Gift Idea Generator

Gifting made easy
Graded Miniproject

AI Foundations

Fabio Zahner & Silvan Lendi

Eastern Switzerland University of Applied Sciences

1 Introduction

This document details the implementation process for the AI Foundations module Miniproject. The project involves generating and evaluating at least three ideas that leverage a GPT-based assistant and building an application around the chosen concept.

The process begins with idea generation and selection based on defined evaluation criteria. Once an idea is selected, we refine it and develop specifications for the application.

Our goal was to choose an idea with practical utility—something we would personally use. During brainstorming, we considered common challenges where a large language model (LLM) could assist us. Within a short time, we generated three ideas:

- A food recipe generator that provides recipes based on ingredients available at home (user input).
- A gift idea generator, inspired by the upcoming holiday season, which suggests gifts based on characteristics of the recipient.
- A belt balancer generator for the game Factorio[1].

Next, we evaluated the practicality and feasibility of each idea. While all were potentially useful, the belt balancer idea presented significant complexity and implementation challenges. A belt balancer in Factorio helps distribute items across multiple input and output belts. Factorio allows players to import such structures as blueprint strings, which are encoded in base64 and compressed using zlib before translating into JSON for in-game structures.

Although it initially seemed promising, generating a valid belt balancer string with an LLM proved difficult. Even after ten attempts, none of the generated strings imported successfully into the game, so we decided to eliminate this idea.

We established the following criteria to evaluate the remaining ideas:

- Practical use (weight: 30)
- Ease of implementation (weight: 5)
- Benefit (weight: 20)
- Originality (weight: 10)

Idea	Practical Use	Ease of Implementation	Benefit	Originality	Total Score
Recipe Generator	5 * **30**	8 * **5**	7 * **20**	5 * **10**	**400**
Gift Idea Generator	7 * **30**	8 * **5**	9 * **20**	8 * **10**	**510**

The Gift Idea Generator scored highest, so we selected it as our project.

Before implementation, we defined specifications for the application. We agreed on the following requirements: The application should accept three input parameters: Gender, Age, and Personal Interests. It should also be accessible on mobile devices for convenient use on the go. When the user enters the parameters and presses the button to generate ideas, they should receive five gift suggestions in text form.

[1] Not affiliated with the product; however, we recommend trying it.

2 Implementation: Backend

In order to make use of ChatGPT in your application so called "assistants" can be used by calling the OpenAI-API. We created our assistant under the name “fabio_silvan_giftassistant” and provided it with system instructions. It was instructed to provide 5 gift ideas and to parse the input as constraints accordingly. As a model we chose “gpt-4o-mini” as it is compatible with json_schema-formats (see A.1) and more than capable for our task. The json-schema helps control the output format which in turn makes parsing the returned message in our application a lot easier.

3
Implementation: Frontend

To create an application that integrates into daily life, we will use the Flutter technology stack to develop software accessible across multiple platforms. For development, we have focused on optimizing the application for Android mobile phones. However, if we decide to expand the application to other platforms, it would require minimal additional work.

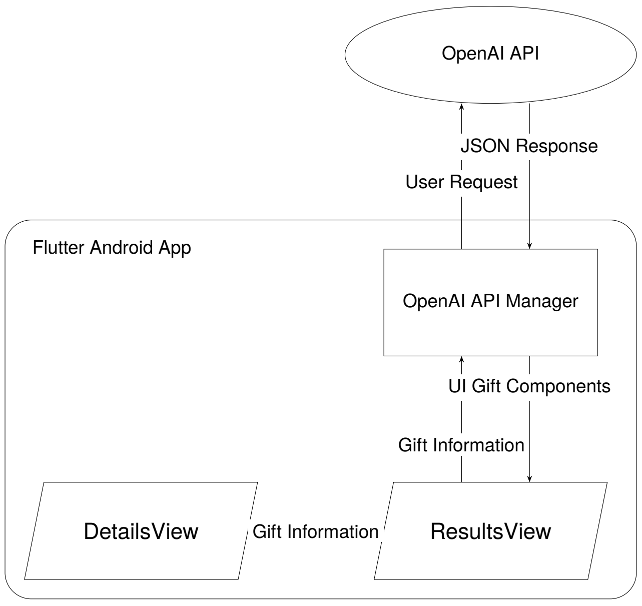


Figure 3.1: Planned architecture of the App

Figure 3.1 describes the planned architecture of the Flutter App. It has two main screens, which the User is guided through:

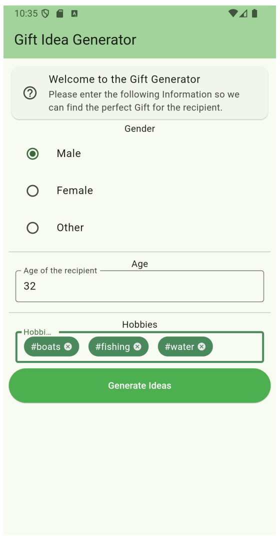
1. In the `DetailsView`, the User can specify different attributes about the Person they would like to gift something to (furthermore called the Recipient)
2. As soon as the User submits the Information, all Gift Information is Passed to the `ResultsView`.
3. The `ResultsView` passes the received Information to the API Manager, which converts it into a prompt, which is then sent to the OpenAI API.
4. The OpenAI Assistant processes the Request and returns a Response in Form of a JSON Object. (For more details see Section **??**)
5. The API Manager receives the JSON Object, extracts the relevant Information and sends it to the `ResultsView` in Form of prebuilt widgets, where they are displayed to the User.

