Grocery List App: Final Report

Amir Mohseni

November 1, 2024

Contents

1 Introduction

Managing grocery lists manually can be inconvenient and time-consuming. While various grocery list apps exist, most lack intelligence, requiring users to manually organize items without adapting to their preferences. Implementing AI for automated categorization can save users time and effort.

This project focuses on developing a smart grocery list app that uses AI to predict item categories, streamlining the shopping process. The report covers requirement extraction, low-fidelity prototyping, AI integration, user testing, and future improvements, including immersive technologies and explainability.

2 Requirement Extraction

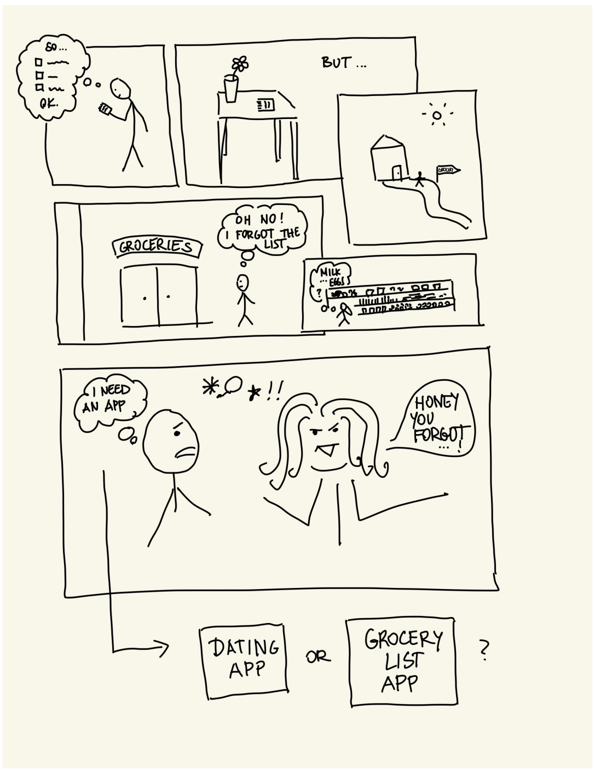
To ensure that the grocery list app met user needs, requirements were collected through a survey distributed to 16 participants, mainly university students and postgraduates (see Appendix A.3 for the questionnaire used). Participants provided feedback on essential features like ease of use, item categorization, and AI-driven suggestions. These insights were used to identify functional and non-functional requirements.

2.1 User Needs

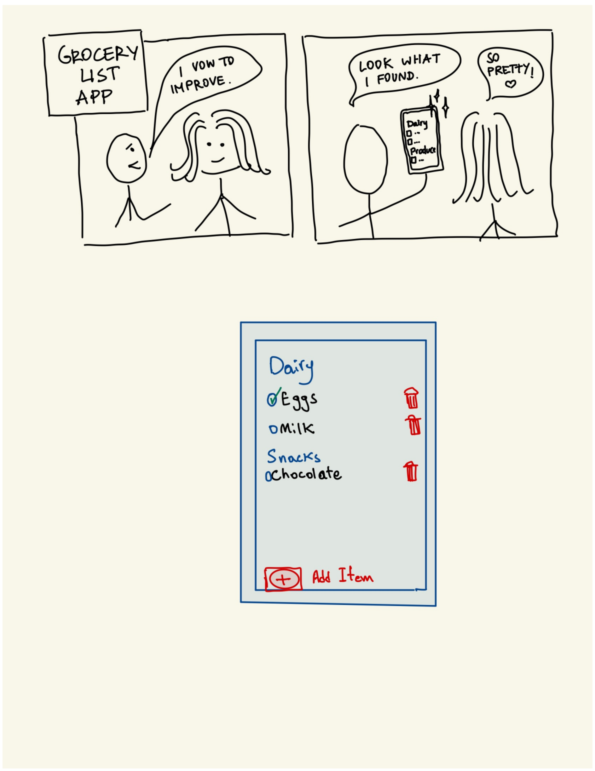
Based on user feedback, key requirements include quick item entry, effective categorization (predefined and user-defined), and AI-driven predictions based on user behavior. Detailed user needs and full requirement lists are provided in Appendix A.1.

