

In the third onboarding step, StepHabit collects information related to time commitment, experience level, and preferred support style. This data allows the system to adapt habit intensity, reminder frequency, and AI feedback tone to the user's lifestyle and expectations.

The user specifies how much time they are willing to dedicate daily, choosing between predefined options such as five minutes, fifteen minutes, thirty minutes, or a flexible schedule. This value is

stored and later used to scale habit goals and task durations.

The user then indicates their current stage in the habit-building journey, ranging from beginner to advanced. This selection helps the AI assistant calibrate the complexity of recommendations and avoid overwhelming new users.

Finally, the user selects a preferred support style, such as gentle nudges, focused reminders, deep insights, or celebratory feedback. This preference is persisted in the database and influences notification tone and AI-generated coaching responses.

An optional motivational input allows the user to express a personal reason for starting, which is stored as a short textual motivation statement and may be referenced by the AI assistant in future interactions.

Step 3: Verification Code Validation

Once the initial account details are successfully submitted, StepHabit initiates a verification process to confirm ownership of the provided email address. A time-limited verification code is sent to the user via email.

The following security measures are applied: This mechanism protects against brute-force attacks and unauthorized account creation attempts.

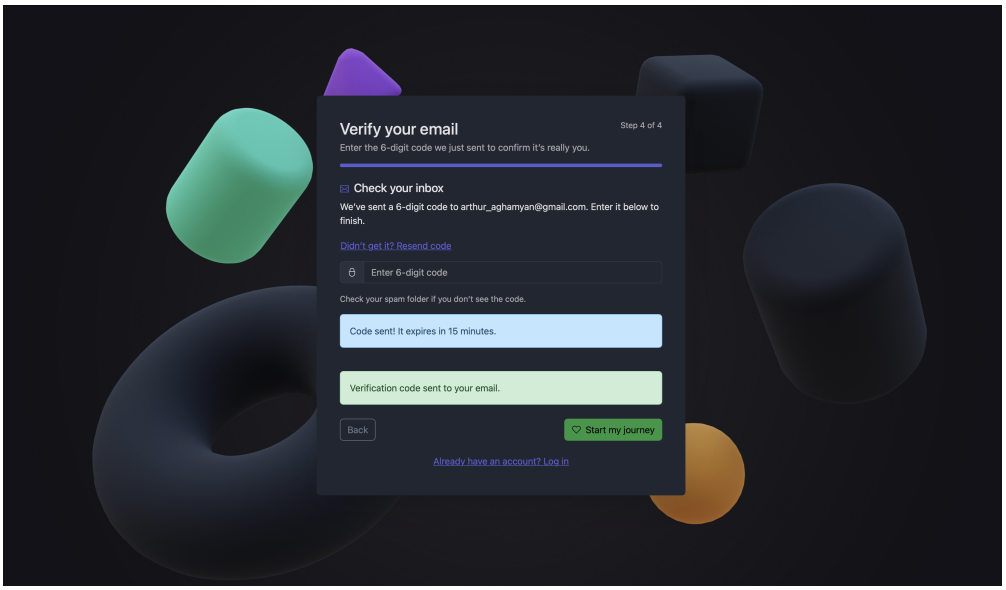

Figure 4: Optional profile initialization for personalized habit coaching

Step 3: Profile Initialization

After successful verification, the user account is activated and the user may optionally complete initial profile preferences. These preferences help tailor the StepHabit experience and AI-assisted coaching features.

Optional profile data may include:

- Primary goal and focus area
- Daily commitment level

- Preferred support and coaching style

Registration Completion

Upon completing all required steps, the user is redirected to the main dashboard and granted full access to the platform's core functionality, including habit creation, task planning, scheduling, and AI-assisted coaching.

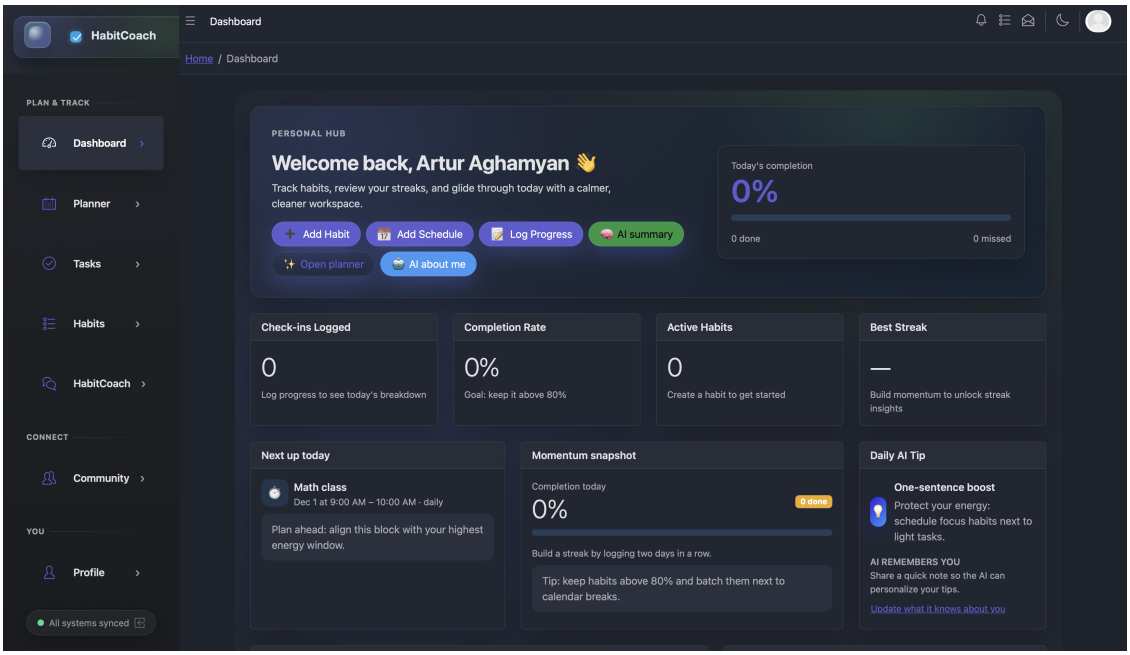

Figure 5: Successful registration and initial access to the StepHabit dashboard

Dashboard (Personal Hub)

The Dashboard is the primary landing screen after authentication. It provides a high-level snapshot of the user's daily progress, shortcuts to core actions, and AI-assisted guidance. The page is designed to minimize friction by letting users create habits, schedule time blocks, and log progress directly from the main hub.

Key elements on this screen include: (i) the *Personal Hub* welcome panel with action buttons (*Add Habit*, *Add Schedule*, *Log Progress*, *AI Summary*), (ii) a *Today's completion* progress indicator, (iii) summary KPI cards (check-ins, completion rate, active habits, best streak), and (iv) contextual widgets such as *Next up today*, *Momentum snapshot*, and a *Daily AI Tip*. Together, these components guide the user toward consistent daily execution while maintaining visibility into progress.

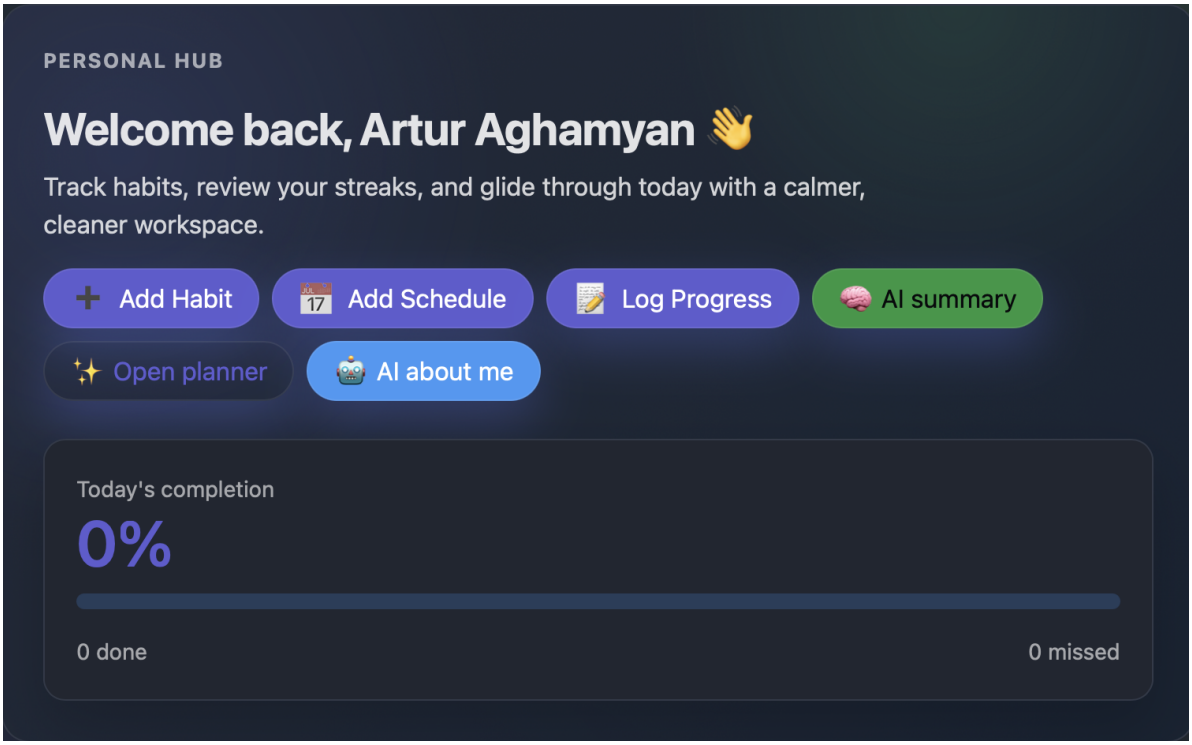

Figure 6: StepHabit Dashboard (Personal Hub) with progress indicators, quick actions, and AI guidance

5.2 Planner Module

The Planner is the central scheduling component of StepHabit. It enables users to design structured daily and weekly routines by combining habit-related time blocks, busy periods, and externally imported calendar events into a single, unified view. This section plays a critical role in bridging long-term habit goals with concrete time allocation. At the top of the Planner, the system presents a high-level status overview, including the current focus date, overall planning health, and calendar synchronization status. Users can create new time blocks, synchronize external calendars, or navigate directly to their saved schedules.

