Monica Clara Petrangeli, Juan Pablo Garcia, Sofia Marroquin, Carlota Tejeda

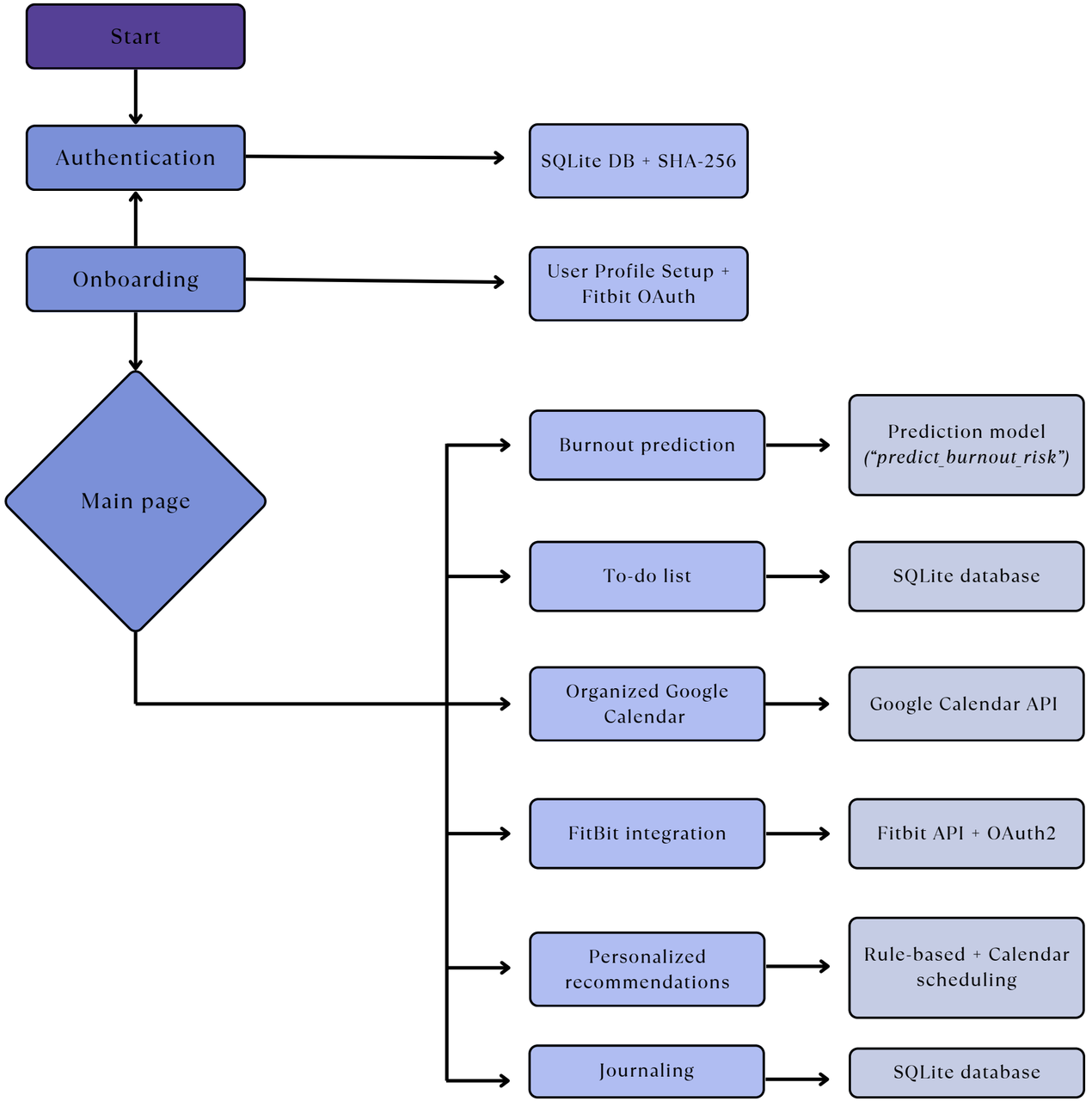
Exhale

Breath out stress, breath in balance

PROJECT OVERVIEW

Code structure

This project is an interactive application designed to help users monitor and reduce their risk of burnout. After signing in and completing a short onboarding, users can access different features such as a personalized burnout risk prediction, a to-do list to manage daily tasks, and a daily stress survey. The app also integrates with Google Calendar and Fitbit to collect real data on habits like sleep, steps, and work hours, giving users a more complete view of their well-being.



PROTOTYPE

Initialization and configuration

Before any user can interact with the app, we need to set the foundations; both for storing user data and for navigating through the app smoothly. In this module, the necessary database tables are created using a **SQLite database**, update the schema if needed, and set the default page to **'sign_in'** to start the user flow correctly.

Key functions:

`create_database()`

Creates all required tables (users, profiles, burnout history, etc.)

`update_database_schema()`

Adds any missing columns without affecting existing data

Navigation set up:

`if 'page' not in st.session_state:`

Sets default page to 'sign_in' to start user flow

This set up ensures the app is ready to handle data and navigation from the very first load.

PROTOTYPE

Stress survey

This module enables users to track their emotional and physical wellbeing by **submitting a short survey each day**. It includes a full flow: checking submission, displaying questions, saving data, and retrieving recent info.

Calling the survey

1. Check if submitted today

`has_submitted_survey_today(user_id)`

Prevents multiple submissions per day by checking the timestamp of the last entry.

2. Saves latest answers

`update_survey_submission_timestamp(user_id)`

Stores the date and time of the latest submission.

3. Load recent survey data

`load_recent_survey_data(user_id)`

Retrieves the latest answers for use in visualizations or recommendations. Applies default values if any data is missing.

Stress survey