2.2 Low-Fidelity Prototype

After extracting the core requirements from users, we developed a low-fidelity prototype storyboard to visualize the key interactions and functionality of the system. The storyboard, as shown in Figures 1a and 1b, playfully illustrates how users might use the application and how it might enhance their experience.



(a) Part 1



(b) Part 2

Figure 1: Storyboard of the Low-Fidelity Prototype

3 Hard-Coded Task and AI Integration

After extracting user requirements and creating the low-fidelity prototype, we developed the first version of the application. It was designed to implement most of the identified user requirements, but without the AI-driven task automation, to establish a baseline for comparison.

3.1 Description of the Hard-Coded Task

In the hard-coded version, users manually add items, create categories, and assign items to categories. Categorized items are displayed under their respective sections, helping users easily distinguish between different categories and enhancing the shopping experience. This version is developed to compare with the AI-enhanced version, isolating the impact of AI integration on user experience. (For more details, refer to the user manual in the Appendix A.4).

3.2 Tested Principles

The AI-enabled app focuses on enhancing the **flexibility principle**, allowing users to adjust AI predictions and switch between AI-driven and manual categorization. The app supports **Dialog Initiative** by enabling manual overrides, **Multi-threading** with real-time predictions, and **Task Migratability** through flexible task handling. Users can choose to rely on AI or manually categorize items, demonstrating **Substitutivity**. Additionally, the AI learns user behavior over time, improving personalization and **Customizability**.

3.3 AI Integration

The AI-powered categorization is based on a logistic regression model, trained using both user survey data and synthetic data generated via ChatGPT. The model is further enhanced by using pre-trained Word2Vec embeddings to capture semantic relations. Detailed data collection processes and model improvements are discussed in Appendix A.9.

3.3.1 Model Adaptability

We wanted the model to learn from its mistakes and adapt to user preferences. Hence, we collected the instances where the user would manually change the AI-predicted category and use them to retrain the model for greater personalization.

3.4 Prototypes

Figure 2, Figure 3 and Figure 5 demonstrate the design and workflow of both app versions.

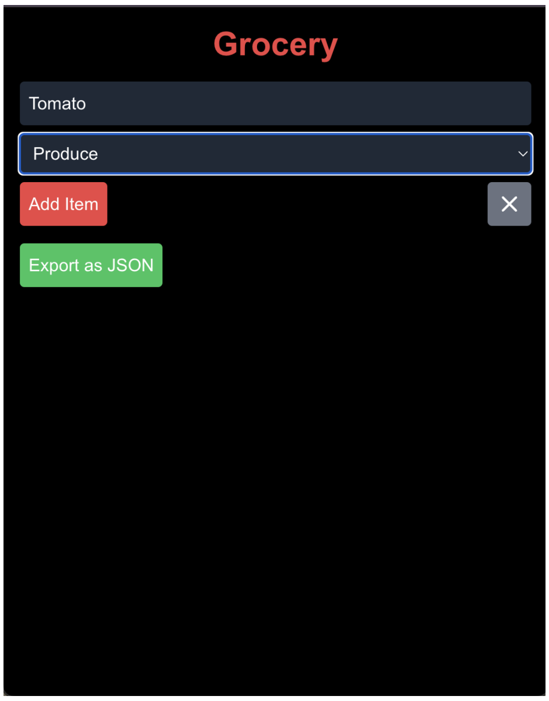


Figure 2: Manual Version (Hard-Coded). The hard-coded version of the interface requires users to assign categories manually

