



HabitFlow - Project Report

Habit Tracking Application

Course: CCCS106 Application Development / Information Assurance / Software Engineering

Institution: Camarines Sur Polytechnic Colleges

Date: December 10, 2025

Prepared by: CodeMates

Version: 1.0.0

Table of Contents

1. Executive Summary
2. Framework Chosen & Rationale
3. Implemented Features
4. Architecture & Module Overview
5. Threat Model & Security Controls
6. Design Decisions & Trade-offs
7. Limitations & Future Work

1. Executive Summary

Project Overview

HabitFlow is a secure, offline-first mobile habit tracking application developed using Python and the Flet framework for mobile phone platforms. The application enables users to build and maintain positive daily routines while providing enterprise-grade security controls, intelligent AI-powered categorization, and comprehensive analytics.

This report outlines the design and architecture of HabitFlow, the security model, and the implementation strategy based on the information assurance best practices, such as threat modeling, access control, cryptographic password storage, and a detailed logging strategy. The system focuses on user privacy by storing data locally, defense-in-depth security by design, and secure coding by extensive unit testing (62 tests, 88% coverage).

Problem Statement

Animal shelters traditionally rely on fragmented tools spreadsheets, phone calls, paper forms, and social media leading to:

- Lack of structured tracking system – Without a systematic way to monitor progress, users lose



motivation.

- Privacy concerns with cloud-based apps – Existing habit trackers often require uploading personal data to third-party servers.
- Complex, feature-heavy interfaces – Many apps overwhelm users with unnecessary complexity instead of simplifying habit formation.
- No intelligent categorization – Users manually organize habits, adding friction to the process.

Solution

HabitFlow addresses these challenges by providing:

- Simple, intuitive mobile interface – Minimal steps to add and track habits daily.
- Local-first architecture – All data remains on the user's device; no cloud transmission required.
- AI-powered smart categorization – Automatic category suggestions as users type habit names.
- Comprehensive analytics – Streaks, completion rates, and category breakdowns to visualize progress.
- Enterprise-grade security – bcrypt password hashing, account lockout protection, security logging.

Target User

- Students – Track study habits, exercise routines, sleep schedules.
- Professionals – Monitor productivity habits, exercise consistency, work-life balance.
- Health-conscious individuals – Build and maintain wellness routines.

Framework Chosen & Rationale

Component	Choice	Rationale
UI Framework	Flet (Python + Flutter)	Cross-platform (Android/iOS) support from single Python codebase; rapid development; native Material Design 3.
Database	SQLite (local)	Embedded, zero-configuration; ACID compliance; thread-safe connections; no server required.
Password Security	bcrypt	Industry-standard; automatic salt generation; configurable work factor (10 rounds default).
Testing Framework	pytest + pytest-cov	Comprehensive assertion library; built-in fixtures; detailed coverage reporting.
UI/UX Design	Material Design 3	Consistent cross-platform look-and-feel; accessibility guidelines; modern design patterns.
Language	Python 3.8+	Familiar to team; extensive security libraries; rapid prototyping; readable code for maintenance.



Alternative Considered

Alternative	Why Not Chosen
React Native / Flutter	Requires JavaScript/Dart expertise; longer learning curve for team; less suitable for quick prototyping.
Web-based (Flask/Django)	Not mobile-native; browser dependency; poor offline capability; not aligned with mobile-first requirement.
Firebase / Cloud backend	Compromises privacy (cloud storage); adds infrastructure complexity; contradicts offline-first design.
PostgreSQL	Overkill for local app; requires server setup; adds deployment complexity.
Argon2 (password hashing)	bcrypt is more universally supported; equally secure for our use case; simpler library integration.

Implemented Feature & Scope

Feature	Status	Details
User Authentication	✓ Implemented	Email/password registration, secure login, password validation (min 6 chars).
Habit CRUD	✓ Implemented	Create, read, update, delete habits with metadata (name, category, frequency).
Daily Tracking	✓ Implemented	Mark habits complete via checkbox toggle on Today screen.
Basic Analytics	✓ Implemented	Current streak, completion rate, per-habit statistics.
Data Storage	✓ Implemented	Local SQLite database with user data isolation via foreign keys.
User Preferences	✓ Implemented	Theme selection (light/dark), password management, account deletion.

Enhanced Features

Feature	Status	Details
AI Categorization	✓ Implemented	Rule-based on-device AI suggests categories using keyword matching, regex patterns, fuzzy matching.
Account Lockout	✓ Implemented	5 failed attempts → 15-minute lockout to prevent brute-force attacks.
Security Logging	✓ Implemented	All login attempts recorded with timestamp and success/failure status in login_history table.