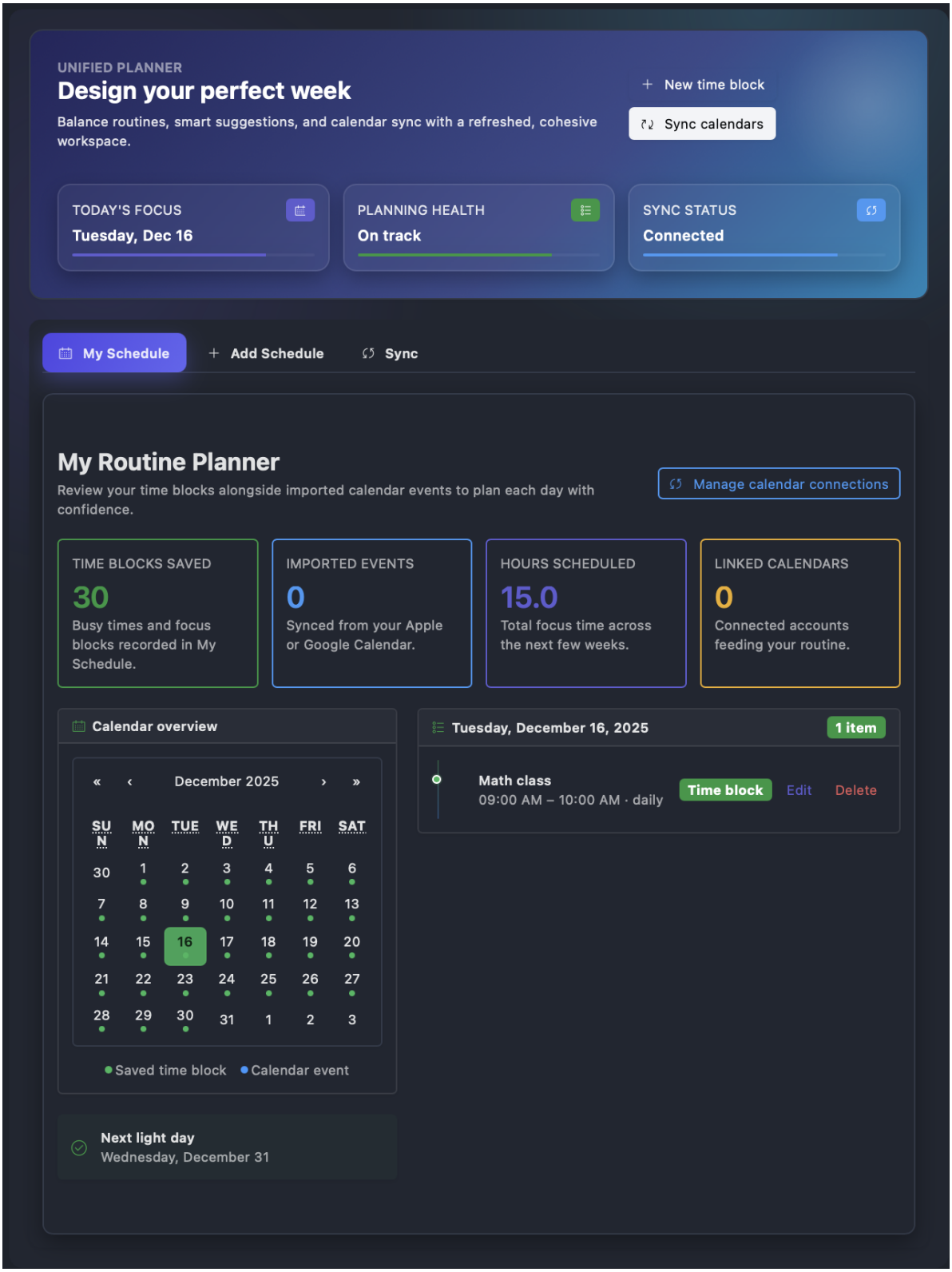

Figure 7: Unified Planner overview with planning health indicators and schedule controls

The core of the Planner is the routine view, which combines a monthly calendar overview with a

detailed daily timeline. Saved time blocks and imported calendar events are visually distinguished, allowing users to quickly identify free windows and scheduling conflicts. Selecting a specific day reveals all associated routines and busy periods for that date.

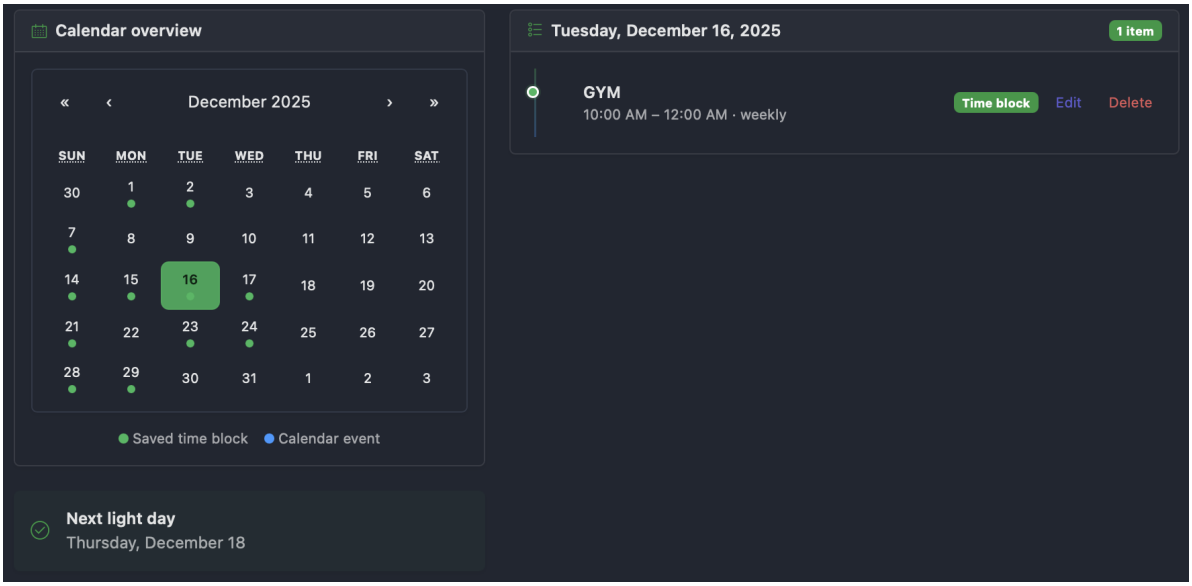

Figure 8: Calendar overview and daily schedule with saved time blocks

Users can add new schedule entries through the *Add Schedule* interface. Schedules can either be linked to a specific habit or marked as busy events. Linking a schedule to a habit allows the AI assistant to understand when routines are expected to occur, enabling more accurate habit suggestions and progress tracking. Busy events, on the other hand, inform the system about unavailable time windows and are not treated as completion-based activities.

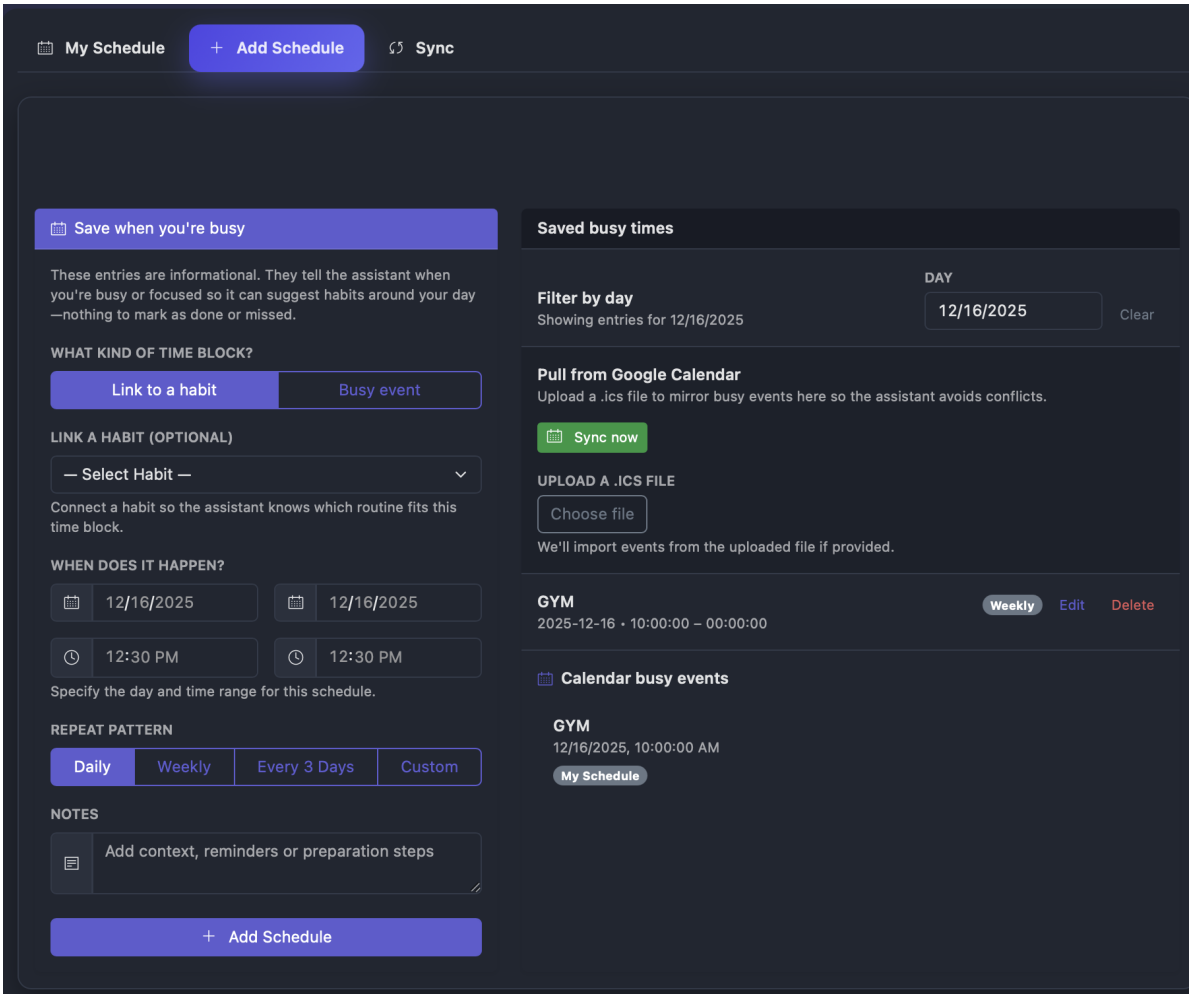

Figure 9: Creating and managing schedule entries and busy time blocks

The Planner also supports calendar synchronization through external providers such as Google Calendar. Users can upload calendar data using `.ics` files or connect accounts directly. Imported events are stored separately from habits and schedules but are displayed alongside them to prevent conflicts and enable smarter time allocation.

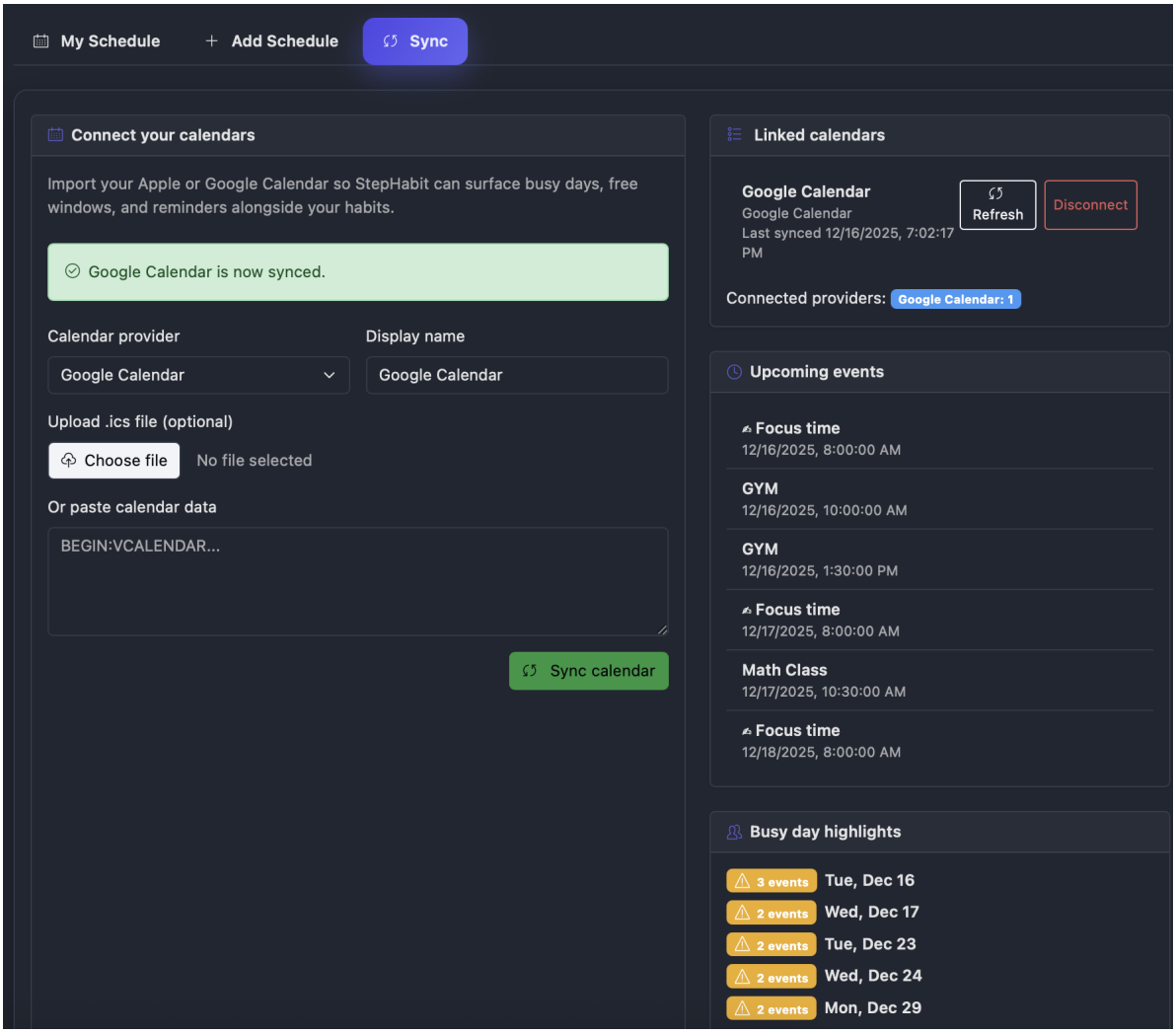

Figure 10: External calendar integration and upcoming event synchronization

All planner data, including schedules, busy blocks, and calendar events, is persisted in the database and associated with the authenticated user. This allows the AI assistant, notification system, and progress analytics modules to operate on a shared, consistent view of the user's time and routines.

