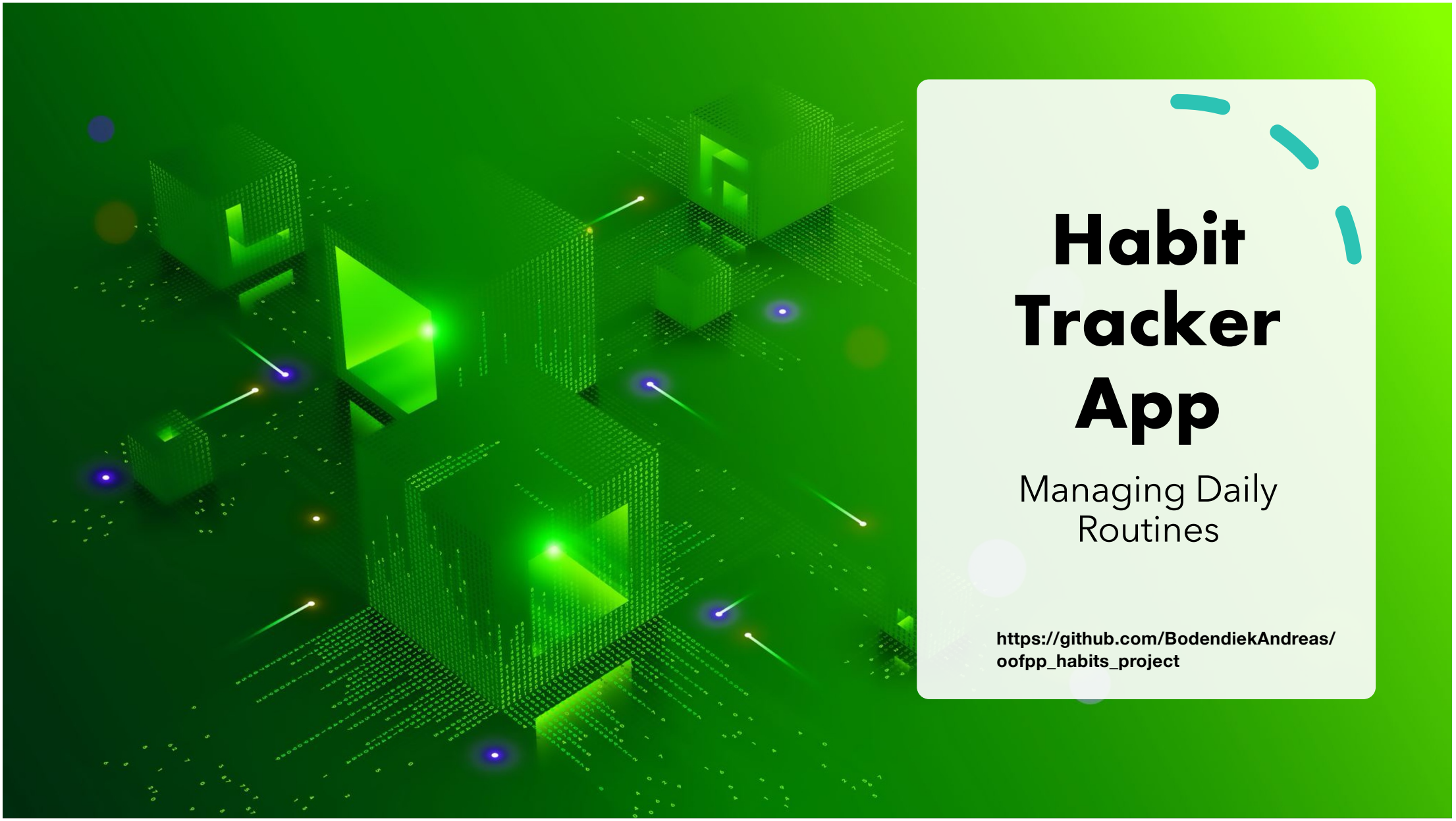