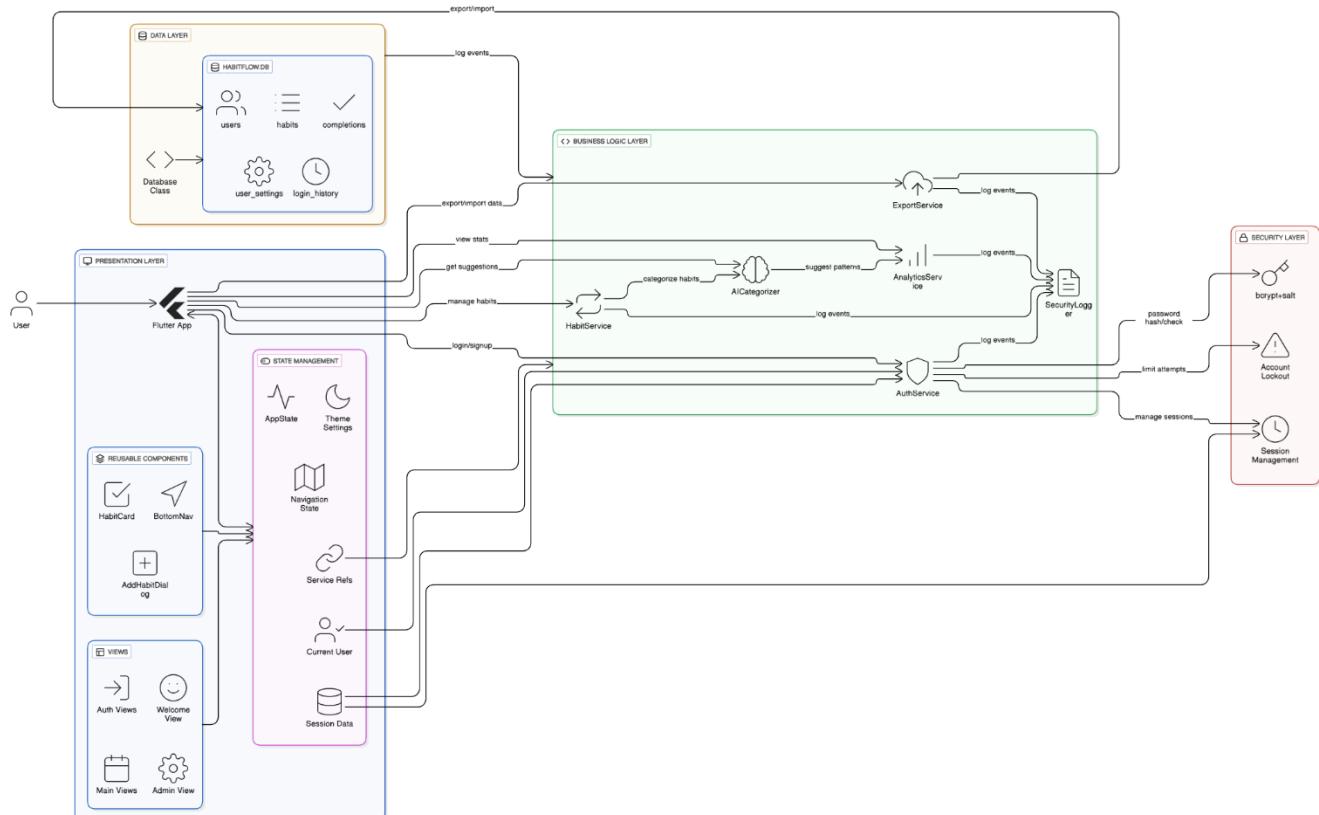


Feature	Status	Details
Advanced Analytics	<input checked="" type="checkbox"/> Implemented	Weekly stats, category breakdown, best day analysis, longest streak tracking.
Data Export	<input checked="" type="checkbox"/> Implemented	JSON backup export from Settings screen for user-controlled backup.
Admin Dashboard	<input checked="" type="checkbox"/> Implemented	Admin users can view user list, disable accounts, access security logs.

Out of Scope (Future Work)

- Cloud sync – Planned for v2.0; requires backend infrastructure.
- Push notifications – Planned for v2.0; requires notification service.
- Wearable integration – Planned for v4.0; requires wearable SDKs.
- Multi-language UI – Currently English-only; planned for v3.0.
- Web client – Planned for future; currently mobile-only.

System Architecture & Module Overview





Module Description

Module	Path	Purpose	Key Classes		
Views	app/views/	Mobile screens	UI	WelcomeView, HabitsView, AdminView	AuthView, StatsView, TodayView, SettingsView,
Components	app/components/	Reusable elements	UI	HabitCard, AddHabitDialog, ThemeToggle	BottomNav,
Services	app/services/	Business logic		AuthService, HabitService, AnalyticsService, ExportService, SecurityLogger, AICategorizer	
Models	app/models/	Data structures		User, Habit, Completion, LoginAttempt	
Storage	app/storage/	Database abstraction		Database (SQLite manager, thread-safe)	
State	app/state/	Global app state		AppState (singleton with current user, theme, cached data)	
Tests	app/tests/	Unit/integration tests		test_auth_service.py, test_habit_service.py, test_analytics_service.py	test_database.py,