5.3 Tasks Module

The Tasks module allows users to capture, organize, and prepare actionable tasks before they are scheduled into the planner. This separation enables users to think clearly about task scope, duration, and deadlines without immediately committing to a specific time block.

Task Board Overview

The Task Board provides a centralized list of all user-created tasks. From this interface, users can view their tasks, create new ones, and track progress at a high level.

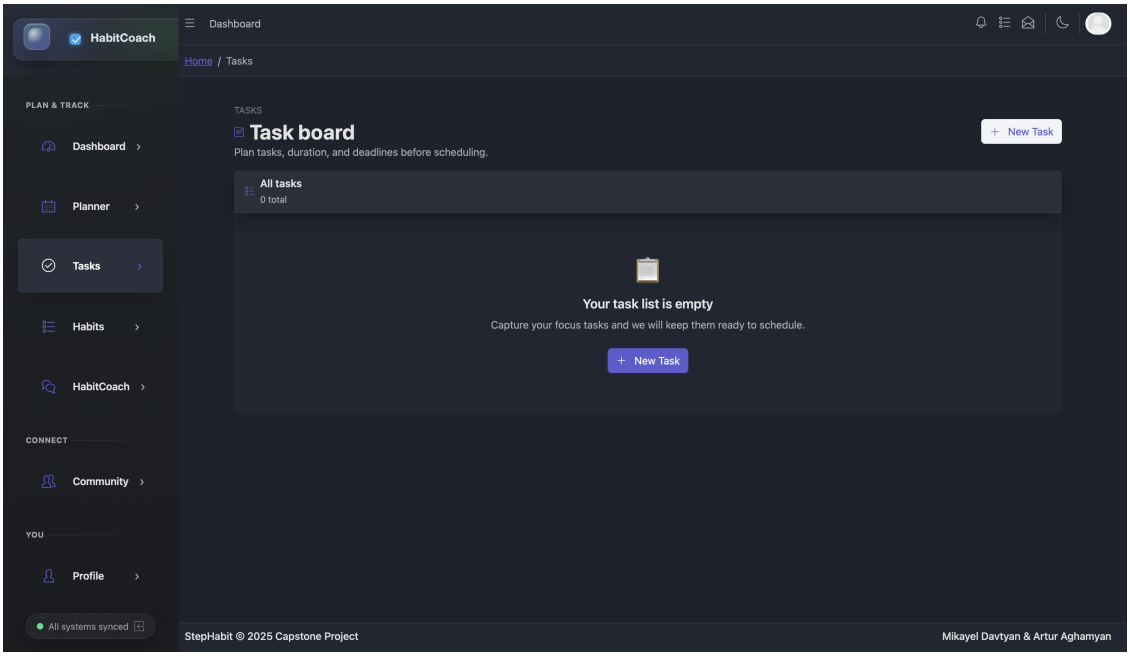

Figure 11: Initial task board view when no tasks are present

When no tasks exist, the system displays a clear call-to-action prompting the user to create their first task.

Creating a New Task

Users can add a new task by clicking the *New Task* button, which opens a modal form. This form collects structured task metadata required for intelligent scheduling and planning.

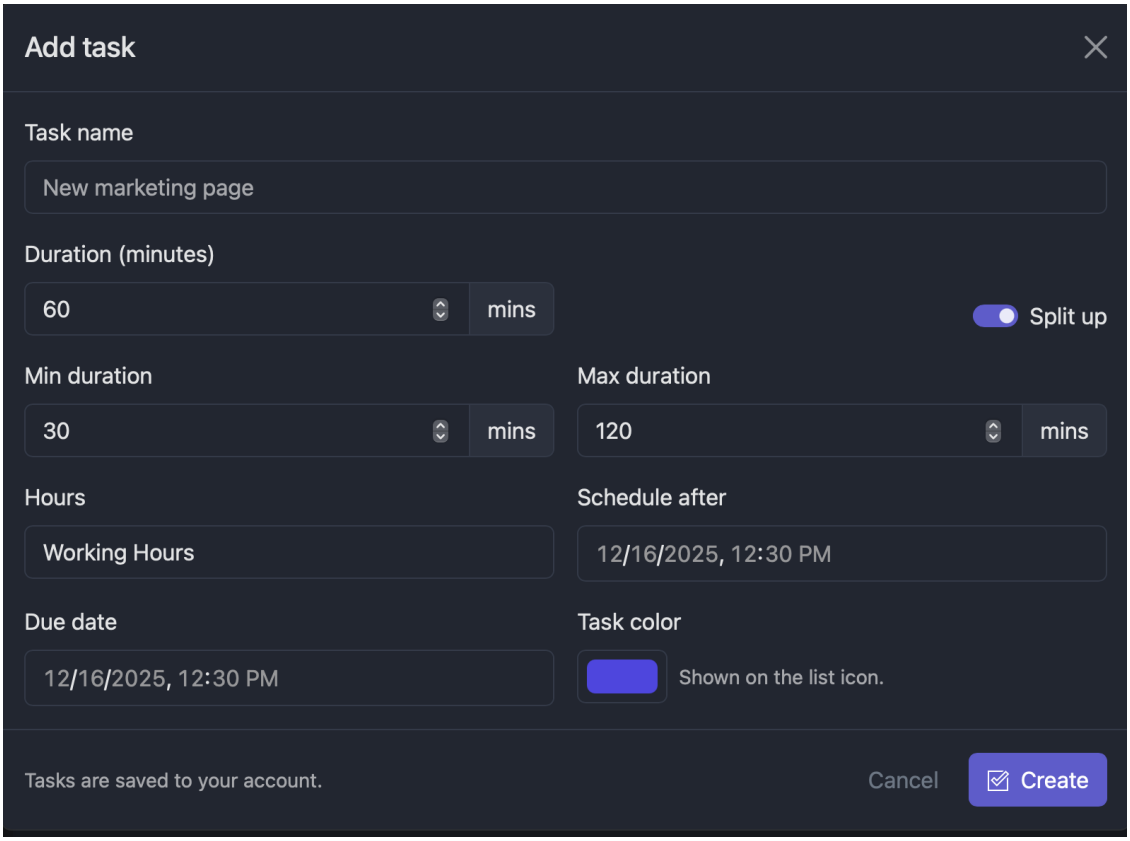

Figure 12: Task creation modal with duration and scheduling constraints

The task creation form includes the following configurable fields:

- Task name
- Estimated duration (in minutes)
- Minimum and maximum allowed duration
- Optional task splitting for flexible scheduling
- Preferred working hours
- Optional scheduling constraint (schedule after)
- Due date
- Visual task color for identification

Once submitted, the task is persisted and becomes visible in the task board.

Task List and Feedback

After a task is created, the system provides immediate visual feedback confirming successful creation. Tasks are displayed in a reorderable list, allowing users to prioritize items manually.

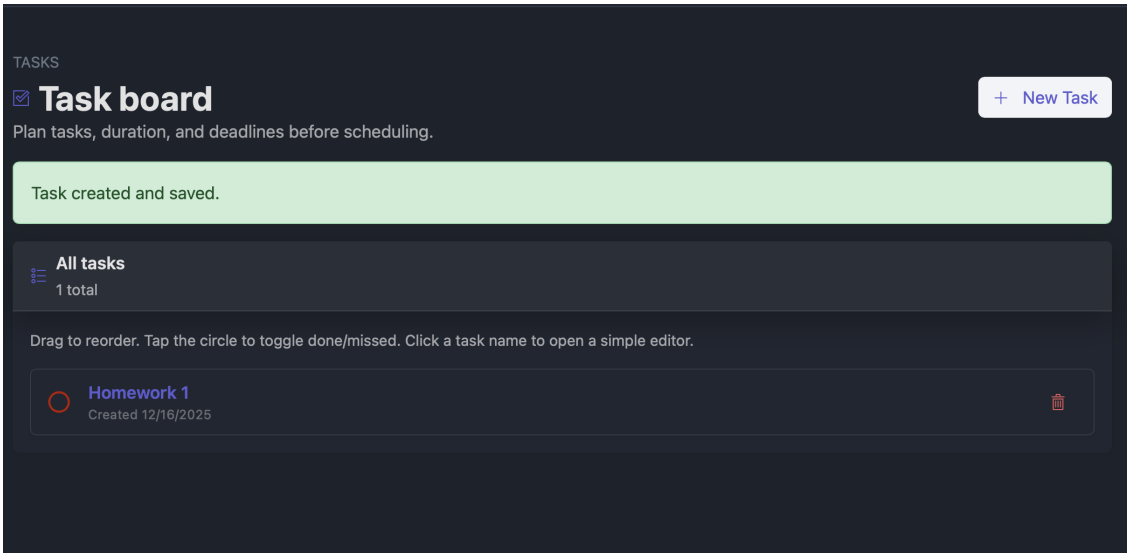

Figure 13: Task board after successful task creation

Tasks can be reordered via drag-and-drop, marked as completed or missed, or opened for further editing.

Task Details and Checklists

Clicking on a task opens a detailed editor where users can enrich the task with descriptions and actionable checklists. This enables breaking down complex tasks into smaller, trackable steps.

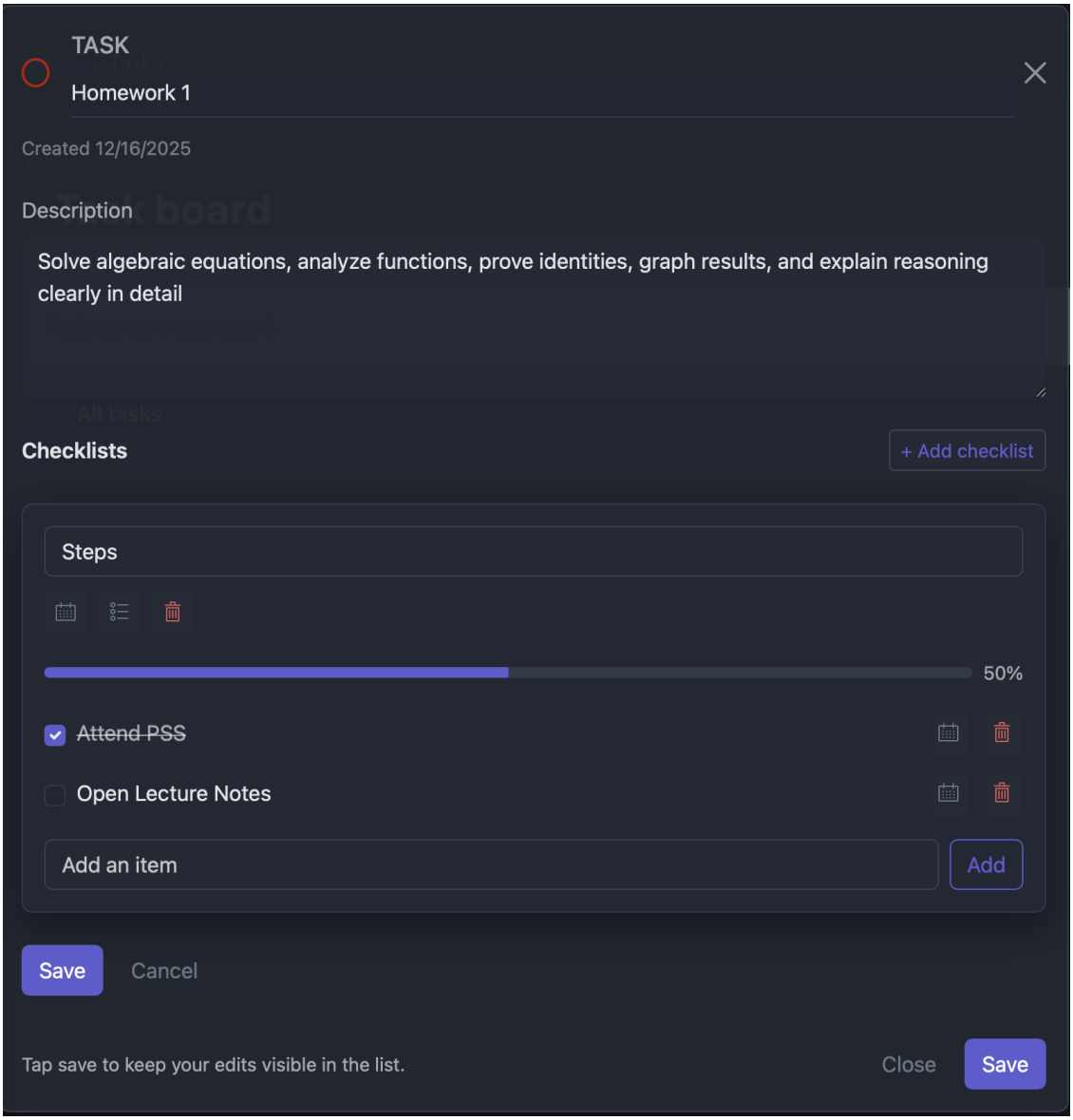

Figure 14: Task detail view with checklist and progress tracking

Each checklist item can be individually marked as complete, with the interface displaying a real-time progress indicator to reflect overall task completion. This design supports both lightweight task tracking and deeper task decomposition, depending on user needs.