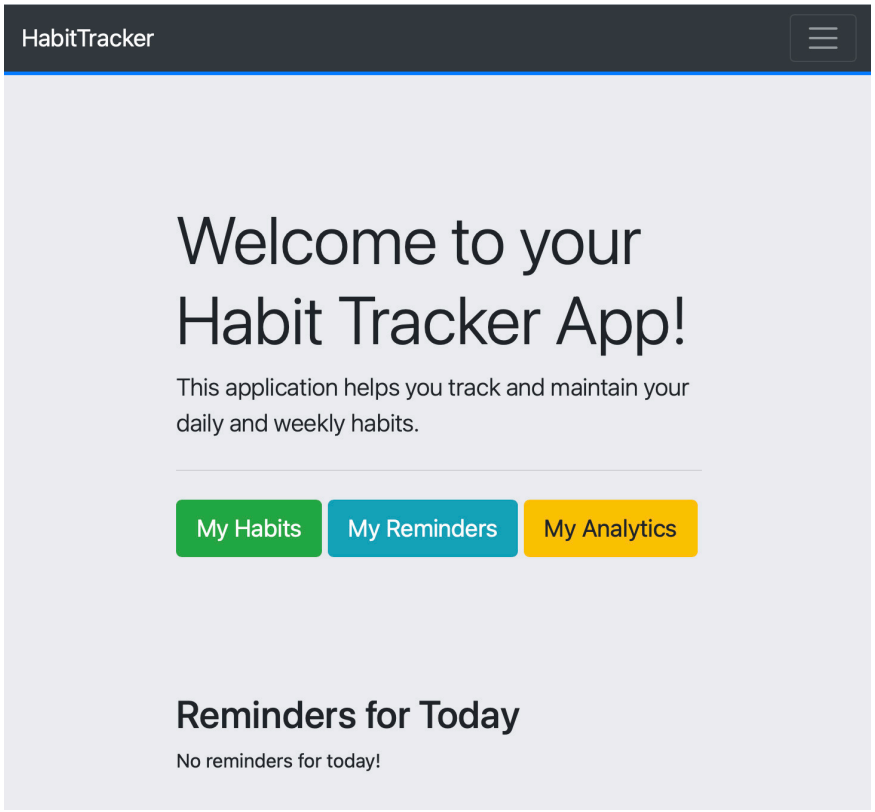