Data Flow Example: Adding a Habit

1. User enters habit name in UI → AddHabitDialog component
2. AI categorization triggered → AICategorizer service analyzes name, suggests category
3. User confirms or overrides category → UI sends to HabitService
4. HabitService validates input → Checks for duplicates, enforces business rules
5. Data persisted to SQLite → Database.insert_habit() via parameterized query
6. AppState updated → Cached habits refreshed
7. UI re-renders → HabitsView and TodayView refresh to show new habit

Threat Model & Security Controls

Threat Modeling (STRIDE)

Threat Category	Specific Threat	Potential Impact	Mitigation
Spoofing	Unauthorized login / credential guessing	Account takeover, data access	bcrypt + 5-attempt lockout + 15-min delay
Tampering	SQL injection via user input	Data modification, data breach	Parameterized SQLite queries



Threat Category	Specific Threat	Potential Impact	Mitigation
Tampering	Database modification / file	Data corruption	File permissions, SQLite integrity checks, CASCADE DELETE constraints
Repudiation	User denies actions (habit creation, deletion)	Accountability loss	Habit creation/deletion timestamps, audit logs possible in v2.0
Information Disclosure	Password theft / plaintext storage	Account compromise	bcrypt hashing with salt; passwords never stored plaintext
Information Disclosure	Session hijacking	Unauthorized access	Session token not used; app-level user tracking only; no cross-device sessions in v1
Denial of Service	Brute-force login	Availability impact	Account lockout after 5 failed attempts
Elevation of Privilege	Regular user accesses admin features	Unauthorized access to user management	Role-based access control; admin check before dashboard access

Security Controls Implemented

- **Authentication & Authorization**
 - Password Hashing
 - Algorithm: bcrypt (Blowfish cipher)
 - Salt: Automatic per-password (10 rounds)
 - Validation: Minimum 6 characters enforced at registration
 - Never plaintext: Hashed passwords only stored in database
- **Account Lockout**
 - Failed attempts threshold: 5
 - Lockout duration: 15 minutes
 - Counter reset: On successful login
 - Implementation: Checked in AuthService.login() before password verification
- **Login Logging**
 - Table: login_history
 - Fields: user_id, timestamp, ip_address (optional), success/failure
 - Purpose: Audit trail; manual inspection for suspicious patterns
- **Role-Based Access Control (RBAC)**



- Admin users: Can access admin dashboard, view security logs, manage users
- Regular users: Can only access their own habits and settings
- Enforcement: Role check before sensitive operations in service layer

Data Protection

- **Local Storage Only**
 - Data stored on device in SQLite; no network transmission
 - Privacy by design: No cloud sync in v1
- **Parameterized Queries**
 - All SQL uses placeholders (?) to prevent injection
 - Example: `SELECT * FROM users WHERE email = ?` with bound parameters
- **Foreign Key Constraints**
 - Enforced at database schema level
 - Prevents orphaned records (e.g., habits without users)
 - CASCADE DELETE: Deleting user removes all related habits, completions, logs
- **Input Validation**
 - Email format validation
 - Password length/complexity checks
 - Habit name length limits (prevent overflow)
 - Category enum validation (only predefined categories allowed)
- **Cryptography**
 - Hashing Algorithm: bcrypt
 - Standard: Based on Blowfish cipher
 - Work factor: 10 rounds (configurable)
 - Output: 60-character hash with embedded salt
 - Collision resistance: Not applicable (one-way function)
- **No encryption of data at rest (by design)**
 - Justification: All data is non-sensitive habit names; privacy comes from local-only storage
 - Future consideration: End-to-end encryption if cloud sync added

Audit & Logging

- **Security Event Logging**
 - Logged events: Login attempts (success/fail), failed lockout, admin actions (future v2.0)
 - Storage: Separate `login_history` table
 - Retention: No automatic purging (manual admin review)



Design Decisions & Trade-offs

Key Design Decisions

Decision	Rationale	Trade-off
Local SQLite over cloud database	Privacy-first; offline-capable; simple deployment	No automatic cross-device sync; v1 limited to single device
Rule-based AI over ML model	Deterministic; no training data needed; runs offline	Less adaptive; English-only; no learning from user feedback
Single-user focus (v1)	Simplicity; lower security complexity; faster development	No team/social features; future expansion needed for collaboration
Material Design 3	Modern UX; accessibility; cross-platform consistency	Limited customization; follows Google standards (may not suit all users)
bcrypt over Argon2	Proven track record; library support; sufficient security level	Slightly slower than Argon2 (by design); less modern
No server-side authentication	Privacy; offline-first; no backend infrastructure	No password recovery feature; users must remember credentials

Alternatives Considered & Rejected

Alternative	Why Rejected	Impact
PostgreSQL with cloud backend	Breaks privacy promise; infrastructure costs; scope creep	Cloud sync would be v2.0+ only
Token-based session management	Unnecessary complexity for local app; tokens would need storage	App-level state management simpler for v1
End-to-end encryption	Adds complexity; premature for local-only v1	Reconsidered if cloud sync added in v2.0
ML categorization	No training data; expensive to compute on mobile; overkill for v1	Rule-based approach sufficient; ML planned for v3.0
Web-first approach	Mobile is primary use case; browser-based less native	Desktop/web client deferred to future