5.4 Habits Module

The Habits module is the core component of the StepHabit platform. It allows users to create, manage, track, and analyze daily habits while receiving structured insights and AI-supported guidance. The interface is designed to remain calm, minimal, and data-driven, encouraging consistency and long-term behavior change.

Habits Overview

The Habits overview page serves as the central hub for habit-related activity. It provides a summary of the user's current progress, including weekly win rate, current streak, and the total number of active habits. From this page, users can navigate to habit creation, the habit library, progress analytics, AI coaching, insights, and historical logs.

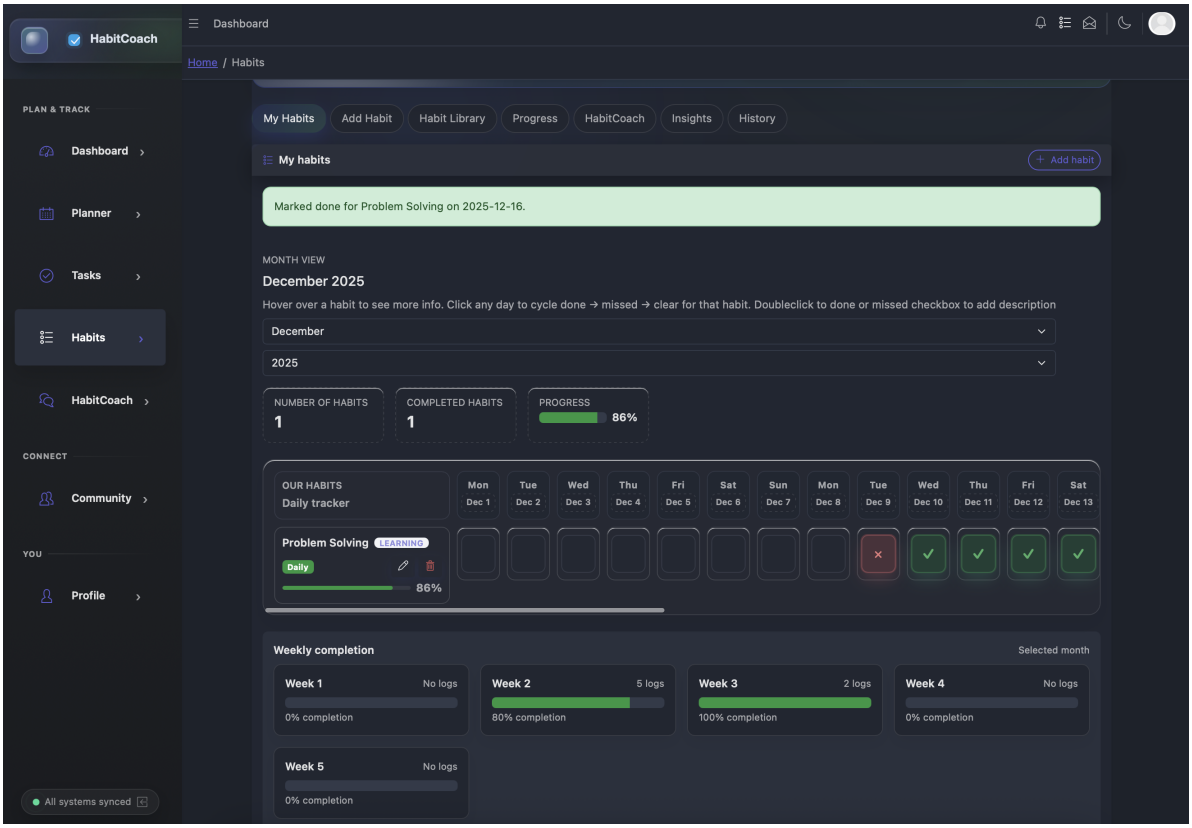

Figure 15: Habits overview dashboard showing summary statistics and navigation tabs

Habit Creation

Users can create a new habit using the structured habit creation form. This form collects essential information such as the habit title, description, category, and target repetitions. An AI-powered habit rewriter is also available, allowing users to input rough ideas and receive refined habit definitions.

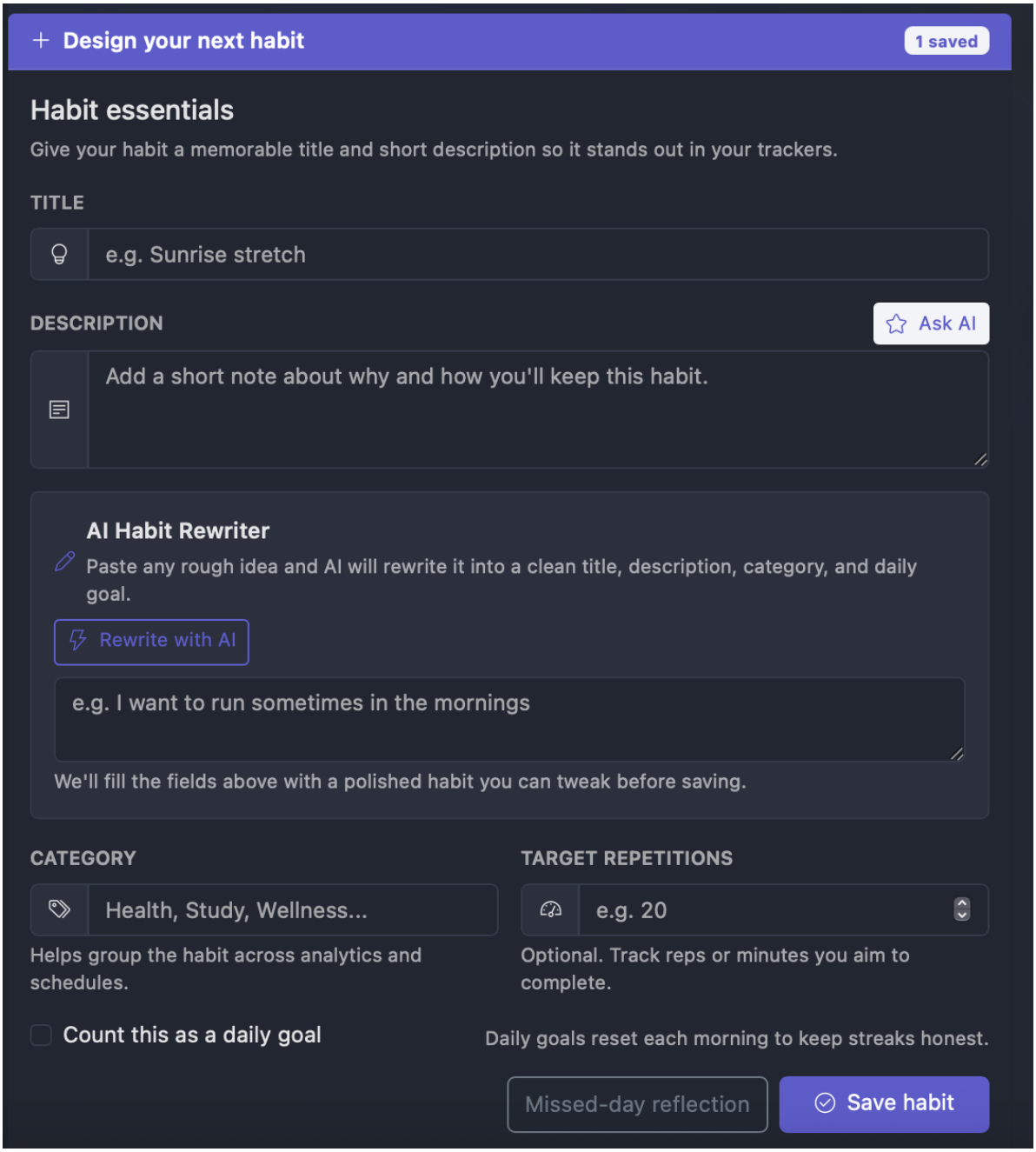

Figure 16: Habit creation interface with AI habit rewriting and live preview

Habit Library

The Habit Library provides a curated collection of pre-designed habits that users can add to their routine with a single click. Habits are categorized by focus area, difficulty, and time of day. Popular and trending habits are highlighted to assist users in discovery.

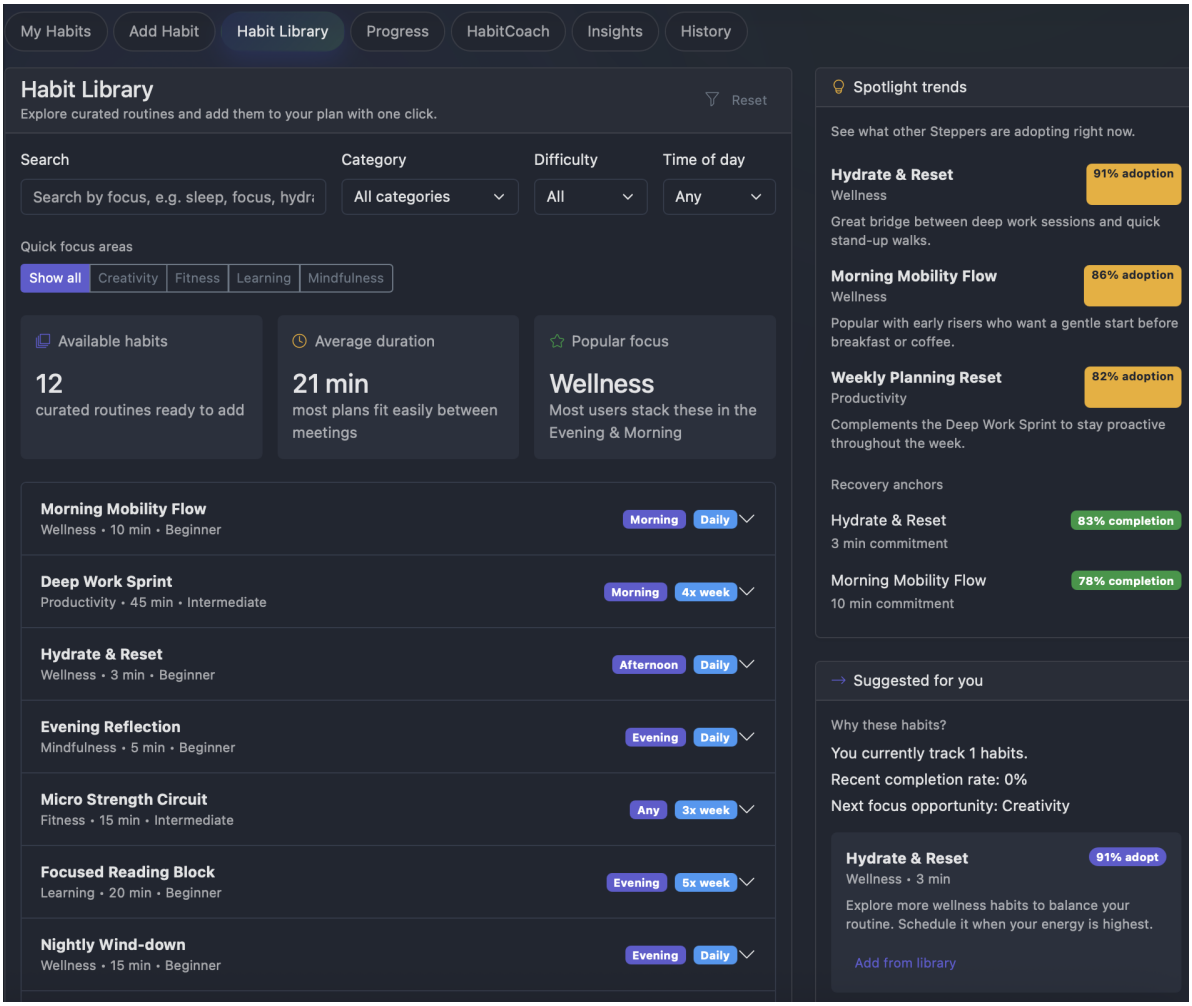

Figure 17: Habit library with filters, curated habits, and trending recommendations

Progress Tracking

The Progress section visualizes habit performance over time. Users can analyze weekly, monthly, or yearly completion rates, monitor streaks, and identify their most productive days. This data-driven view reinforces consistency and helps users adjust their routines.