Habit Tracker App

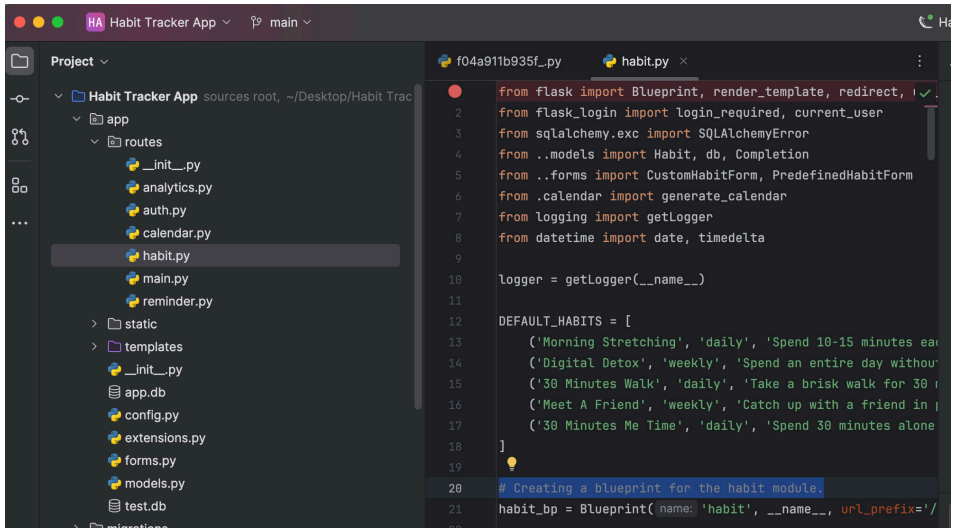
Managing Daily Routines

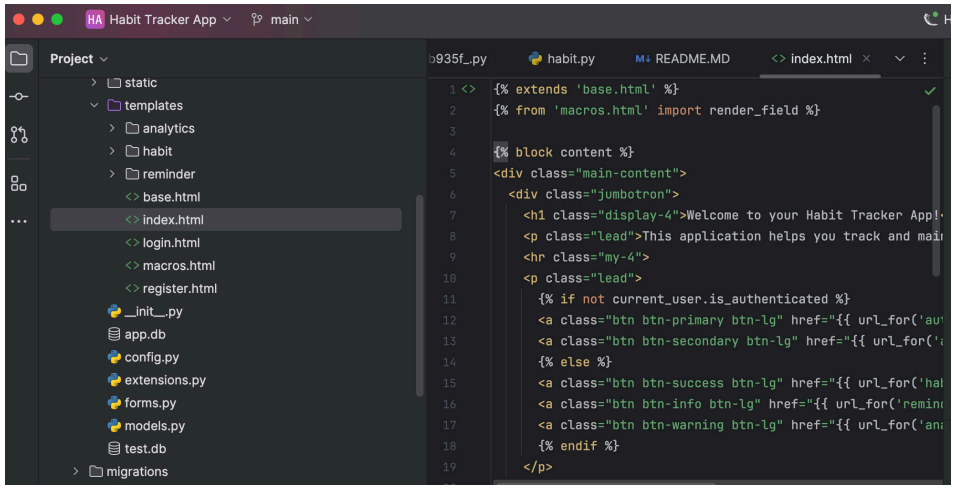
https://github.com/BodendiekAndreas/oofpp_habits_project



Introduction

Welcome to **Habit Tracker App** – the driver for your personal evolution. Our habit tracker app is engineered for motivation, designed for simplicity, and tailored for results. Whether you're chasing goals, elevating routines, or starting fresh paths, **Habit Tracker App** is your daily companion. Let's embark on this transformative journey together, where your next checkmark is a milestone in your success story.