Trade-off Analysis

Security vs. Usability

- Decision: Require strong passwords (min 6 chars) but not enforce uppercase/numbers/symbols
- Rationale: Balance security (prevent weak passwords) with usability (avoid excessive requirements that users circumvent)

Privacy vs. Features

- Decision: Local-only storage in v1 instead of cloud features



- Rationale: Privacy is more important for v1; cloud sync requested but deferred to v2.0 when infrastructure can be properly secured

Complexity vs. Functionality

- Decision: Rule-based AI instead of ML model
- Rationale: Keep v1 simple and lightweight; ML categorization planned for v3.0 once user base provides training data

Limitations & Future Work

Limitation	Impact	Severity
No cloud sync	Data stuck on one device; users lose data if phone lost	High
No push notifications	Users must manually open app; may forget to track	Medium
Rule-based AI only	Categorization can misclassify ambiguous habits; doesn't improve over time	Medium
English-only UI	Excludes non-English speakers; habit names must be typed in English	Low-Medium
Single-user design	Cannot share progress with friends; no team accountability	Medium
No password recovery	Users locked out if they forget password (no reset email)	Medium
Limited admin features	Basic user management only; no advanced analytics or compliance reporting	Low
No wearable integration	Cannot pull data from fitness trackers or smartwatches	Low

Identified Risks & Mitigations

Risk	Likelihood	Impact	Mitigation
Data loss (device damage/loss)	Medium	High	JSON export feature; user responsibility to backup
Password compromise	Low	High	bcrypt hashing + account lockout + login logging
Brute-force attack	Low	Medium	5-attempt lockout + 15-min delay prevents scalable attacks
SQL injection	Very Low	Critical	Parameterized queries + input validation
Concurrent access issues	Low	Medium	SQLite connection pooling + transaction



Risk	Likelihood	Impact	Mitigation
			management
Large data set performance	Low	Medium	Efficient queries; pagination planned for v2.0

Future Roadmap

Version 2.0

- Cloud backup with optional sync (Firebase or custom backend)
- Push notifications for daily reminders
- Widget support for home screen quick access
- PDF export for progress reports
- Multi-language UI (Filipino, Spanish)

Version 3.0

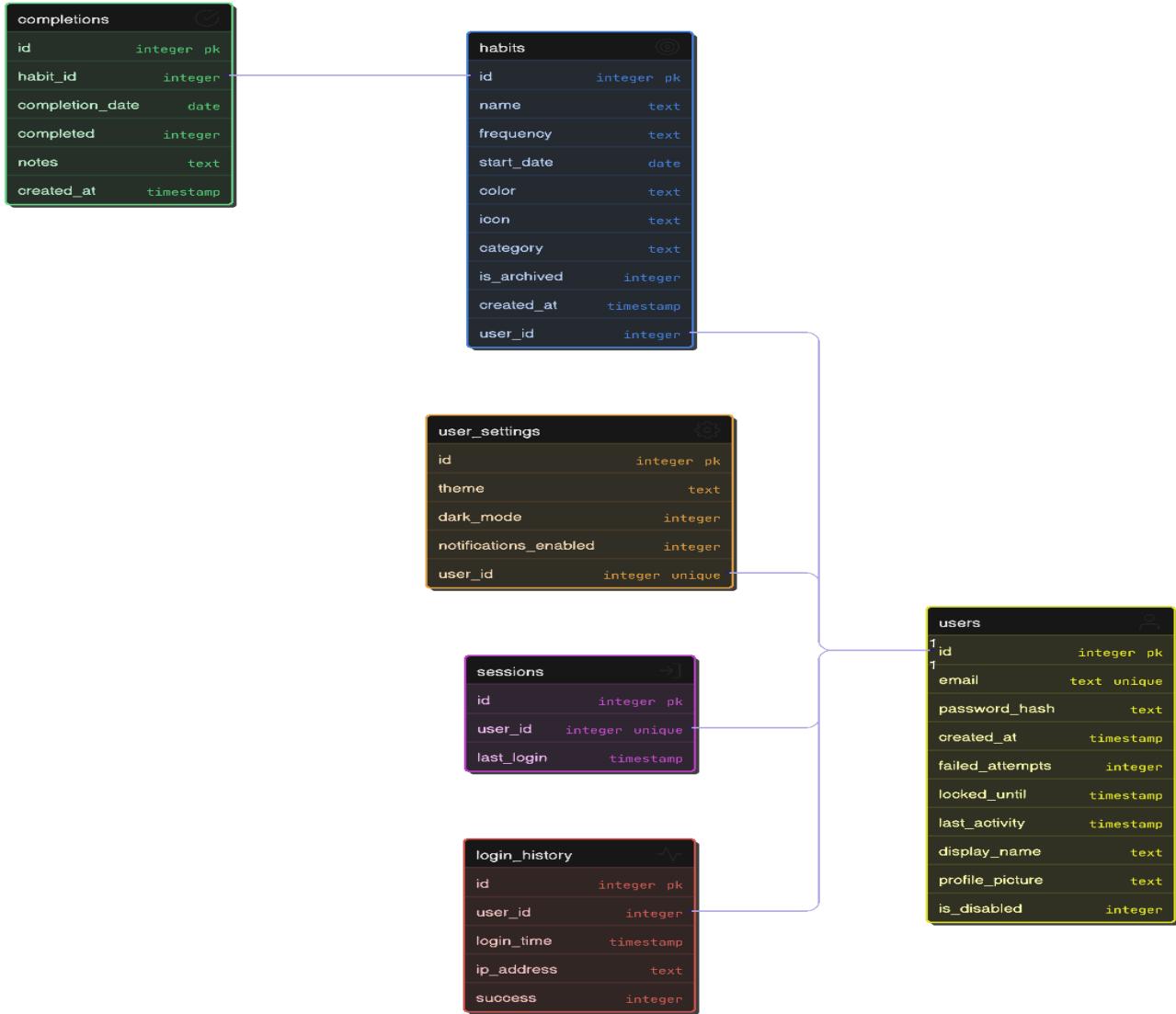
- Social features: share streaks, friend accountability groups
- Gamification: badges, levels, leaderboards
- ML-powered predictions: identify at-risk habits
- Calendar integration (Google, Apple)
- Advanced admin dashboard: analytics, compliance reports