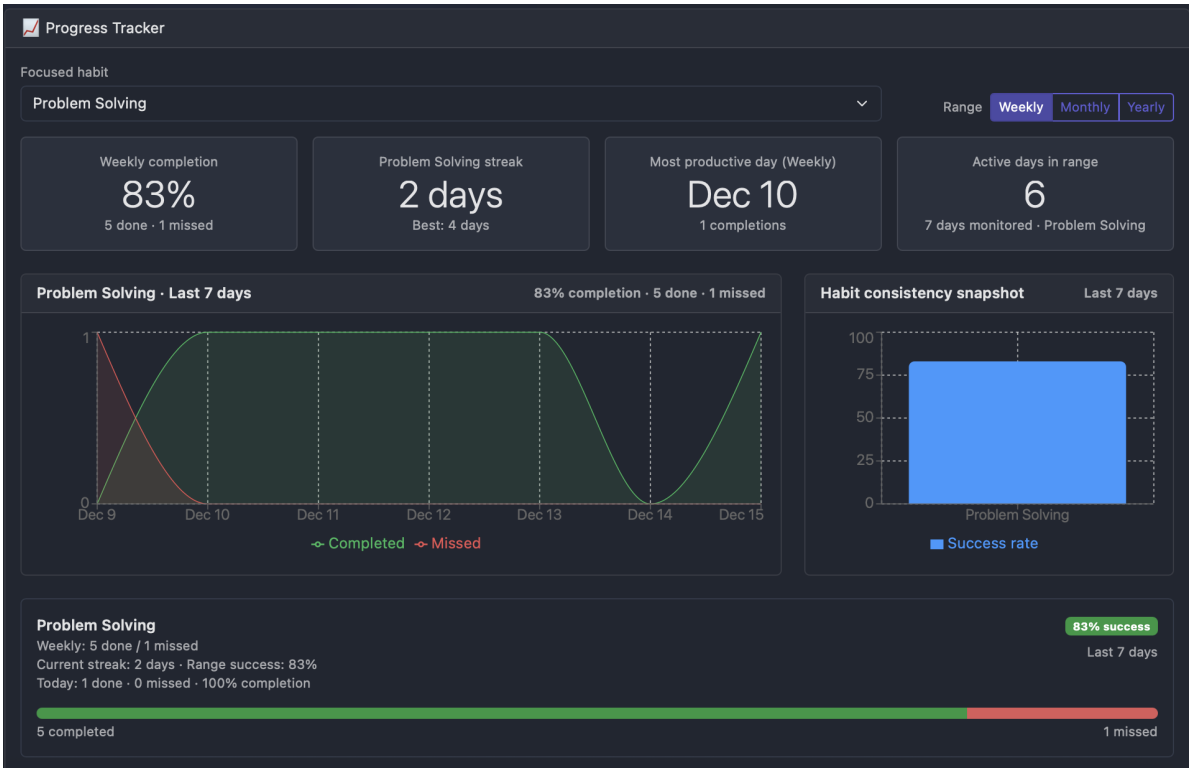

Figure 18: Habit progress tracker with completion rates and streak analysis

Insights and Analytics

The Insights page transforms logged habit data into meaningful feedback. It highlights high-performing habits, identifies habits needing attention, and provides recommendations for optimization. Insights are refreshed dynamically as users log progress.

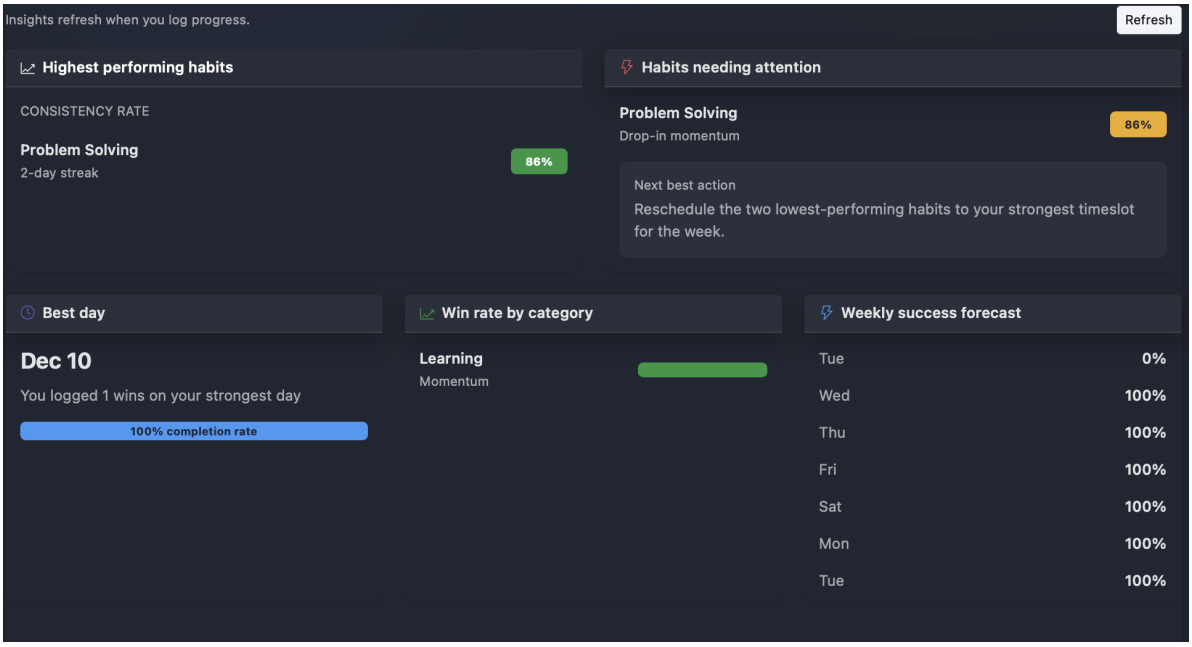

Figure 19: Insights dashboard showing habit performance analysis and recommendations

Habit History

The History section maintains a chronological record of the user's habit check-ins, including optional reflections. Users can filter by habit and date range or export data for external analysis.

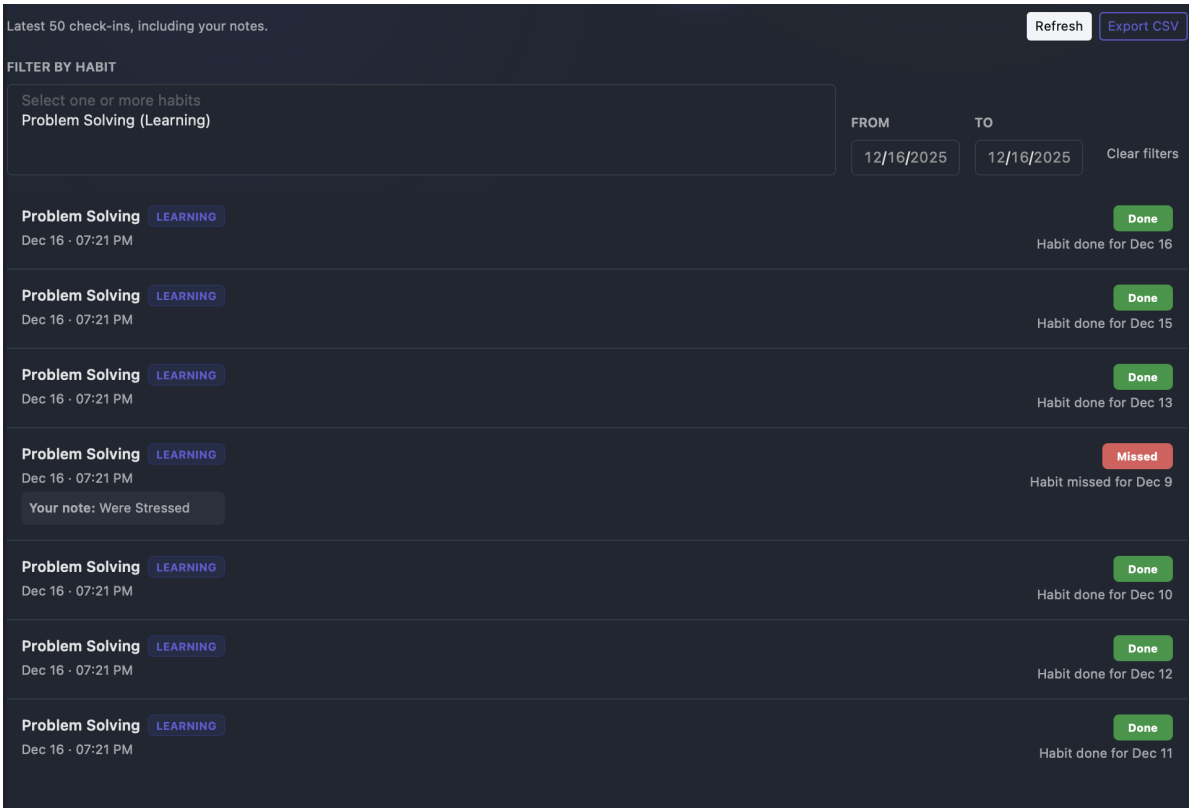

Figure 20: Habit history view with filters and export functionality

5.5 HabitCoach(AI Assistant) Module

5.6 HabitCoach: AI-Powered Personal Assistant

HabitCoach is an intelligent conversational assistant integrated deeply into the StepHabit platform. Unlike a traditional chatbot, HabitCoach has full contextual awareness of the user's profile, habits, schedules, tasks, calendar events, and historical activity. This allows it to act as a personalized habit and planning assistant rather than a generic question–answer system. HabitCoach continuously follows the user's conversation context and can safely access all relevant database tables, including habits, schedules, tasks, progress logs, notifications, and calendar integrations. This enables it to perform direct actions on behalf of the user instead of only providing suggestions.

Smart Scheduling and Reminders

HabitCoach can create, modify, and manage schedules through natural language commands. Users can request recurring events, time blocks, or habit-linked activities without manual form interaction. For example, a user can ask HabitCoach to schedule a daily class or recurring routine across a specific date range. In addition to scheduling, HabitCoach automatically configures reminders based on context. Reminder behavior is adaptive and priority-aware:

- For structured events such as classes or meetings, reminders are scheduled earlier (e.g., one hour before the session).

- For habits and short routines, reminders may be delivered closer to execution time (e.g., five minutes before).
- Reminder priority is adjusted based on event type, user preferences, and habit importance.