User Interface

For the Design, we wanted a simple but clear User Interface. We used the Material Design Guidelines to ensure a consistent Design. At first, we created a basic UI to test the app's core functionality. After developing the initial version of the app, we proceeded to validate the API Manager's ability to convert JSON responses into visually appealing Widgets.

Once the functionality had been confirmed, we focused on refining the User Interface. While not the primary objective of our project, we recognized that an improved UI can significantly enhance the overall user experience. We therefore implemented the following changes:

- A small infobox which instructs potential first-time users.
- Improved display of input fields for age and hobbies.
- The option to specify the Gender when selecting "Other".
- Added titles for each Gift Idea to improve overview.
- A button to instantly search for a gift on `galaxus.ch`



10:35

Gift Idea Generator

Welcome to the Gift Generator
Please enter the following Information so we can find the perfect Gift for the recipient.

Gender

[✓ Male]
Female
Other

Age

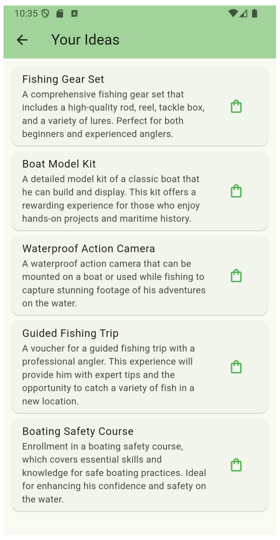
Age of the recipient
32

Hobbies

Hobbi...
#boats #fishing #water

Generate Ideas

(a) `DetailsView`



10:35

Your Ideas

Fishing Gear Set
A comprehensive fishing gear set that includes a high-quality rod, reel, tackle box, and a variety of lures. Perfect for both beginners and experienced anglers.

Boat Model Kit
A detailed model kit of a classic boat that he can build and display. This kit offers a rewarding experience for those who enjoy hands-on projects and maritime history.

Waterproof Action Camera
A waterproof action camera that can be mounted on a boat or used while fishing to capture stunning footage of his adventures on the water.

Guided Fishing Trip
A voucher for a guided fishing trip with a professional angler. This experience will provide him with expert tips and the opportunity to catch a variety of fish in a new location.

Boating Safety Course
Enrollment in a boating safety course, which covers essential skills and knowledge for safe boating practices. Ideal for enhancing his confidence and safety on the water.

(b) `ResultsView`

Figure 3.2: Final Version of the Application

OpenAI API Manager

The `OpenAI API Manager` is the backend logic which allows the Application to communicate with the OpenAI Assistant. It provides the functionality to convert User Input into a usable text prompt, as well as convert the received JSON into a user-friendly interface.

```
factory GiftResponse.fromJson(Map<String, dynamic> data) {
    GiftResponse gr = GiftResponse();
    for (Map<String, dynamic> idea in data['presents']) { // Extract "present" objects
      gr.ideas.add(Idea(idea["title"], idea['description'])); // Fill information into new Idea() Object
    }
    return gr; // Return a GiftResponse() Object containing Ideas()
}


List<Widget> cards(Set<Idea> ideas) {
    List<Widget> cards = List.empty(growable: true);
    for (Idea idea in ideas) { // For all Ideas in the List
        cards.add(Card( // Add a new widget (Flutter Card Widget in this Case)
        child: ListTile(
            title: Text(idea.title), // Title
            subtitle: Text(idea.description), // Description
            trailing: IconButton( // Shopping Bag Button
                onPressed: () {
                    _launchUrl(
                        "https://www.galaxus.ch/de/search?searchSectors=0&q=${idea.title}");
                },
                icon: Icon(Icons.shopping_bag_outlined, color: Colors.green,))),
        ));
    }
    return cards;
}
```

Figure 3.3: Code section responsible for converting JSON into Widgets

4 Conclusion

In this project, we successfully developed a Gift Idea Generator, demonstrating the practical application of Large Language Models in solving real-world challenges. By leveraging the OpenAI Assistant through a Flutter-based mobile application, we created a tool that simplifies the often challenging task of finding the perfect gift.

During Developement, we encountered the following challanges:

- Prompt engineering: Creating effective prompts to generate the desired JSON
- JSON parsing: Handling and processing JSON data received from the API, with Error handling in case returned JSON is not valid.
- User interface design: Creating an intuitive and visually appealing user interface.
- API integration: Integrating the OpenAI API into the Flutter app.

While we successfully developed a functional Gift Idea Generator, we recognize that we didn't fully exploit the potential of the OpenAI Assistant. More advanced features like Code Interpreters or external API integrations could have significantly enhanced the app's capabilities. Although, given the inherent complexity of AI integration in this relatively straightforward use case, we are satisfied with the current outcome.

Future Work

While our current implementation focuses on Android mobile platforms, the Flutter framework allows for easy future expansion to other platforms. Potential future improvements could include:

1. Expanding the range of input parameters
2. Implementing more sophisticated filtering of gift ideas
3. Adding user feedback functionality to improve recommendation accuracy
4. Supporting multiple languages

A

Source Code Appendix

```
{
    "name": "present_response_schema",
    "strict": true,
    "schema": {
    "properties": {
        "presents": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "description": {
                        "type": "string",
                        "description": "A detailed description of the present."
                    }
                },
                "required": [
                "description"
                ],
                "additionalProperties": false
            }
        }
    },
    "additionalProperties": false,
    "required": [
        "presents"
    ],
    "type": "object"
    }
}
```

Figure A.1: json schema