Version 4.0

- AI coaching assistant with personalized recommendations
- Wearable integration (Apple Watch, Fitbit, Garmin)
- Enterprise/team features for workplace wellness programs
- End-to-end encryption for cloud-synced data
- Multi-platform deployment (web, desktop apps)



Technical Documentation



Setup & Installation

Prerequisites:

- Python 3.8 or higher
- pip package manager
- ~100 MB disk space

Installation Steps:

1. Clone repository:

```
git clone https://github.com/lililhuan/Habit-Flow.git
cd Habit-Flow
```



2. Create virtual environment (recommended):

```
python -m venv venv
source venv/bin/activate # macOS/Linux
venv\Scripts\activate # Windows
```

3. Install dependencies:

```
pip install -r requirements.txt
```

4. Run application:

```
flet run app/main.py
```

Configuration (Optional):

- Copy .env.example to .env
- Adjust settings: database path, admin emails, security parameters

How to Run Tests

Run all tests:

```
pytest
```

Verbose output:

```
pytest -v
```

Coverage report (terminal):

```
pytest --cov=app --cov-report=term-missing
```

Coverage report (HTML):

```
pytest --cov=app --cov-report=html
# Open htmlcov/index.html in browser
```

Run specific test file:

```
pytest app/tests/test_auth_service.py
```

Testing & Quality Assurance

Test Plan & Coverage

Module	Test File	Test Count	Coverage	Focus Areas
Authentication	test_auth_service.py	15	90%	Password hashing, login/logout, account lockout, registration validation
Database	test_database.py	12	90%	CRUD ops, foreign keys, cascading deletes, thread safety, connection pooling



Module	Test File	Test Count	Coverage	Focus Areas
Habit Management	test_habit_service.py	20	86%	Habit creation, editing, deletion, AI categorization, user isolation
Analytics	test_analytics_service.py	15	83%	Streak calculation, completion rates, category stats, trend analysis
Models	(Inline tests)	–	91%+	Data validation, edge cases

Total: 62 passing tests, ~88% overall coverage

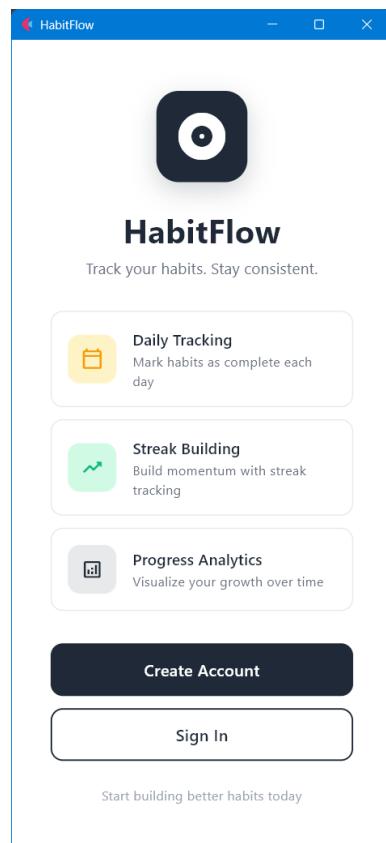
Test Results Summary

```
=====
test      session      starts
=====
platform win32 -- Python 3.11.0, pytest-7.4.0
collected 62 items

app/tests/test_auth_service.py ..... 15 PASSED
app/tests/test_database.py ..... 12 PASSED
app/tests/test_habit_service.py ..... 20 PASSED
app/tests/test_analytics_service.py ..... 15 PASSED

===== 62 passed in 2.34s =====
```

HabitFlow User Manual



Welcome Screen

From the welcome screen you can:

- Read the key features: Daily Tracking, Streak Building, Progress Analytics.
- Tap **Create Account** to register, or **Sign In** if you already have an account.