Tools and Frameworks

PyCharm IDE: A Hub for Efficient Development

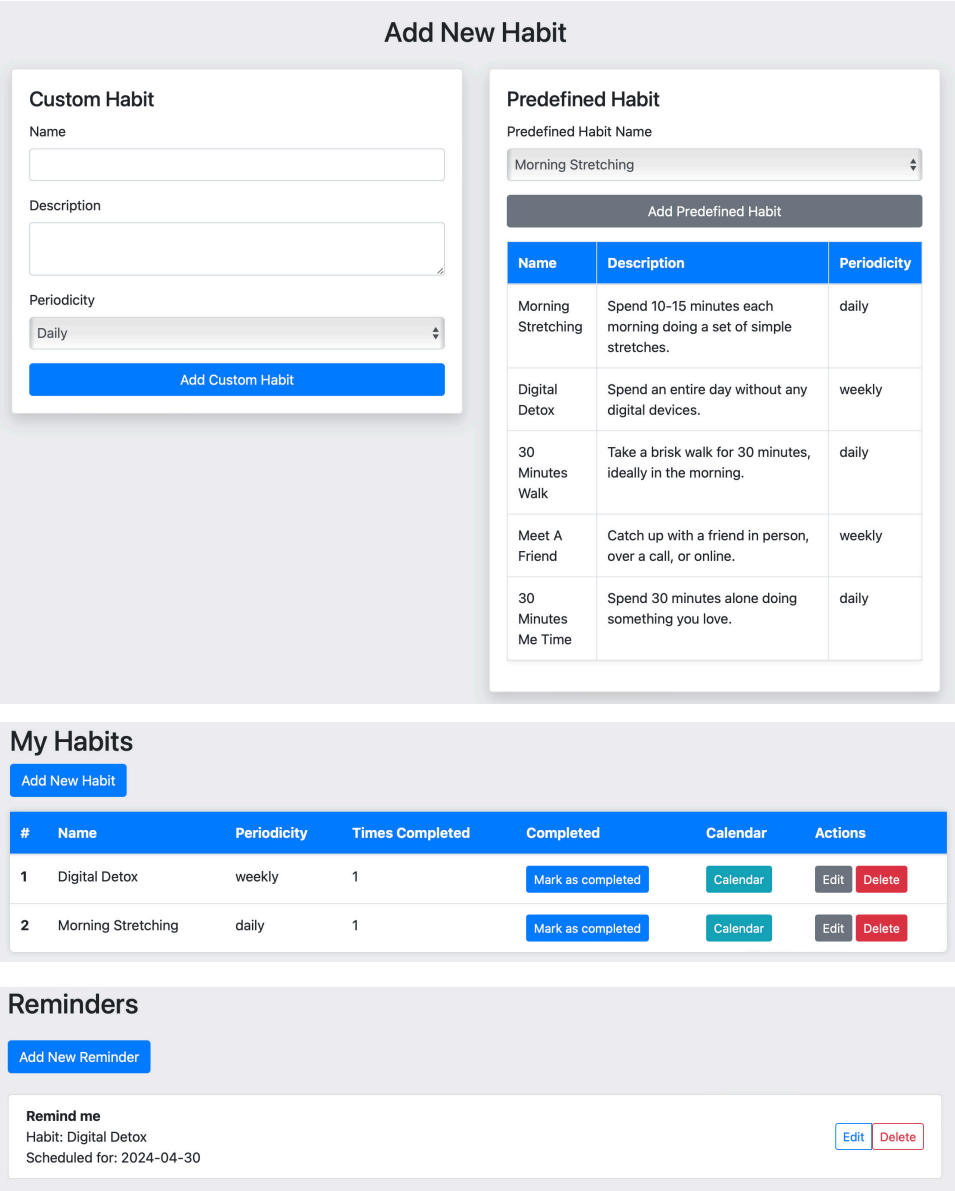
PyCharm emphasizing key features such as integrated debugging, intelligent code completion, and systematic project navigation. These functionalities collectively streamline software development, boosting efficiency and accuracy. PyCharm's intuitive design allows for rapid coding, easy error detection, and seamless project management, thus making it a power-pack tool for robust software creation.

Flask Framework: The Bedrock

Flask is a powerful, flexible micro web framework for Python. Highlights the ease of building robust web applications with clean and pragmatic design. Flask is capable to scale up to complex applications, making it a suitable choice for the project.