`daily_stress_survey()`

- Styled using custom CSS to create a friendly interaction.
- Questions cover:
  - Mood (emoji + label)
  - Stress level (1–10)
  - Work & overtime hours
  - Exercise time
  - Sleep hours
- Responses are saved to the database and the user is redirected to the main page.

This feature promotes daily reflection, helps detect early signs of burnout, and supports personalized feedback later in the app.

PROTOTYPE

Burnout risk prediction

This module calculates the user's burnout risk using a **rule-based model**. It takes into account daily habits and personal factors, and returns an easy-to-understand percentage.

Key function:

`predict_burnout_risk`

Calculates a burnout risk score based on multiple personal and lifestyle variables.

Key code:

```
risk_percentage, score = predict_burnout_risk(age, gender, work_hours, ...)
```

This line calls the burnout prediction function and returns two values: the user's risk percentage and an internal score based on their input data.

Main parameters:

→ *Age*
→ *Gender*
→ *Work*
→ *Sleep hours*
→ *Stress level*
→ *Exercise*

Logic behind:

→ *Sleep and exercise reduce the risk*
→ *Long work hours and high stress increase the risk*

Output:

→ *Burnout percentage*
→ *Raw score (used internally)*

PROTOTYPE

Fitbit

This module connects the app to the **Fitbit API** using secure OAuth 2.0 authentication and allows users to pull personalized health metrics directly from their wearable devices. Token handling is done automatically in the background.

Authorize user access

Launches the Fitbit OAuth flow. The user grants permission via a URL and connects their account to the app.

authorize(email)

Fetch heart rate data

Retrieves minute-by-minute heart rate data for the specified date.

Useful for detecting stress peaks or recovery patterns.

get_heart_rate_data(email, date)

Fetch sleep data

Retrieves detailed sleep metrics, including phases and total duration.

Can help monitor rest quality and identify signs of burnout.

get_sleep_data(email, date)

This functionality gathers data-driven wellness insights, enabling real-time analysis of how users are doing physically and emotionally based on their Fitbit data.