Creating an Account

The screenshot shows the 'Create Account' screen of the HabitFlow app. It features a header with the HabitFlow logo and a 'Create Account' button. Below the header is a sub-header 'Start your habit journey today'. The main form contains three input fields: 'EMAIL' with placeholder 'Enter your email', 'PASSWORD' with placeholder 'Create a password', and 'CONFIRM PASSWORD' with placeholder 'Confirm your password'. Each password field has a visibility toggle icon. Below the password fields is a box titled 'Password must have:' containing three radio buttons: 'At least 8 characters', 'Contains a number', and 'Contains uppercase letter'. At the bottom is a large dark blue 'Create Account' button with a user icon.

Steps:

1. Enter a valid email address.
2. Create and confirm your password.
3. Make sure the password meets the requirements shown on screen (length and complexity).
4. Tap **Create Account** to finish registration.

Signing In

The screenshot shows the 'Sign In' screen of the HabitFlow app. It features a header with the HabitFlow logo and a large right-pointing arrow icon. Below the header is a sub-header 'Welcome Back' with the text 'Sign in to continue your journey'. The main form contains two input fields: 'EMAIL' with placeholder 'Enter your email' and 'PASSWORD' with placeholder 'Enter your password'. Each field has a visibility toggle icon. Below the fields is a large dark blue 'Sign In' button with a right-pointing arrow icon. At the bottom left is the text 'Don't have an account? [Sign Up](#)'.

Steps:

1. Enter the email and password you used at registration.
 2. Tap **Sign In** to open your HabitFlow dashboard.
- If you enter the wrong password too many times, your account may be temporarily locked as a security measure.



Main Navigation

The screenshot shows the HabitFlow app's sign-in interface. At the top, there is a navigation bar with a back arrow, a search icon, and a close button. Below the navigation bar is a large central area with a dark blue header containing a white right-pointing arrow icon. The text "Welcome Back" is displayed in bold black font, followed by the subtext "Sign in to continue your journey". Below this, there is a form with two input fields: "EMAIL" with an envelope icon and "PASSWORD" with a lock icon. Both fields have placeholder text "Enter your email" and "Enter your password" respectively. To the right of the password field is a visibility toggle icon. At the bottom of the form is a dark blue "Sign In" button with a white right-pointing arrow icon. At the very bottom of the screen, there is a small link "Don't have an account? [Sign Up](#)".

The bottom navigation bar contains:

- **Habits** – manage all your habits.
- **Today** – view and complete today's habits.
- **Add (+)** – quickly add a new habit.
- **Stats** – see analytics and streaks.
- **Settings** – manage account, appearance, and data

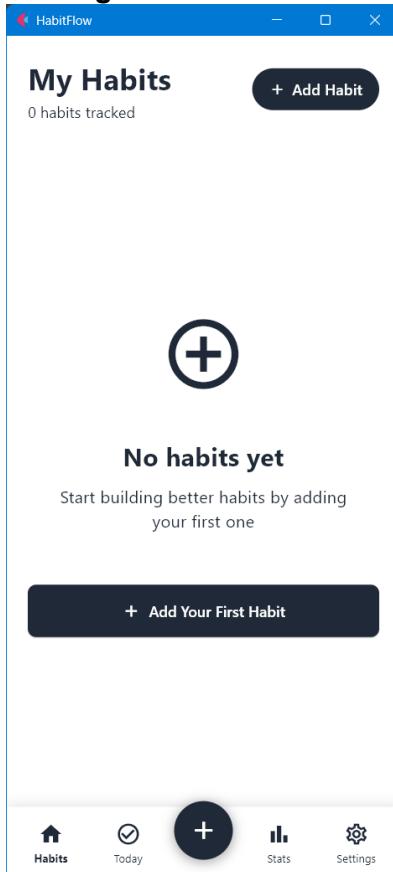
Core Features

The screenshot shows the HabitFlow app's sign-in interface, identical to the one in the previous section. It features a dark blue header with a white right-pointing arrow icon, the text "Welcome Back" and "Sign in to continue your journey", a form with "EMAIL" and "PASSWORD" fields, and a "Sign In" button at the bottom. A small "Don't have an account? [Sign Up](#)" link is also present at the very bottom.

- Tap **Add Habit** or **Add Your First Habit** to create a new habit.
- Use the list to review existing habits and open them for editing or deletion.