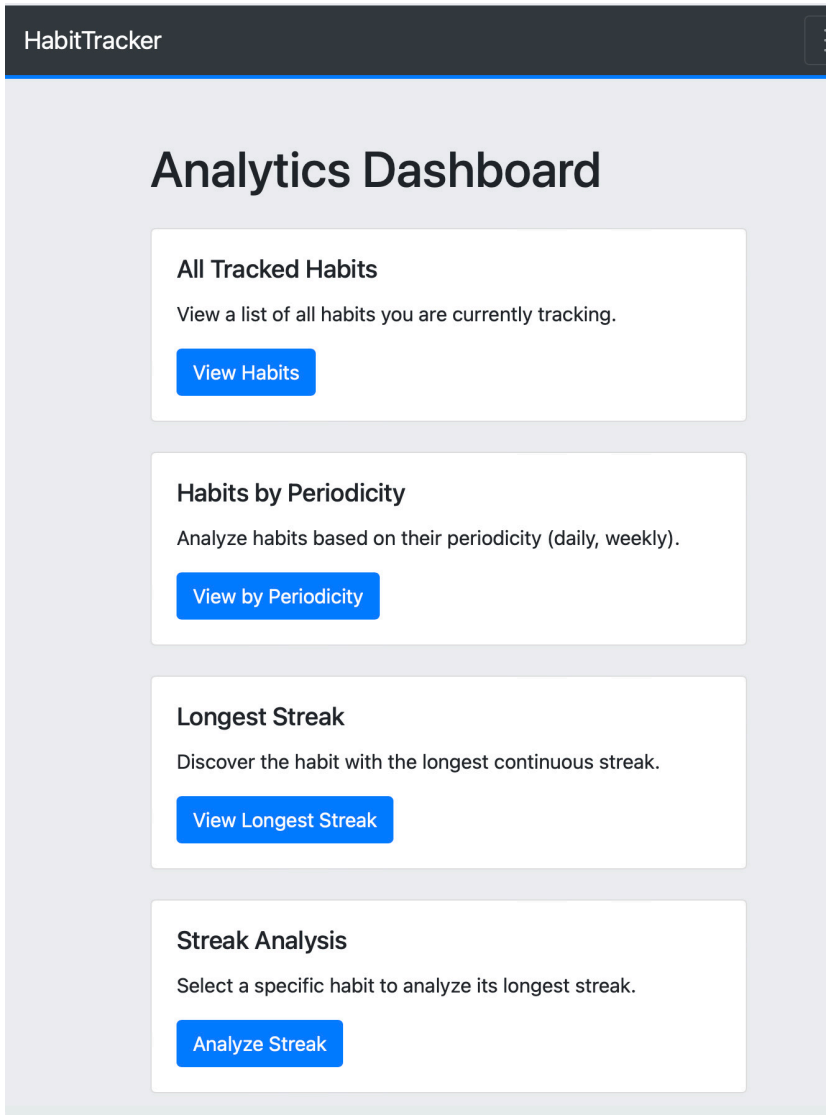
SQLite3 Database: The Data Spine

SQLite3 stands out for its lightweight nature, serverless operation, cross-platform compatibility, transactional integrity, and versatility. It’s a good choice for efficient and reliable data management across various applications.



User Interface and Interaction with App

- **User Registration:** Users starts by creating a new account through the registration form. The form captures details such as username, email, and password. If the credentials are valid and the username or email isn't already taken, the user is registered by storing their data in the database.
- **User Login:** The existing users can login by providing their email and password. The credentials are validated. If they are valid, the user is logged in.
- **Add/Edit Habits:** Once logged in, the user lands on the dashboards where they can add new habits. The habits can be custom defined by the user or they can select from a list of predefined habits. Users can also edit or delete an existing habit.
- **Habit Completion:** Users can mark their habits as completed. The application records the completion time, allowing users to keep a track of their progress. This is supported by a colorcoded calendar.
- **View Progress:** The applications provides analytics to help users monitor their progress. They can review all their habits, filter habits by periodicity (daily, weekly) and see the longest streak of habit completions. This keeps users motivated and encourages them to form new habits.
- **Logout**: Finally, users are able to logout from their accounts, which ends their current session.



Analytics

All Tracked Habits:

Users can view all their habits that are currently being tracked in one place. Each habit displays its details like name, description, periodicity, the count of completions, and the longest streak.

Habits by Periodicity:

Habits can be filtered based on their periodicity. Whether the habits are meant to be daily or weekly, users can segregate and view them accordingly.