At the moment, we are not gathering real data as a Fitbit wearable is not yet connected.

PROTOTYPE

To-Do List

This module manages a personal to-do list for each user using a **SQLite database**. The four main operations are the following:

1. Add tasks

`save_todo(user_id, task)`

Stores a new task for a specific user and marks it as incomplete by default.

2. View tasks

`get_todo_list(user_id)`

Retrieves all tasks (with their status) for a given user.

3. Update task status

`update_todo_status(user_id, task_id, completed)`

Changes a task's status to either completed or incomplete.

4. Delete a task

`delete_todo(user_id, task_id)`

Removes a specific task from the user's to-do list.

All actions are securely tied to the user via their **user_id** to keep each to-do list personalized.

PROTOTYPE

Google calendar

This module connects the app with the **Google Calendar API** to allow event scheduling and weekly calendar viewing. It includes three key functions:

Authenticate calendar access

→ Manages user authentication using OAuth and returns a calendar service instance.
→ Stores credentials securely for reuse.

`get_calendar_service()`

Personalized recommendation

→ Adds a new activity to the user's calendar, only if no other task has been scheduled in the last 24 hours (walking, meditation).
→ Automatically sets the start time to 5 minutes from now and adjusts the end.

`schedule_event(summary, description, duration_minutes)`

Schedule a custom task

→ Allows the user to schedule an event at a specific date and time, with a custom duration.
→ Ensures correct timezone formatting for Google Calendar.

`schedule_custom_task(summary, description, start_datetime, duration_minutes)`

Get upcoming events

→ Retrieves and formats all calendar events for the next 7 days, using the user's timezone and including each event's summary, timing, and ID.

`get_weekly_calendar_events()`

This system ensures users don't overschedule and allows the app to stay in sync with their real calendar.

PROTOTYPE

Journaling

This module allows users to reflect and write daily journal entries, which are stored in a local database and displayed in an elegant layout. All logic is handled within the main_page() function, under **Tab 2: Journal.**

Main functionalities

1. Create a journal entry

→ *Users can write a title and content using input fields.*

→ *Upon clicking "Save", the entry is saved with the current date.*

2. View past entries

→ *All entries are retrieved from the database and displayed as styled cards, sorted by date (newest first).*

3. Delete entries

→ *Each entry has a small ❌ button to remove it.*

→ *The database is updated, and the page refreshes to reflect the change.*

How it works

Journal entries are saved in a table with:

user_id, title, date, and content.

This feature encourages daily reflection and helps users track their mental and emotional journey over time.

CONCLUSIONS & NEXT STEPS

Conclusions

1. Built for the user

By combining journaling, stress surveys, and a smooth onboarding process, we created a personalized and intuitive experience that helps users in tracking both their mental well-being and time management

2. Real time storage of data

Thanks to SQLite, user data is stored securely and immediately, allowing users to revisit past inputs, stress levels, and survey results without losing any information

3. Ready to grow

The app is designed in a modular way, making it easy to expand with new features in the future, such as additional surveys or advanced analytics

4. Learnings

This project helped us understand better the use of user-centered design, API integration, data management, and how mental health tools can be made approachable and engaging.

Next steps

Add a mood calendar

→ Visualize mood entries over time to help users reflect and identify emotional trends.

Refine burnout prediction model

→ Revisit and adjust assumptions (e.g., age impact, feature weights) to improve how inputs influence burnout scoring.

Deploy the app & improve fitbit

→ Make the platform available online so users can access it anywhere, anytime and connect real fitbit wearable.

Personalize task scheduling

→ Collect user preferences during onboarding to tailor scheduled tasks and boost engagement.

Integrate sentiment analysis

→ Analyze daily journal entries to extract emotions and keywords, offering a more honest and passive way to understand users' mental states, beyond self-reported surveys.

Monica Petrangeli, Juan Pablo Garcia, Sofia Marroquin, Carlota Tejeda

Thank you!