Personalization and Learning

HabitCoach continuously learns from user behavior, preferences, and interaction patterns. As users log habits, complete tasks, miss routines, or reschedule activities, the assistant refines its understanding of the user's lifestyle and energy patterns. Over time, this learning process allows HabitCoach to:

- Suggest optimal scheduling windows based on past consistency.
- Recommend habit adjustments when performance drops.
- Personalize reminder timing and tone.
- Provide increasingly relevant guidance and next-step suggestions.

Conversational Control

HabitCoach supports free-form natural language interaction. Users can communicate with the assistant as they would with a human coach, asking questions, requesting changes, or seeking advice. The assistant responds in a supportive, encouraging tone while maintaining clear confirmation of actions performed. This conversational layer reduces friction and lowers the cognitive barrier for planning, making the system accessible even for users who prefer not to interact with complex interfaces.

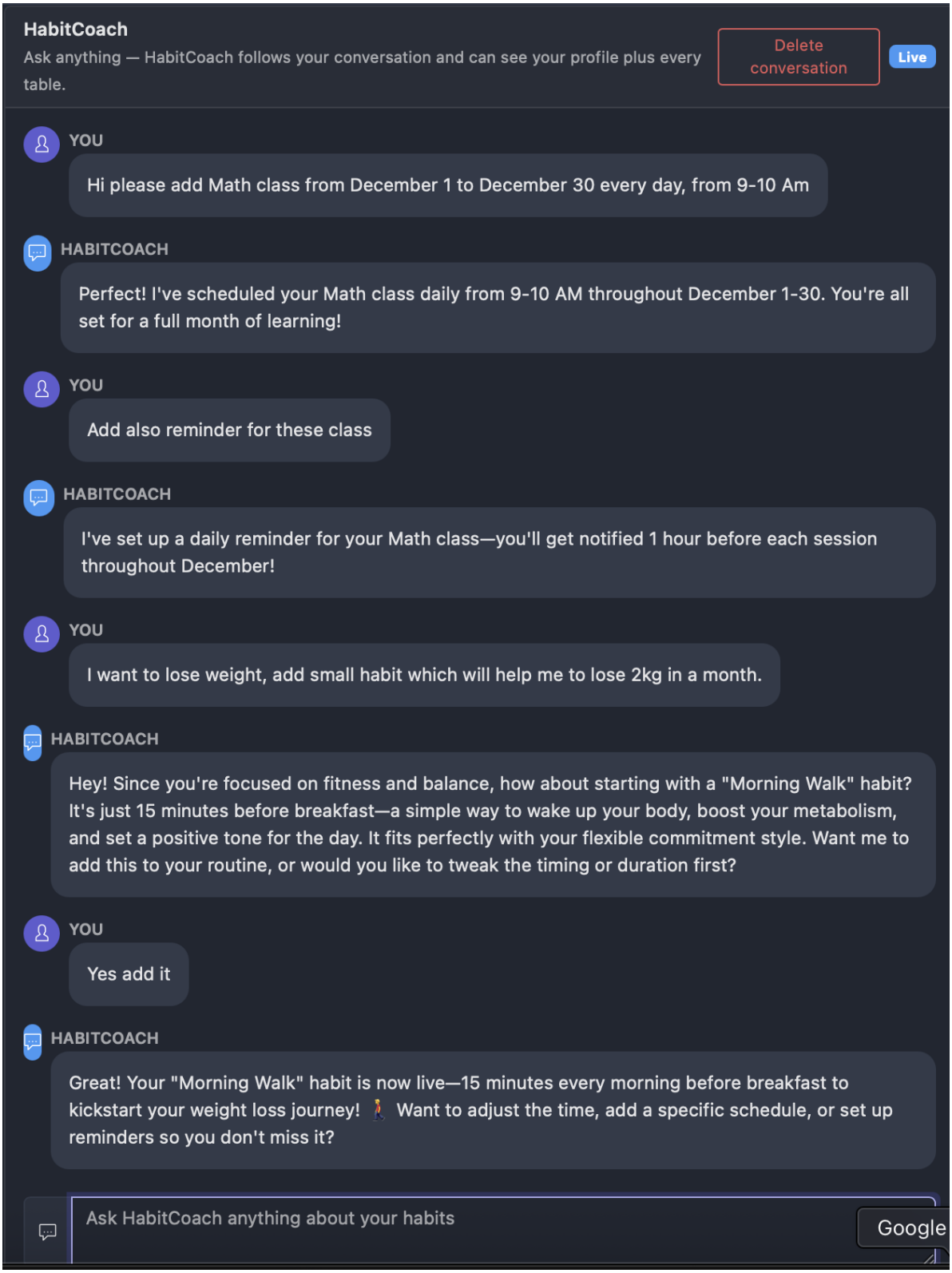

Figure 21: HabitCoach scheduling a recurring class and configuring a smart reminder through conversation

5.7 Communiy Module

The Community module of StepHabit provides a controlled social layer that enhances accountability, motivation, and long-term habit adherence. Rather than functioning as a traditional social network, this module is designed around purposeful interaction, progress sharing, and collaborative challenges, while maintaining strong privacy controls.

Friends Management

Users can discover, add, and manage friends through the Community interface. Friend connections are established via searchable identifiers such as name or email and require explicit acceptance, ensuring consent-based social interaction. Once connected, users can selectively share habit activity with friends. Sharing preferences are configurable per friend, allowing users to control whether their habits and progress are visible. This enables accountability partnerships without compromising personal boundaries.

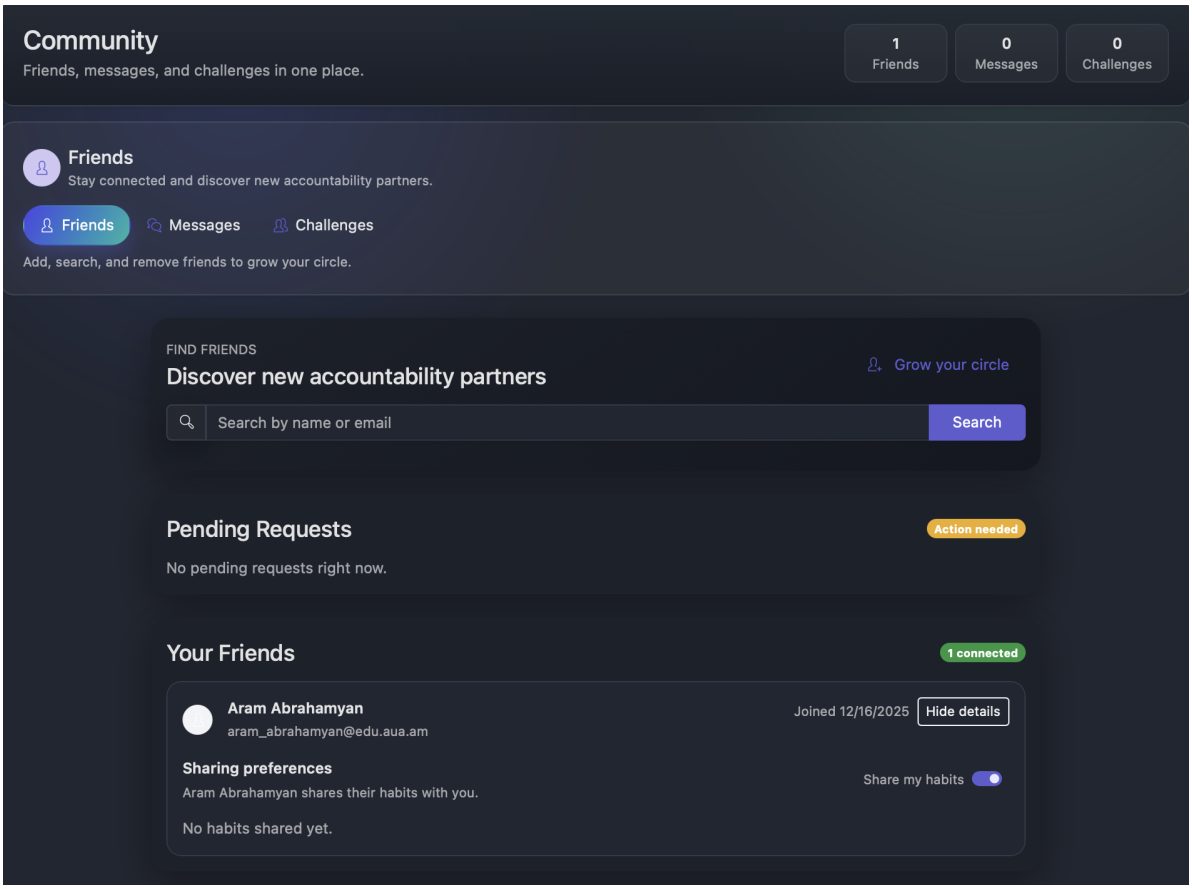

Figure 22: Community friends view with pending requests, search, and sharing controls

Private Messaging

The messaging system allows users to communicate directly with their connected friends. Conversations are organized into clean, threaded dialogs that support text messages and optional attachments such as images or location data. This feature is intended to support encouragement,

progress updates, and habit-related discussions rather than high-volume social interaction. Messages are private and accessible only to authenticated participants in the conversation.

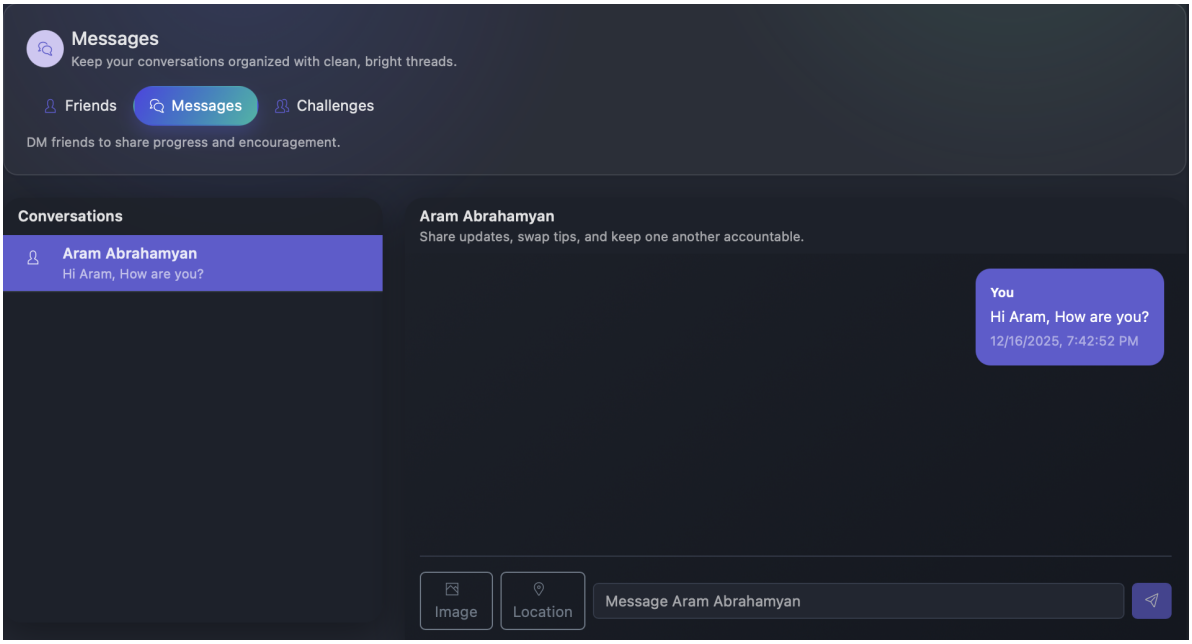

Figure 23: Direct messaging interface between connected users

Group Challenges

The Challenges feature allows users to create or join group-based goals designed to promote collective accountability. A challenge includes a title, date range, optional approval requirements, and a host who manages participation. Participants can join challenges, track progress together, and communicate through a dedicated challenge chat. This structure encourages social motivation while keeping the focus on shared objectives rather than competition.