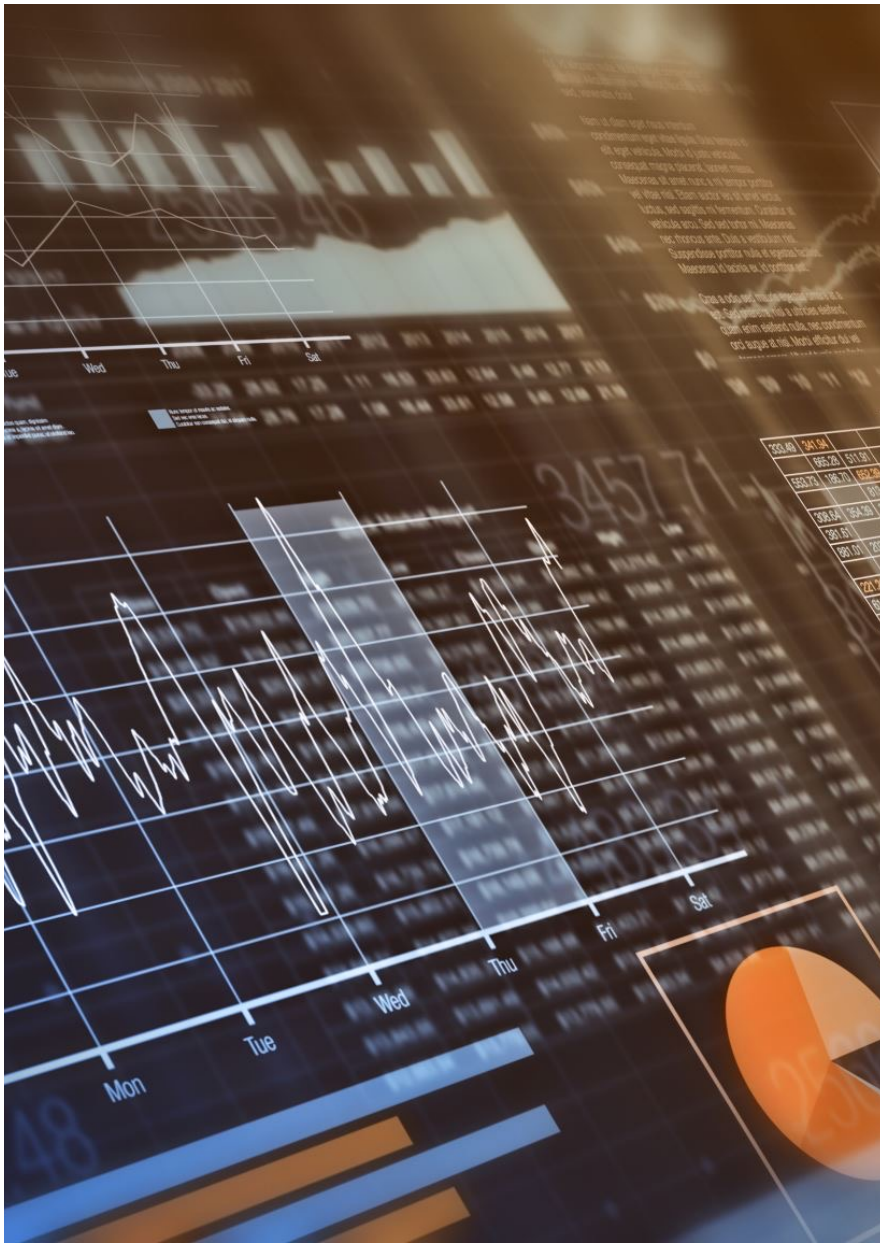
Longest Streak:

The application calculates and displays the longest streak the user has achieved accross all habits. It helps users understand their consistency.

Streak Analysis:

Users can select a specific habit to analyze its longest streak of completion. This deep-dive view allows users to measure their progress in a habit over time.

Python code manages the back-end functionality, analyzing user habits and calculating streaks. The front-end uses Jinja2 templating to deliver a clean and engaging user interface that displays these analytics.



Testing & Reliability

The application uses **pytest**, a Python testing framework. Tests are customized using *pytest.fixture* to set up test instances for the application, database, client, and test entities.

Key components of testing include:

Application Setup: Uses pytest's create_app for application context creation, needed throughout all tests.