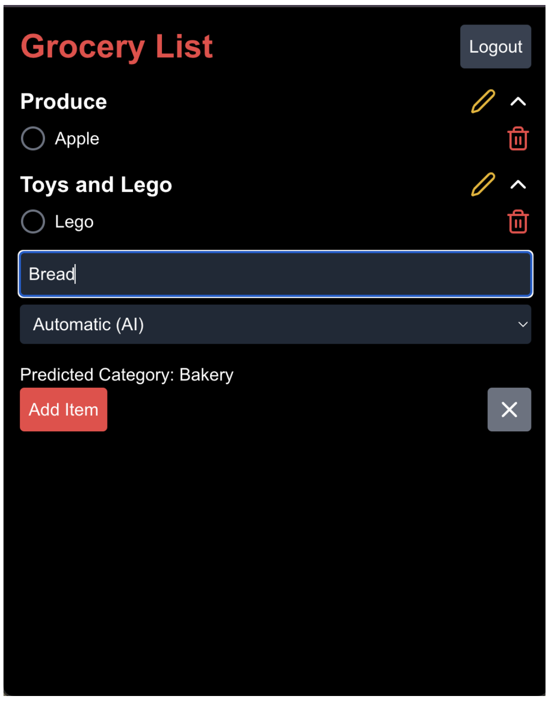


Figure 3: AI-Integrated Version. The system automatically predicts categories based on user input

Figure 4: Comparison of Key Interfaces: Hard-Coded vs. AI-Integrated

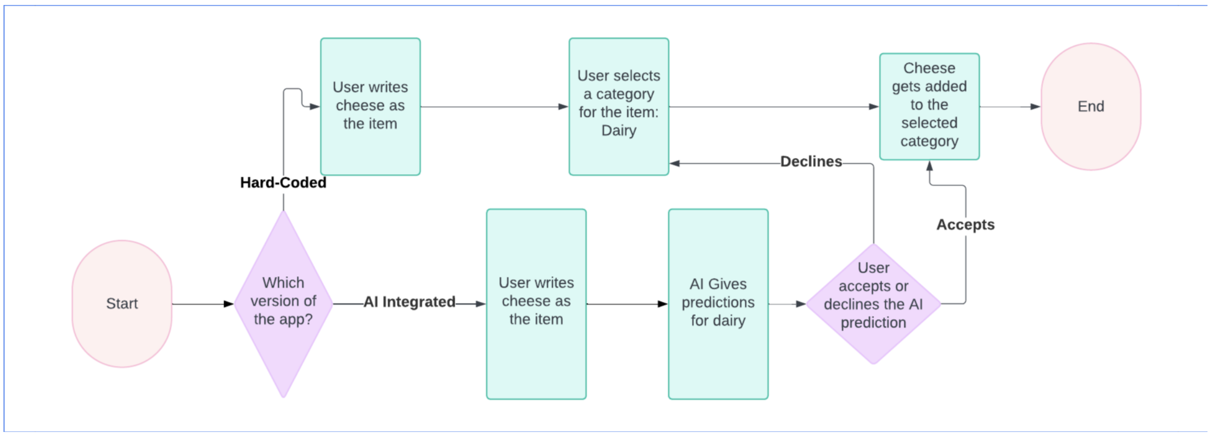

Start
Which version of the app?
Hard-Coded
AI Integrated
User writes cheese as the item
User selects a category for the item: Dairy
Cheese gets added to the selected category
End
User writes cheese as the item
AI Gives predictions for dairy
User accepts or declines the AI prediction
Declines
Accepts

Figure 5: Comparison between the workflows of AI and Hard-Coded Versions

Links:

- Link to functional prototype
- Link to video demonstration

4 Methodology

This study aimed to evaluate the performance and usability of the AI-enabled grocery list app compared to a manual (hard-coded) version. The focus was on task completion time, user emotions, and app flexibility. The data were collected using user testing and Likert scale-based questionnaires.

4.1 Data Collection

4.1.1 Participants

A total of 12 participants, aged between 15 and 59 years, participated in the user testing. The group consisted of 58.3% male and 41.7% female participants. The majority of the participants came from academic backgrounds, representing both technical and non-technical fields.

4.1.2 User Testing Setup

User testing was conducted in two configurations: one with AI-driven categorization and the other without AI (hard-coded task). Each participant completed two rounds of tasks using one version of the app per round. The rounds were not conducted consecutively to avoid bias.

Task 1: Participants were asked to add 10 pre-determined grocery items to the list. These items were the same for all participants.