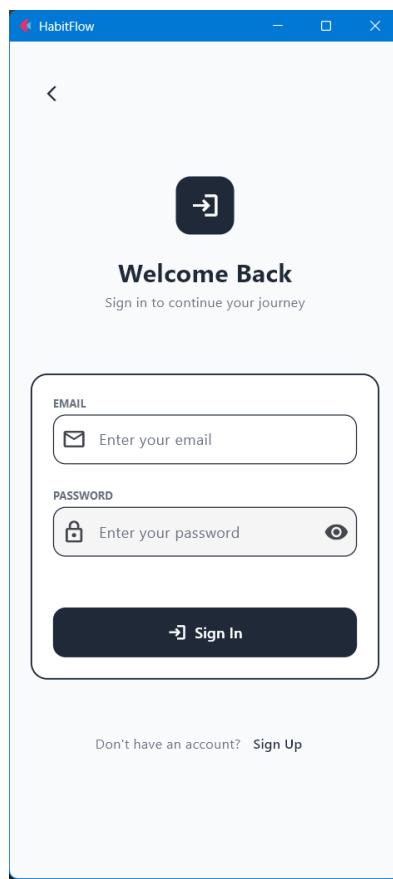
Adding a New Habit



Steps:

1. Enter the **Habit Name**.
2. Check the **Category (AI)** field; HabitFlow will suggest a category automatically based on the name, but you can change it manually.
3. Choose a **Frequency** (Daily, Weekly, or Custom).
4. Set the **Start Date**.
5. Tap **Create Habit**

Today View & Daily Tracking



- The Today tab shows all habits scheduled for the selected date.
- Tap the checkbox on a habit card to mark it as complete.
- The progress bar and percentage at the top update automatically as you complete habits.



Analytics

The screenshot shows the HabitFlow app interface. At the top, there's a navigation bar with a back arrow, a square icon with a right-pointing arrow, and a close button. Below the navigation is a header section with a left arrow, a square icon with a right-pointing arrow, and a right arrow. The main content area is titled "Welcome Back" with the sub-instruction "Sign in to continue your journey". There are two input fields: "EMAIL" with a placeholder "Enter your email" and "PASSWORD" with a placeholder "Enter your password" and a visibility toggle icon. A "Sign In" button with a right-pointing arrow icon is below the password field. At the bottom of the screen, there's a link "Don't have an account? Sign Up".

The **Stats** tab shows:

- Total habits and total completions.
- Average completion rate and best streak.
- Weekly progress chart and habit performance summaries.

Account Settings

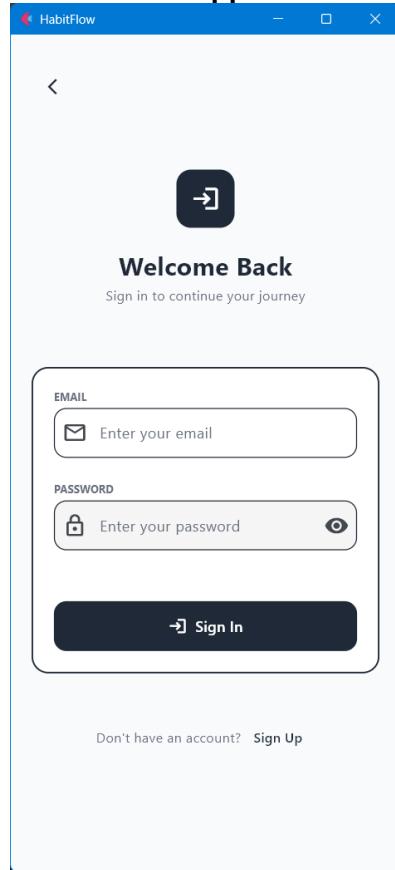
The screenshot shows the HabitFlow app interface, similar to the Analytics screen. It features a navigation bar at the top with a back arrow, a square icon with a right-pointing arrow, and a close button. Below the navigation is a header section with a left arrow, a square icon with a right-pointing arrow, and a right arrow. The main content area is titled "Welcome Back" with the sub-instruction "Sign in to continue your journey". There are two input fields: "EMAIL" with a placeholder "Enter your email" and "PASSWORD" with a placeholder "Enter your password" and a visibility toggle icon. A "Sign In" button with a right-pointing arrow icon is below the password field. At the bottom of the screen, there's a link "Don't have an account? Sign Up".

From **Settings → Account** you can:

- View your email address.
- Sign out of your account.
- Change your password to keep your account secure.



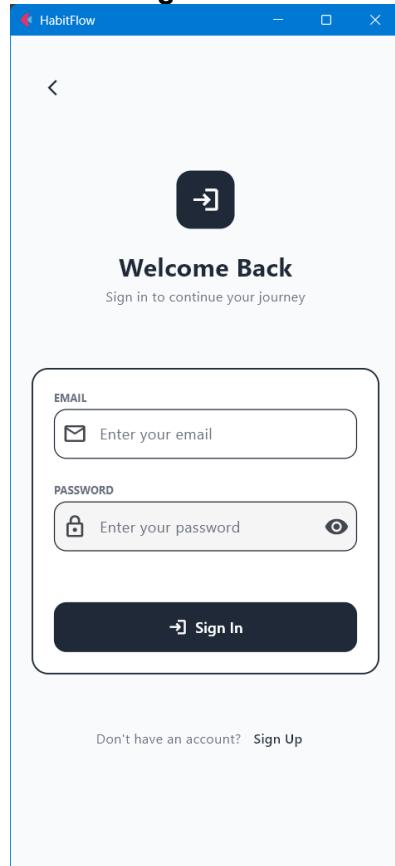
Themes and Appearance



From **Settings → Appearance** you can:

- Toggle **Dark / Light Mode**.
- Choose from multiple theme colors like Ocean Blue, Forest Green, Purple Dream, and more.
- See a live preview when a theme is applied successfully.

