Deliverable 1

Jump to bottom

DanielC19 edited this page on Feb 17 · 21 revisions

1. Introduction

1.1 Purpose

Our software is a tool for people who want to change their habits, by fomenting or discouraging certain actions/behaviors specified by the user.

1.2 Scope

NAX is a tool that provides yet another way of changing habits, this is done by providing the user with the tools to set the actions/behaviors they want to encourage, the web app will remind the users of their selected habits in a time period of their choosing between daily (x hours a day) or weekly (x times a week).

1.3 Product overview

1.3.1 Product perspective

NAX will be a mostly isolated product, it will be developed using Tailwind and Django, most of its interfaces and the like will be developed by our team, but we do plan to integrate the habit tracking with pre-existing services like Google Calendar, but mostly just recollecting data from our own DB and exporting to other calendars for easier user experience. We also plan to integrate OpenAI's GPT-3.5 to give users custom habits generated by AI.

1.3.2 Product functions

NAX's main function is helping users develop habits that they want, by reminding them of the action/behaviors and using AI to generate based on the users info.

1.3.3 User characteristics

Our target group is people above 20 years of age that want to take control of their routines, it is expected to have a minimum of technological knowledge such as be aware of how notifications and reminders work, so our product is pretty accessible since usually the people that would be interested in using our product fit the target group.

1.3.4 Limitations

The biggest limitation is internet connectivity, since it will be necessary for the user to have internet access to be notified of the action/behaviors they have selected, we also rely heavily on the users will to actually change their lifestyle, since most users that use this kind of services will probably start ignoring them after a few days.

1.4 Definitions

- **AI:** Artificial Intelligence. In this case, we are referring to an external AI, used to automatically generate valuable content to the application.
- **API:** Application Programming Interface. Interface that brings an external given functionality or data to the application.
- **Database:** Server where the application's data is stored for later reading in order to display something to the user.
- **NAX:** Navigate, Achieve, eXcellence. Our slogan.
- **TOS:** Terms Of Service.

2. References

The cost of this project, besides all the time invested on it, it's mainly about usage of external APIs (like OpenAI to use GPT-3.5), pricing costs are measured in usage and amount of data. Prices here https://openai.com/pricing. Also, the server and hosting in which we would deploy the application has some cost, but NAX sees a free option more viable in the current phase.

We haven't done any survey so far, but we personally know about many fellow students who feel stressed about not having enough time to do their responsibilities. And that's exactly what we want to help with, managing your time properly, you would even have time to rest, implement new habits of your liking and also identity by yourself toxic habits you have, to diminish or completely delete them from your daily life.

In terms of development, we will be using the following technologies and libraries. They can slightly change over time due to punctual needs found during development.

- Django
  - https://docs.djangoproject.com/en/5.0/
- Django-REST-framework (library-package)
  - https://www.django-rest-framework.org

- HTML
  - https://developer.mozilla.org/en-US/docs/Web/HTML
- CSS
  - https://developer.mozilla.org/en-US/docs/Web/CSS
- Tailwind
  - https://tailwindcss.com/docs/installation
- JavaScript
  - https://developer.mozilla.org/en-US/docs/Web/javascript
- React
  - https://react.dev/

3. Specific requirements

3.1 External interfaces

At first, we are only planning to have two external integrations in NAX. Which can bring a lot of dynamism and comfort for the daily use of the application. Generating valuable content and integrating with other known and used applications, respectively.

- OpenAI - GPT-3.5
  - https://platform.openai.com/docs/introduction
- Google Calendar API
  - https://developers.google.com/calendar/api/guides/overview

3.2 Functions

- **FN-01** - The web app shall display logged user's daily habits.
- **FN-02** - The web app shall track habit completion throughout the week.
- **FN-03** - The web app shall provide the users with the ability to create a daily habit.
- **FN-04** - The web app shall remind users about their daily habits.
- **FN-05** - The web app shall provide the users with the ability to delete a daily habit that was already created.
- **FN-06** - The web app shall provide the users with the ability to edit a daily habit that was already created.
- **FN-07** - The web app shall provide the user with the ability to prioritize some habits that were already created.
- **FN-08** - After creating a habit the web app shall save it in the database within 1 second.
- **FN-09** - The web app shall support user authentication including sign-up functionality.
- **FN-10** - The web app shall provide the users with the ability to logout themselves.

- **FN-11** - The web app should authenticate user’s credentials within 2 seconds with ideal conditions.
- **FN-12** - After a user is registered the web app should create some default habits as recommendation.
- **FN-13** - The web app should display logged user's weekly habits.
- **FN-14** - The web app should provide the users with the ability to create each weekly habit.
- **FN-15** - The web app should provide the users with the ability to delete a weekly habit that was already created.
- **FN-16** - The web app should provide the users with the ability to edit a weekly habit that was already created.
- **FN-17** - After an user accepts TOS, the web app should collect info from the user’s habits to see which are the most common.
- **FN-18** - The web app could be able to integrate a calendar from Google calendar API.
- **FN-19** - The web app could update a pool of new habits each month for user recommendation using AI.
- **FN-20** - If the user pays a membership, the web app could recommend strategies to implement habits powered by AI and based on the user’s data.

3.3 Usability requirements

- **US-01** - The web app shall be consistent in all its visual structure, using the same style and color palette.
- **US-02** - The web app shall load quickly in any device connected to a stable internet connection.
- **US-03** - The web app should provide clear feedback when actions are taken like add or delete a habit to confirm that the action was successful.
- **US-04** - The web app should control errors giving clear and useful error messages.
- **US-05** - The web app should create a habit within 2 seconds under normal load conditions.

3.4 Performance requirements

- **PF-01** - The web app shall load the home page within 3 seconds after login, displaying all current habits and their completion statuses.
- **PF-02** - The web app shall save a user habit into DB within 2 seconds under normal conditions
- **PF-03** - The web app should optimize the database queries to retrieve habit data within 500 milliseconds ensuring fast response times for user interactions
- **PF-04** - The web app should support concurrent processing of at least 20 habit creation/deletion/editing requests per second, ensuring scalability under heavy under interactions.
- **PF-05** - The web app could aim to keep server responses times below 1 second for 90% of requests, optimizing overall user experience

3.5 Logical database requirements

- **DB-01** - The User entity must be related to habit entity in a 1 to 0 or many relationship, since a user can have up to 10 habits at once.
- **DB-02** - The habit and task relationship must be 1 to many since this entity is to check and store when a habit has been completed.
- **DB-03** - Since the habit table will be the most frequently used, the system must be able to handle a lot of activity.
- **DB-04** - The user must be able to always have access to the habits it is connected with, since a user can check at any time.
- **DB-05** - Users must be constrained to only 15 habits in the first iteration to allow an easier handling of the database.

3.6 Design constraints

NAX is a project designed for being a website, that means it shall be responsive for both desktop and mobile. We will be using the last versions of any technology and library we use, trying to keep them up to date, but sometimes it will be inevitable to keep old versions in order to avoid a lot of re-programming. Also, we shall be aware of our own limitations due to knowledge and time, on that regard, we should prioritize tasks and keep them real and doable, avoiding very difficult implementations that are not part of the core functionality of the application, and try to do them only when the core is done.

4. Video

https://youtu.be/nyocKOpG1g4

Clone this wiki locally

```
https://github.com/M4C4NO5/NAX.wiki.git
```