Task 2: Participants added 10 grocery items that they typically buy from the store. These items were unique to each participant and determined before the experiment began.

In the manual version, participants categorized items manually, whereas in the AI-enabled version, the app automatically categorized items, allowing participants to correct the AI's predictions if necessary.

4.1.3 Questionnaires

After using each version, participants filled out a questionnaire assessing their emotions, task completion experiences, and the flexibility of the app. Emotional responses were recorded on a 5-point Likert scale and categorized using Ekman's emotional model.

4.2 Data Analysis

4.2.1 Task Completion Time

Task completion time was recorded in seconds for both tasks and for each version of the app. A **Paired T-test** was conducted to compare the average task completion time between the manual and AI-enabled versions, as the data followed a normal distribution according to the Shapiro-Wilk test.

4.2.2 User Emotions

User emotions were measured using a Likert scale, and emotional data was mapped to categories based on Ekman's model:

- **Enjoyment:** Satisfaction, Excitement, Confidence, Efficiency
- **Sadness:** Overwhelmed
- **Fear:** Confusion
- **Disgust:** Boredom
- **Anger:** Frustration, Annoyance
- **Surprise:** Interest

A **Wilcoxon Signed-Rank Test** was applied to analyze differences in emotional responses between the manual and AI-enabled versions since the data were ordinal.

4.2.3 App Flexibility and Usability

Flexibility was evaluated using Likert scale-based questions regarding:

- **Override AI Predictions:** Ease of adjusting or overriding the AI's category predictions.
- **Control:** User's sense of control over the categorization process.
- **Time Savings:** Reduction in time and effort with the AI-enabled version.
- **Adaptability:** AI's ability to learn from the user's behavior and improve over time.

A **Wilcoxon Signed-Rank Test** was also used to analyze the differences in flexibility ratings between the two versions.

4.2.4 Normality Test

The **Shapiro-Wilk test** was performed to assess the normality of the task completion time data. Since the p-value was greater than 0.05, indicating a normal distribution, parametric tests such as the **Paired T-test** were used for task completion time analysis. For ordinal data, non-parametric tests were used without assuming normality.

5 Results

5.1 Task Completion Time

The results from the Paired T-test are summarized below, comparing task completion times for Task 1 and Task 2 between the AI-enabled app and the manual version.

Task	Shapiro-Wilk Statistic (W)	p-value
Task 1	0.9612	0.8004
Task 2	0.9527	0.6770

Table 1: Shapiro-Wilk Test results for normality of task completion time data. Both Task 1 and Task 2 data are normally distributed.

Task	Version	Mean Time (seconds)	p-value
Task 1	Manual	107.4	**5.02e-07**
Task 1	AI-enabled	62.5	
Task 2	Manual	121.3	**3.52e-05**
Task 2	AI-enabled	76.4	

Table 2: Comparison of task completion times between manual and AI-enabled versions for Task 1 and Task 2.

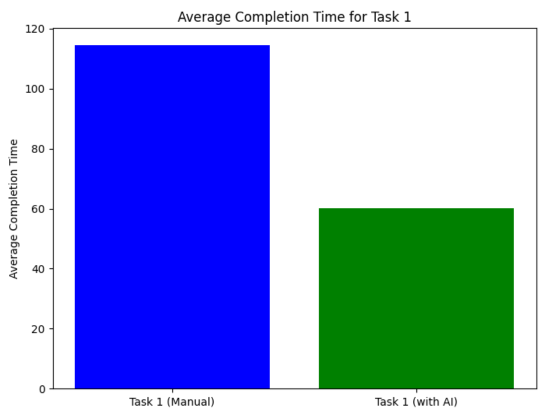


Figure 6: Average completion time for Task 1.