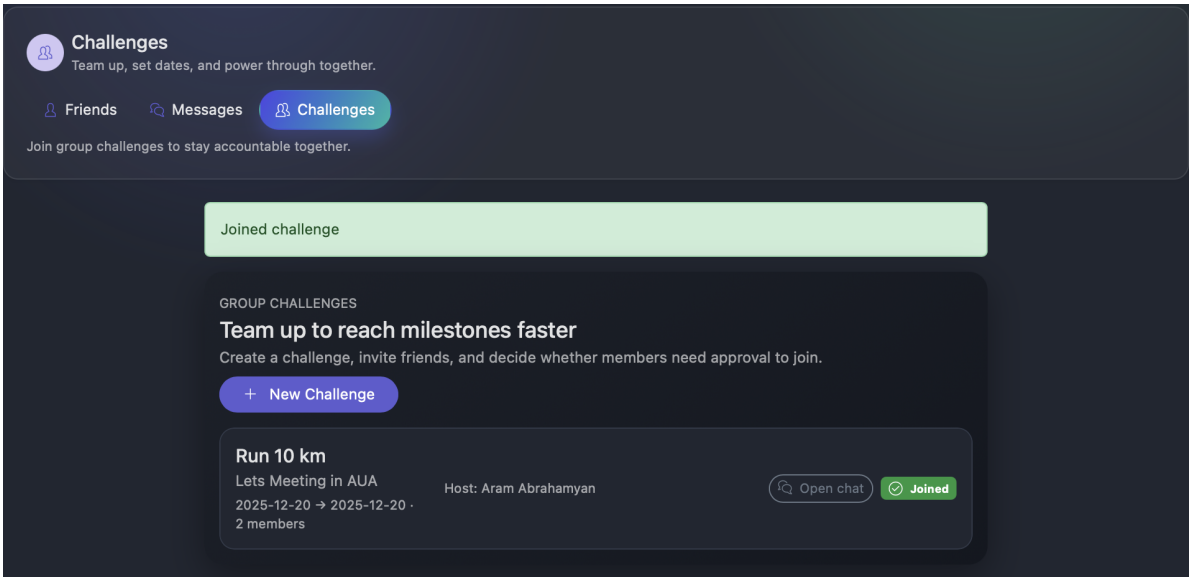

Figure 24: Group challenges overview with joinable challenges and challenge creation

Challenge Chat

Each group challenge includes an embedded chat interface where members can share updates, encouragement, or progress milestones. The challenge chat is isolated to the group and distinct from direct messages, ensuring contextual and goal-oriented communication. This communication channel strengthens group cohesion and reinforces accountability through shared progress visibility.

5.8 Profile Module

The Profile module serves as the centralized control center for user identity, preferences, personalization, and system integrations. It allows users to manage their account information, configure how the platform behaves, and fine-tune how AI-driven features such as HabitCoach interact with them. This module is designed to balance personalization, transparency, and user control, ensuring that recommendations and reminders remain aligned with individual goals and preferences.

Account Information

Users can manage their core account details, including display name, email address, and optional demographic information. Profile completion progress is visualized to encourage users to provide sufficient data for better personalization while keeping all non-essential fields optional. Users can upload a profile avatar and securely reset their password from this section. Sensitive operations such as password updates are protected by authentication and server-side validation.

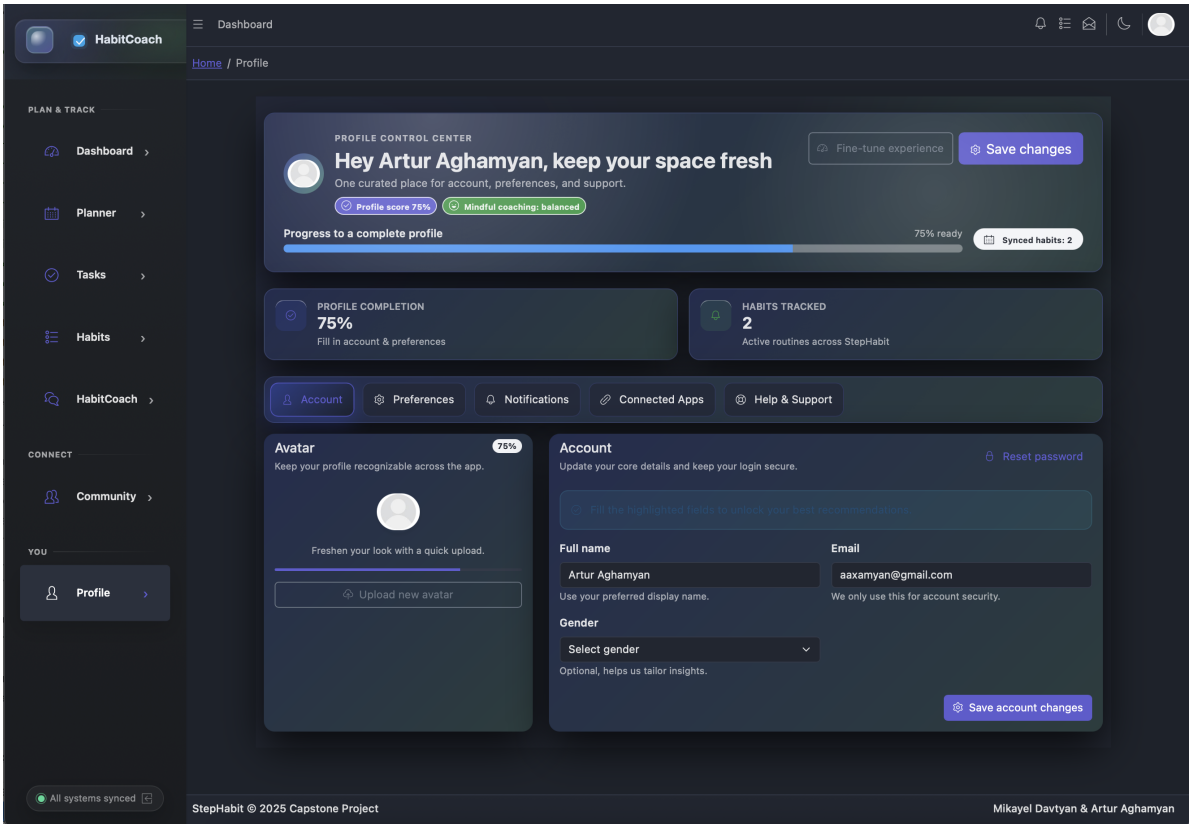

Figure 25: Profile account management with completion progress and avatar upload

User Preferences

The Preferences tab allows users to shape how StepHabit feels and responds. Users can select visual themes (e.g., light or dark mode), configure the tone of AI interactions, and choose a preferred support style such as encouragement or celebration of wins. These settings directly influence how HabitCoach communicates, ensuring that feedback and suggestions match the user's motivational style.

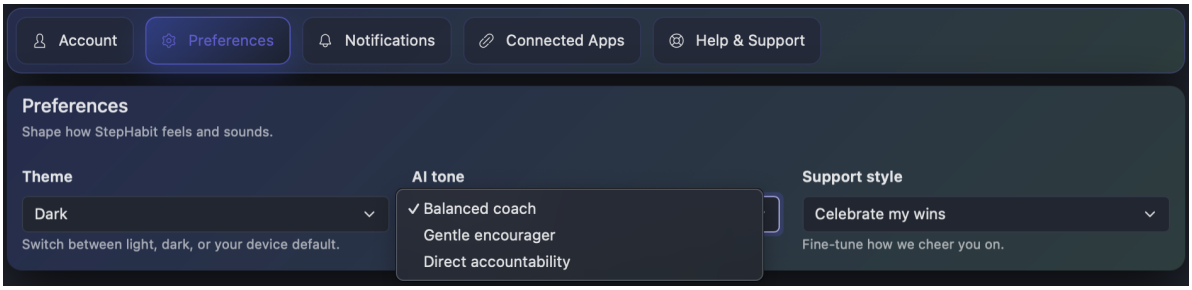

Figure 26: User preference controls for theme, AI tone, and support style

Notification Settings

Users can configure how and when they receive notifications. The system supports email and push-based reminders, with smart spacing to avoid notification fatigue. Notifications are context-aware and adapt based on habit type, importance, and user behavior. For example, critical events may trigger earlier reminders, while routine habits generate lighter prompts closer to execution time.

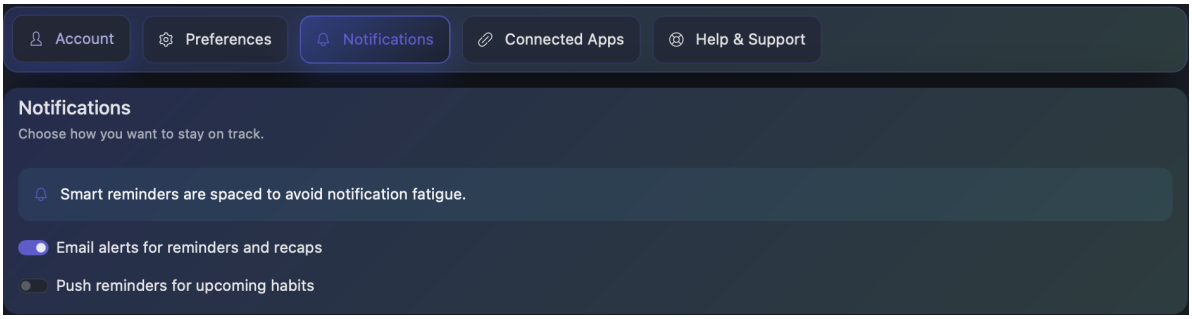

Figure 27: Notification preferences with smart reminder controls

Connected Applications

The Connected Apps section enables integration with external services such as calendar providers and fitness platforms. Calendar synchronization allows StepHabit to plan habits around real-world commitments, avoiding scheduling conflicts. Some integrations are active, while others are marked as future extensions. This design allows the system to remain extensible without disrupting the current user experience.

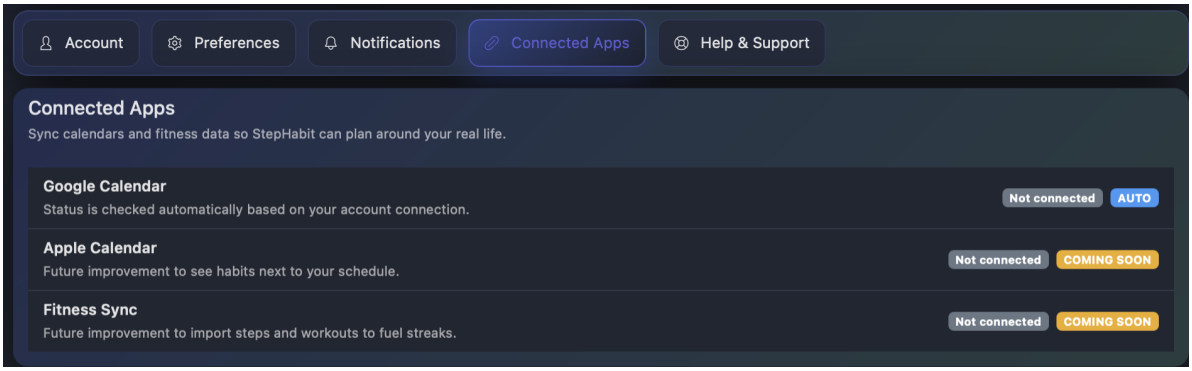

Figure 28: Connected applications for calendar and fitness synchronization

Help and Support

The Help & Support section provides users with access to documentation, FAQs, and direct communication channels with the development team. This ensures that users can resolve issues, learn system features, and provide feedback when needed. Support content is integrated directly into the platform to minimize disruption and maintain user trust.

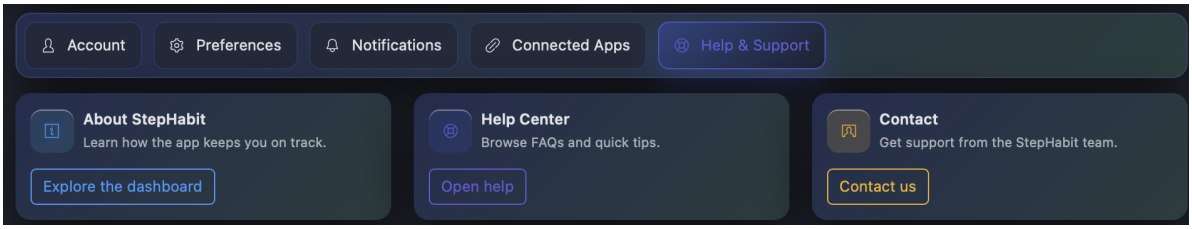

Figure 29: Help center and contact options within the profile module

Role of Profile Data in AI Personalization

Profile settings directly influence the behavior of AI-driven components such as HabitCoach and smart reminders. By combining explicit preferences with observed behavior, the system continuously refines scheduling suggestions, reminder timing, and communication style. This feedback loop ensures that StepHabit adapts to the user over time while remaining transparent and user-controlled.