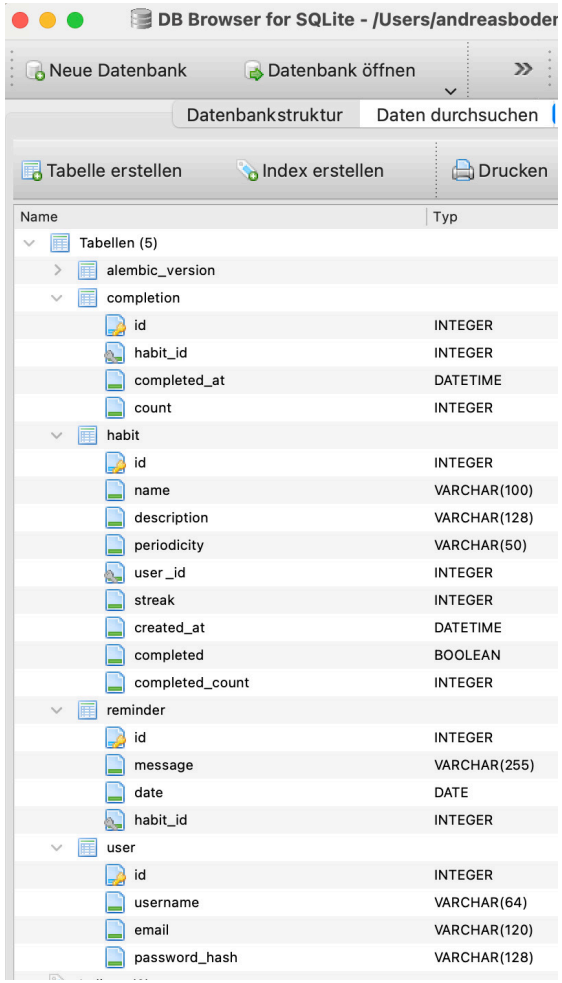
User Management Tests: Checks user registration and login; ensuring user details are processed and stored correctly.

Habit Management Tests: Ensures habits can be created, edited, deleted, and marked as completed.

Reminder Management Tests: Checks the abilities to create, edit, and delete reminders.

Analytics Service Tests: Validates habit processing and calculations, including getting habits, habits by periodicity, and calculating the longest habit streak.

Each set of tests follow the sequence - setup, execute, verify and teardown, ensuring precise, isolated tests.



Storage of Data

The application uses **SQLAlchemy**, a Python SQL toolkit and ORM (Object-Relational Mapping), to facilitate efficient and high-performing database operations. This is integrated into the Flask framework via Flask-SQLAlchemy.

User Data: The User model is used to store attributes such as id, username, email, and password_hash. The User model also includes password hashing functions for added security during user registration and authentication.

Habit Tracking: The Habit model stores information about each habit, including name, description, periodicity (daily, weekly), streak (the longest period a user has maintained the habit), and completed_count (number of times the habit has been completed).