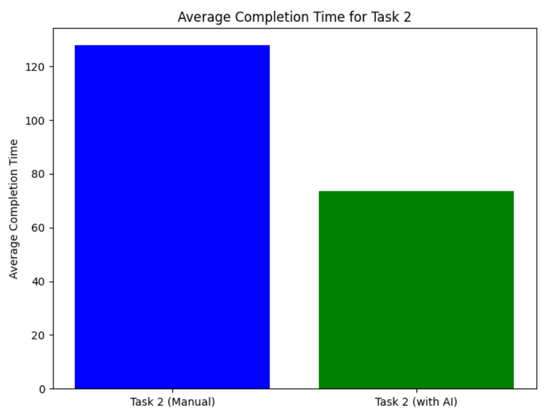


Figure 7: Average completion time for Task 2.

5.2 User Emotions

The emotional responses were collected using a 5-point Likert scale across various emotions.

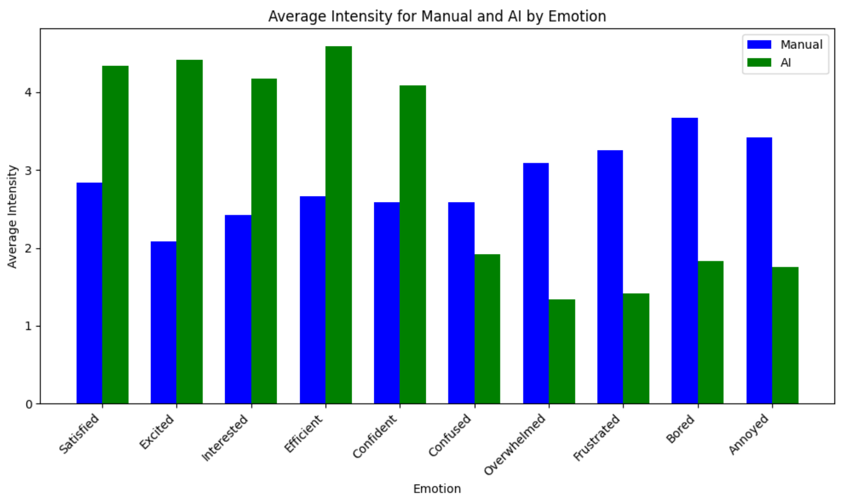


Figure 8: Average emotional intensity of AI-enabled and manual versions by emotion.

Emotion	**Ekman Category**	**Statistic**	**p-value**	**Change (AI vs Manual)**
Satisfied	Enjoyment	0.0	0.0044	Significantly greater
Excited	Enjoyment	0.0	0.0047	Significantly greater
Interested	Surprise	2.5	0.0093	Significantly greater
Efficient	Enjoyment	0.0	0.0072	Significantly greater
Confident	Enjoyment	2.0	0.0144	Significantly greater
Confused	Fear	3.5	0.1389	No significant change
Overwhelmed	Sadness	0.0	0.0097	Significantly lower
Frustrated	Anger	2.0	0.0053	Significantly lower
Bored	Disgust	0.0	0.0047	Significantly lower
Annoyed	Anger	3.0	0.0199	Significantly lower

Table 3: Wilcoxon Signed-Rank Test results for user emotions (AI-enabled vs. manual), categorized by Ekman emotional model.