5.9 Notification Center and Smart Reminder Module

The Notification Center acts as the central hub for all system-generated and user-created alerts within StepHabit. It consolidates reminders, progress nudges, schedule alerts, and AI-driven notifications into a single, organized interface, allowing users to stay informed without being overwhelmed. Notifications are categorized, prioritized, and timestamped, giving users full visibility into what requires attention, what is upcoming, and what has already been acknowledged.

Notification Overview

At the top level, the Notification Center provides a real-time summary of notification activity, including total notifications, unread items, upcoming scheduled reminders, and the most recently opened alert. Users can filter notifications by status or category to quickly locate relevant information. This design ensures clarity and reduces cognitive load, especially for users managing multiple habits, schedules, and reminders.

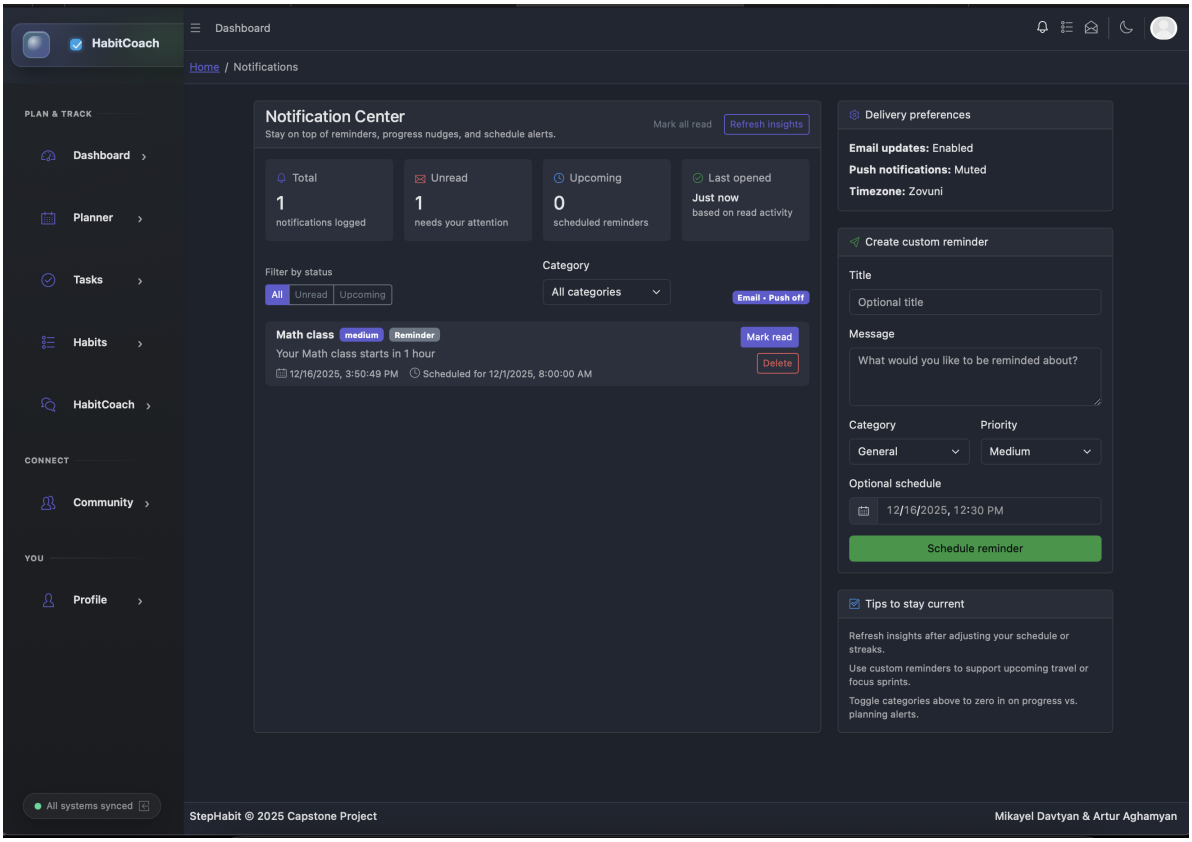

Figure 30: Notification Center overview with status indicators and filtering options

6 Challenges and Lessons Learned

Developing StepHabit as a full-stack, AI-assisted habit management platform introduced several technical and architectural challenges. Addressing these challenges required careful system design, iteration, and trade-off analysis, ultimately contributing to a deeper understanding of real-world software engineering practices.

6.1 Real-Time Communication and Event Handling

One of the main challenges was implementing real-time features that support instant feedback and synchronization across the application. StepHabit uses real-time communication mechanisms to support features such as notifications, HabitCoach responses, and dynamic UI updates. Managing user sessions in real time required handling edge cases such as unexpected disconnections, reconnections, and multiple active sessions. Ensuring consistent event delivery and avoiding duplicate or missed updates was particularly challenging when users navigated across different sections of the application. **Lesson learned:** This challenge strengthened my understanding of real-time client–server synchronization, event lifecycle management, and the importance of designing efficient and well-structured payloads for scalable communication.

6.2 Frontend and Backend Integration

Integrating the frontend and backend layers required maintaining strict consistency across data models, validation rules, and API contracts. Several issues emerged when frontend expectations did not fully align with backend responses, particularly during authentication, form validation, habit creation flows, and error handling. Ensuring predictable behavior required refining API response formats and improving validation feedback so that errors could be clearly communicated to the user interface. **Lesson learned:** Clearly documented APIs and a shared data contract between the frontend and backend are essential for reliable integration. Tools such as Swagger were critical in aligning expectations and accelerating development.

6.3 AI Integration and User Experience

Integrating the HabitCoach AI presented both technical and design challenges. The AI system needed access to user context, habits, schedules, and preferences while maintaining performance, safety, and responsiveness. Handling AI latency, fallback behavior, and usage limits required careful coordination between the backend logic and frontend experience. Additionally, AI-generated suggestions had to remain supportive, relevant, and trustworthy, which required balancing automation with user control and transparency. **Lesson learned:** AI integration extends beyond model outputs. It requires careful UX design, safety considerations, rate limiting, and graceful degradation to maintain user trust and system reliability.

6.4 Scheduling Logic and Smart Notifications

Designing the scheduling and reminder system was another significant challenge. StepHabit supports both habit-based and event-based reminders, each requiring different timing, priority levels, and delivery logic. For example, scheduled classes may trigger reminders earlier, while habits require minimal but timely nudges. Preventing notification fatigue while still providing effective

guidance required implementing priority-based logic and respecting user-defined notification preferences. **Lesson learned:** Effective reminder systems must balance automation with personalization. Smart defaults combined with user control significantly improve engagement and usability.

6.5 Overall Reflection

Developing StepHabit highlighted the importance of system thinking, clear contracts between components, and iterative refinement. Each challenge contributed to a deeper understanding of scalable architecture, AI-assisted systems, and user-centered software design. These experiences collectively reinforced the value of building robust, maintainable systems that balance technical complexity with real-world usability.

7 Future Improvements

While StepHabit already provides a comprehensive habit, scheduling, and AI-assisted coaching experience, several enhancements can further improve scalability, usability, and long-term impact.

7.1 Mobile Application Development

A natural next step for StepHabit is the development of native mobile applications for iOS and Android platforms. Mobile access would enable real-time habit logging, instant push notifications, and tighter integration with device features such as calendars, health data, and location-based reminders.

Planned improvement: Develop a cross-platform mobile application (e.g., using Flutter or React Native) that shares the same backend and authentication system to ensure consistency across web and mobile experiences.

7.2 Production Deployment and Cloud Infrastructure

Currently, StepHabit is designed and tested in a development environment. Deploying the application to a production-grade cloud infrastructure would significantly improve reliability, availability, and performance for real users.

Planned improvement: Host the backend on a cloud platform with scalable resources, configure environment-based deployments, and introduce monitoring and logging to track system health and usage patterns.

7.3 Advanced Personalization and Recommendation Engine

Although StepHabit already provides AI-driven guidance through HabitCoach, future iterations could introduce a dedicated recommendation engine that learns from user behavior over time. This system could suggest habits, optimal scheduling windows, and adjust reminder timing based on historical success rates and engagement patterns.

Planned improvement: Implement a behavior-aware recommendation system that analyzes habit completion history, streak consistency, and time-of-day performance to deliver increasingly personalized suggestions.

7.4 Enhanced Notification Intelligence

While the current notification system supports smart reminders, future versions could introduce adaptive notification strategies that respond dynamically to user behavior. For example, the system could reduce reminder frequency when habits are consistently completed or increase support during periods of decline.

Planned improvement: Introduce adaptive reminder logic that balances motivation with notification fatigue by learning optimal reminder timing and priority per user.

7.5 Expanded AI Capabilities

HabitCoach currently assists users with scheduling, reminders, and habit-related guidance. Future enhancements could enable deeper conversational memory, long-term behavior analysis, and proactive coaching strategies.

Planned improvement: Extend HabitCoach with long-term context awareness, goal-based planning, and voice interaction support to create a more human-like and supportive coaching experience.

7.6 Scalability and Performance Optimization

As the user base grows, optimizing performance becomes increasingly important. Certain frequently accessed data—such as habit summaries, schedules, and insight metrics—can be optimized to reduce response time and backend load.

Planned improvement: Introduce caching strategies and query optimization to ensure the system remains responsive under increased traffic.

7.7 Overall Vision

These future improvements aim to evolve StepHabit from a feature-rich academic project into a scalable, production-ready platform. By expanding to mobile, strengthening AI personalization, and improving system performance, StepHabit has the potential to become a long-term personal growth companion for users across different platforms.

8 Conclusion

StepHabit represents a comprehensive and thoughtfully engineered solution for habit formation, productivity planning, and long-term personal growth. By combining habit tracking, task management, intelligent scheduling, and AI-powered coaching, the platform supports users in building sustainable routines while maintaining clarity, balance, and motivation in their daily lives. Features such as smart reminders, progress visualization, streak tracking, and adaptive insights encourage consistency, self-awareness, and accountability over time.

The system is built on a modular and scalable architecture that clearly separates concerns across its core components. The frontend interface provides a clean, calm, and intuitive user experience that enables users to plan schedules, manage tasks, track habits, and interact with the AI HabitCoach seamlessly. The backend layer handles authentication, business logic, data processing, and real-time operations securely and efficiently, ensuring reliable communication between system components.

A central aspect of StepHabit is the integration of AI-driven assistance through *HabitCoach*. The AI assistant is capable of accessing user context, schedules, habits, tasks, and preferences to suggest routines, create schedules, set smart reminders, and provide personalized guidance. Unlike static reminder systems, HabitCoach adapts notification timing and priority based on context—for example, reminding users earlier for high-priority events such as classes, while using shorter reminders for habits. Over time, the assistant learns user behavior and preferences, enabling a more supportive and human-like coaching experience.

The platform also includes community-oriented features that enhance motivation through social accountability. Users can connect with friends, exchange messages, participate in group challenges, and share progress selectively. These features encourage collaboration, peer support, and healthy competition, reinforcing habit consistency beyond individual use.

StepHabit leverages a modern technology stack designed for scalability and future growth. Secure authentication, real-time updates, structured APIs, and well-defined data models ensure system reliability and maintainability. The development workflow is supported by modern tooling and best practices, enabling efficient collaboration, debugging, and iteration throughout the project lifecycle.

Looking ahead, StepHabit is well-positioned for further expansion. Planned improvements include deploying the system to a production-grade server environment, developing native mobile applications, enhancing AI personalization, and deepening integration with external services such as calendars and fitness platforms. These enhancements aim to transform StepHabit from a feature-rich academic project into a scalable, real-world productivity and habit-building platform.

In conclusion, StepHabit demonstrates how modern web technologies, intelligent system design, and AI-driven personalization can be combined to support meaningful behavior change. By focusing on clarity, adaptability, and user-centered design, the platform illustrates the potential of digital tools to help users build better habits, stay consistent, and grow intentionally over time.