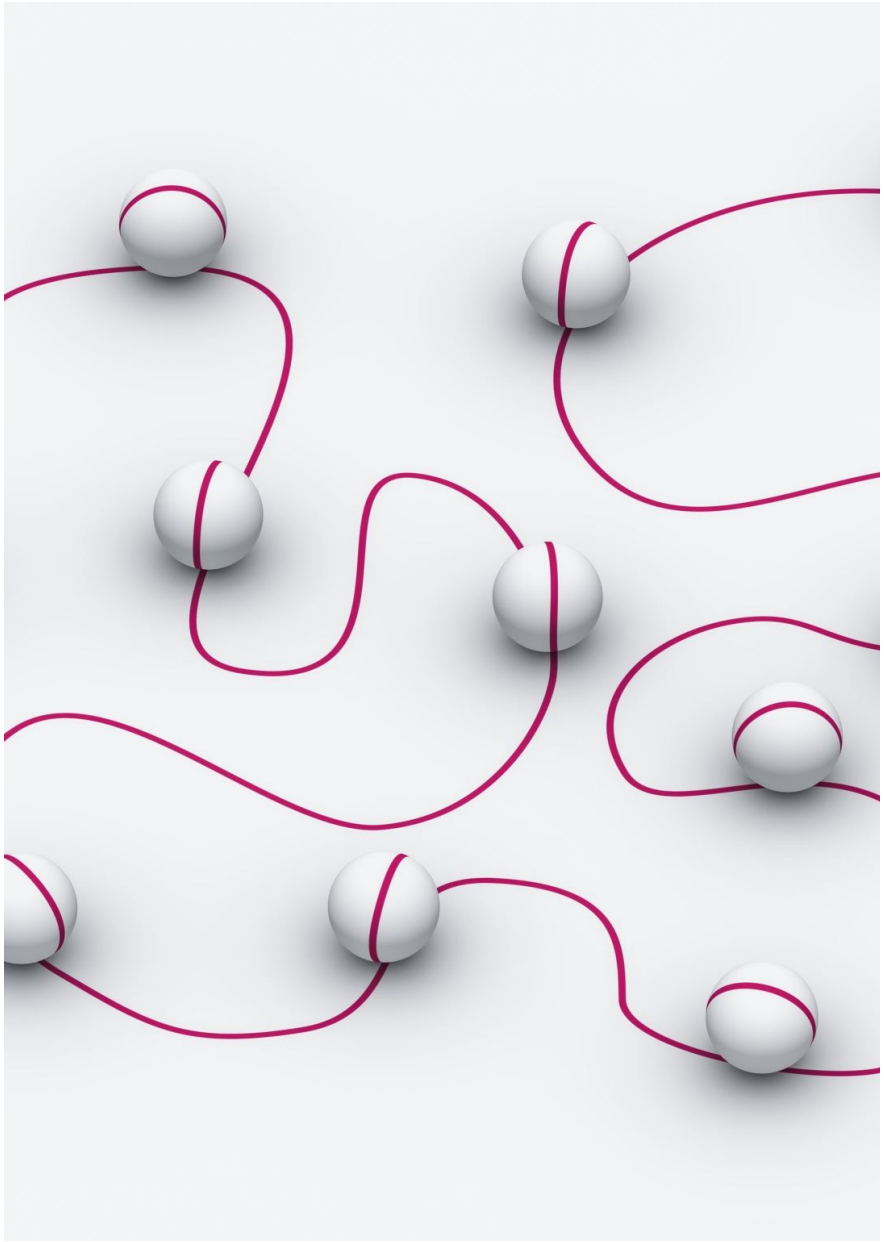
Habit Completion: Each time a user completes a habit, a Completion record is created. This includes a timestamp (completed_at) and a reference to the associated Habit.

Reminders: The Reminder model stores user-defined reminders related to specific habits. Each reminder includes a message and associated date.

These models are linked together using relationships (db.relationship), creating connections between users, their habits, habit completions, and reminders.

The application also uses the **Flask-Migrate** extension, which handles SQLAlchemy database migrations. Database migrations are essentially a version control system for the database, enabling changes to the database schema and data to be managed systematically.

The **Flask-Login** extension provides user session management, handling the common tasks of logging in, logging out, and remembering users' sessions over extended periods.



Conclusion & Contact Information

Habit Tracker App is more than just an app; it's a revolution in personal development. Our innovative platform combines intuitive design with powerful analytics, transforming your daily routines into stepping stones for success.

With features like customizable habit tracking, insightful analytics, and motivational reminders, **Habit Tracker App** is designed to inspire you every day. Whether you're looking to cultivate new habits or maintain existing ones, our app provides the perfect blend of support and challenge.

What makes our app the best choice:

- **Personalized Experience**: Tailor your habit tracking with detailed habit descriptions and flexible periodicity to fit your unique goals.
- **Insightful Analytics**: Dive deep into your progress with analytics that highlight your successes and areas for improvement.
- **Engaging Reminders**: Stay on track with dynamic reminders that keep you focused and motivated.
- **Community and Support**: Join a community of like-minded individuals all striving towards personal betterment.

Stay Connected & Supported

- Your journey is important to us. For any inquiries, support, or feedback, reach out to us:
    - **Email**: andreas.bodendiek@iu-akademie.de
    - **Website**: https://github.com/BodendiekAndreas/oofpp_habits_project