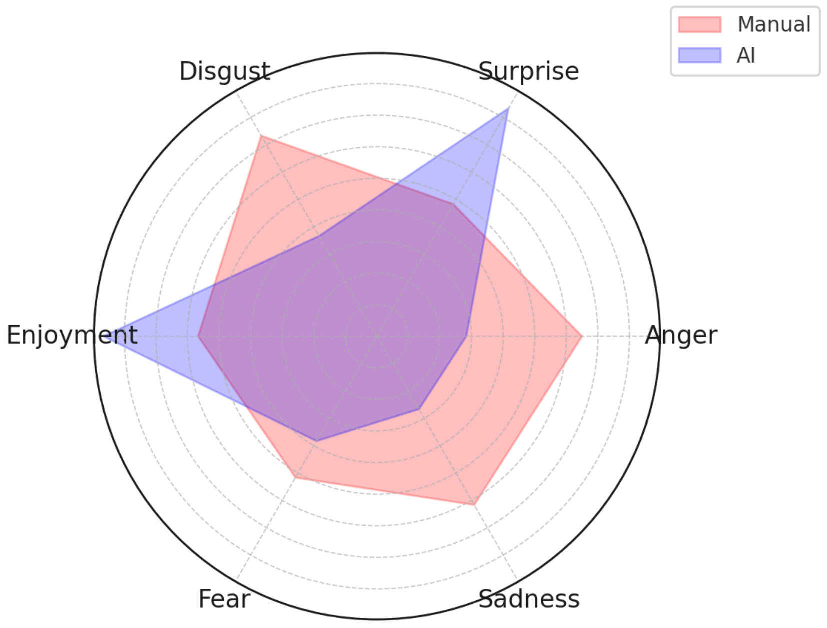


Figure 9: Comparison of user emotions using Ekman's model between AI-enabled and manual prototypes.

Emotion	Manual	AI
Satisfied	2.833333	4.333333
Excited	2.083333	4.416667
Interested	2.416667	4.166667
Efficient	2.666667	4.583333
Confident	2.583333	4.083333
Confused	2.583333	1.916667
Overwhelmed	3.083333	1.333333
Frustrated	3.250000	1.416667
Bored	3.666667	1.833333
Annoyed	3.416667	1.750000

Table 4: Comparison of emotions between Manual and AI prototypes.

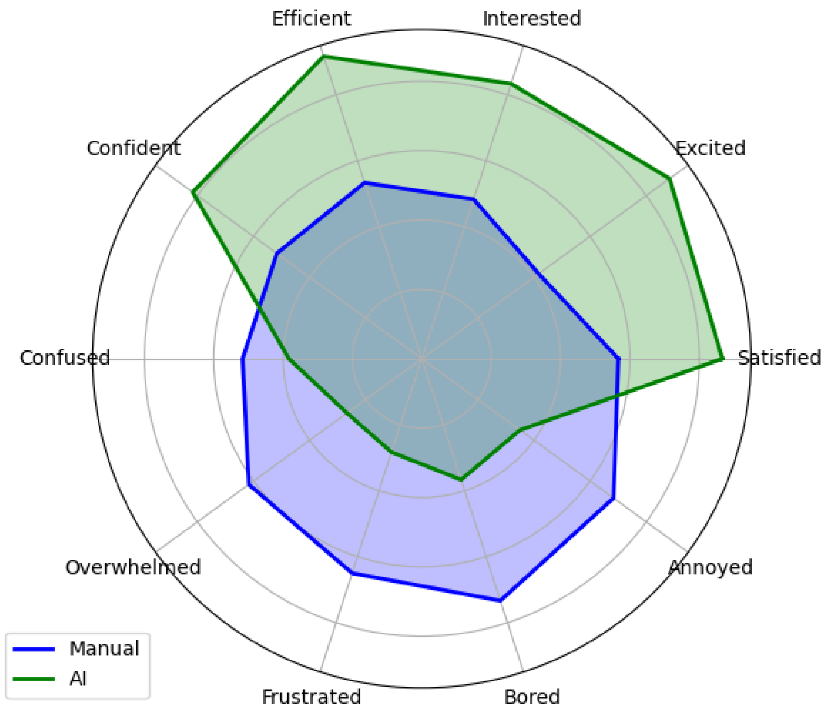


Figure 10: Average emotional intensity of AI-enabled and manual versions.

5.3 Usability and Flexibility

The usability and flexibility results, measured through various Likert scale questions, are summarized in Table 5.

Metric	Average Score (out of 5)
AI Prediction Accuracy	4.08
Flexibility to Override AI Predictions	4.33
User Control over Categorization	4.42
Time Savings Compared to Manual Categorization	4.58
Adaptability to User Shopping Habits	4.42

Table 5: Summary of Usability and Flexibility Ratings (AI-enabled version).

6 Discussion

The results of this study clearly indicate that the AI-enabled grocery list app significantly improved user performance and satisfaction compared to the manual (hard-coded) version. The integration of AI had a positive impact on task completion time, user emotions, and app flexibility, as summarized below.