Data Management

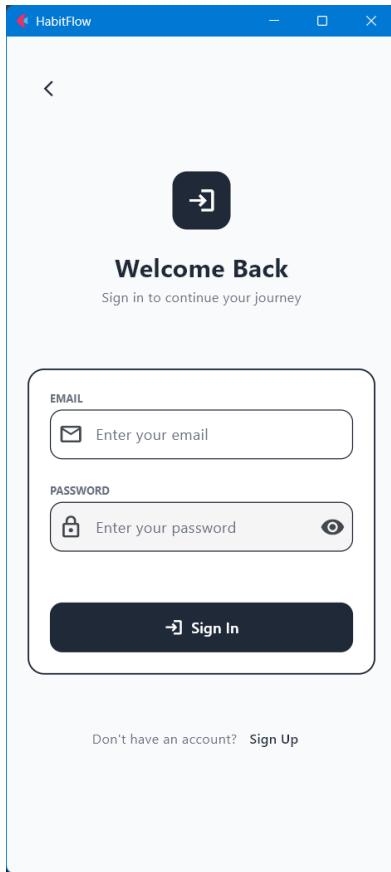


From **Settings → Data Management** you can:

- **Export Data** – download your habits and progress to a JSON file for backup.
- **Import Data** – restore from a previously exported backup.
- **Reset All Data** – remove all habits and completions but keep your account.
- **Delete Account** – permanently delete your account and all stored data.



About & Storage Information



The About section shows:

- App version.
- Short description of HabitFlow.
- Storage summary (total habits/completions, storage location = Local Device).



Admin Dashboard (for Admin Accounts)

User Management

The screenshot shows a "Welcome Back" screen with a "Sign in to continue your journey" button. Below it is a login form with fields for "EMAIL" and "PASSWORD", both with placeholder text "Enter your email" and "Enter your password". A "Sign In" button is at the bottom of the form. At the very bottom of the page, there is a link "Don't have an account? Sign Up".

The **Users** tab in the Admin Dashboard lets administrators:

- View all registered users with their email and basic stats (for example, number of habits).
- See which accounts are marked as **ADMIN**.
- Perform actions such as view-only, disable, or delete a user account depending on what controls your build exposes

Activity Monitoring

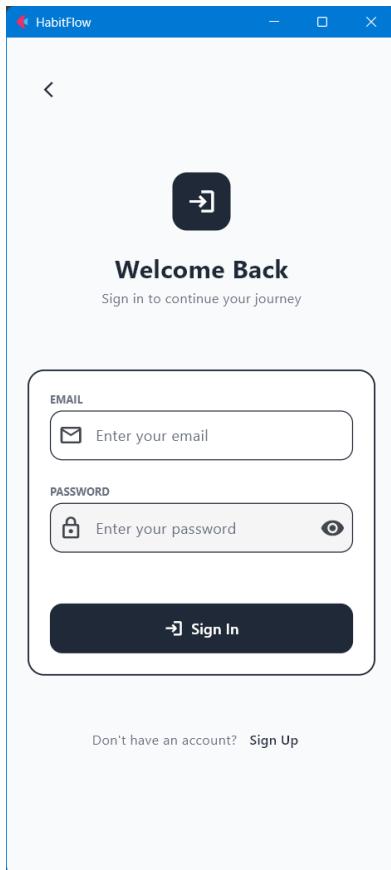
The screenshot shows a "Welcome Back" screen with a "Sign in to continue your journey" button. Below it is a login form with fields for "EMAIL" and "PASSWORD", both with placeholder text "Enter your email" and "Enter your password". A "Sign In" button is at the bottom of the form. At the very bottom of the page, there is a link "Don't have an account? Sign Up".

The **Activity** tab shows recent login activity:

- A list of login events with email address and timestamp.
 - A quick status label such as **Success** for successful logins.
- This view helps admins monitor whether accounts are being accessed as expected and supports manual security reviews.



Security Logs



The **Logs** tab shows security-related events, such as:

- Successful logins, logouts, and sign-ups.
- Failed login attempts that may indicate incorrect passwords or attempted brute-force attacks.

Each entry includes the event type, email, and timestamp, and can optionally be exported for auditing.

Tips for Effective Habit Tracking

- Start with a small number of habits (3–5) so you don't feel overwhelmed.
- Set realistic frequencies (daily or a few times per week) instead of "perfect" schedules you can't maintain.
- Check the Today view at a consistent time each day—morning or evening—to mark completions and review progress.
- Use the Stats tab weekly to see which areas you're improving in and which ones need attention.

Document Prepared By: CodeMates Development Team

Last Updated: December 10, 2025

Version: 1.0.0