6.1 Task Completion Time

One of the most striking improvements observed was the reduction in task completion time. Users were able to complete both tasks substantially faster using the AI-enabled version. The AI's ability to automate the categorization process eliminated the need for manual item entry and correction, reducing cognitive load and minimizing repetitive actions. The task completion times in both Task 1 and Task 2 were significantly shorter in the AI-enabled version, demonstrating the efficiency gains offered by the system. This finding highlights the core advantage of AI: speeding up routine tasks without compromising accuracy.

The statistical significance (p-values of 5.02e-07 for Task 1 and 3.52e-05 for Task 2) further confirms that the improvement was not due to random variation, but rather a consistent enhancement attributable to AI integration.

6.2 User Emotions

The Wilcoxon Signed-Rank Test results in Table 3 show that the AI-enabled version significantly improved positive emotions, such as satisfaction, excitement, confidence, and interest. This indicates that the AI enhanced the user experience by streamlining categorization and reducing errors, making interactions more engaging and efficient.

Negative emotions, including frustration, boredom, and annoyance, were significantly reduced, demonstrating a positive impact on user satisfaction. However, the lack of change in confusion suggests that some users still faced uncertainty with unclear AI predictions, highlighting an area for improvement.

Overall, the AI integration not only optimized task performance but also positively influenced user emotions, creating a more enjoyable and less stressful experience.

6.3 Flexibility Principles in the AI-Enabled App

The usability and flexibility results in Table 5 confirm that the AI-enabled app successfully met its design goals. Each flexibility principle—dialog initiative, multi-threading, task migratability, substitutivity, and customizability—was effectively enhanced by the AI integration.

Dialog Initiative: A high user control score (4.42) demonstrates that users felt empowered to override AI predictions, reinforcing their sense of control in categorization decisions.

Task Migratability: Users valued the flexibility to shift between AI and manual categorization, reflected in a flexibility score of 4.33.

Customizability: The AI's ability to learn and adapt to user habits improved customizability, as shown by the high adaptability score (4.42), leading to more personalized and accurate category predictions.

Multi-threading: Positive feedback shows that real-time AI predictions allowed users to input items without interruption from the model's predictions, confirming the effectiveness of the AI's multi-tasking support.

Substitutivity: The adaptability score (4.42) highlights users' appreciation for the option to switch between AI-driven and manual methods of task completion.

These results collectively demonstrate that the AI-enabled app effectively enhanced the flexibility and usability principles. By enabling users to retain control while benefiting from AI support, the app delivered on its promise to create a user-friendly system that adapted to individual preferences.

6.4 Limitations and Future Improvements

This study had some limitations. With only 12 participants, the sample size was too small to generalize the findings to a wider population of grocery shoppers. A larger, more diverse sample would provide more reliable results. The small sample size also required the use of non-parametric methods, which are less robust with limited data.

Additionally, all users experienced incorrect AI predictions at some point, indicating that while the AI improved over time, its initial accuracy needs enhancement. Clearer explanations of AI decisions would also improve user trust.

Future improvements could include adding advanced input methods, such as voice commands or predictive text, to reduce cognitive load and further enhance flexibility. Larger studies are needed to gain more comprehensive insights into the app's usability across different user groups.

7 Conclusion

In summary, the AI-enabled grocery list app demonstrated clear advantages over the manual version in terms of task completion time, user emotions, and flexibility. The integration of AI not only sped up the task but also enhanced user satisfaction by offering adaptable, context-aware suggestions. Although there are still areas to improve, particularly in terms of making the AI more transparent and explainable, the results suggest that AI can significantly enhance the usability of everyday applications by reducing effort, improving user experience, and maintaining flexibility.

8 Immersive Communication and Explainability

8.1 Exploiting Immersive Communication Technologies

Immersive technologies such as augmented reality (AR) and virtual reality (VR) could significantly enhance the user interface in future versions of the grocery list app. AR could be used to visually guide users within a physical grocery store, directing them to the specific section where an item is located. This would make shopping more efficient by providing real-time directions through their mobile device. For instance, the app could overlay digital arrows or markers onto the real-world store environment, helping users navigate to the correct aisle.

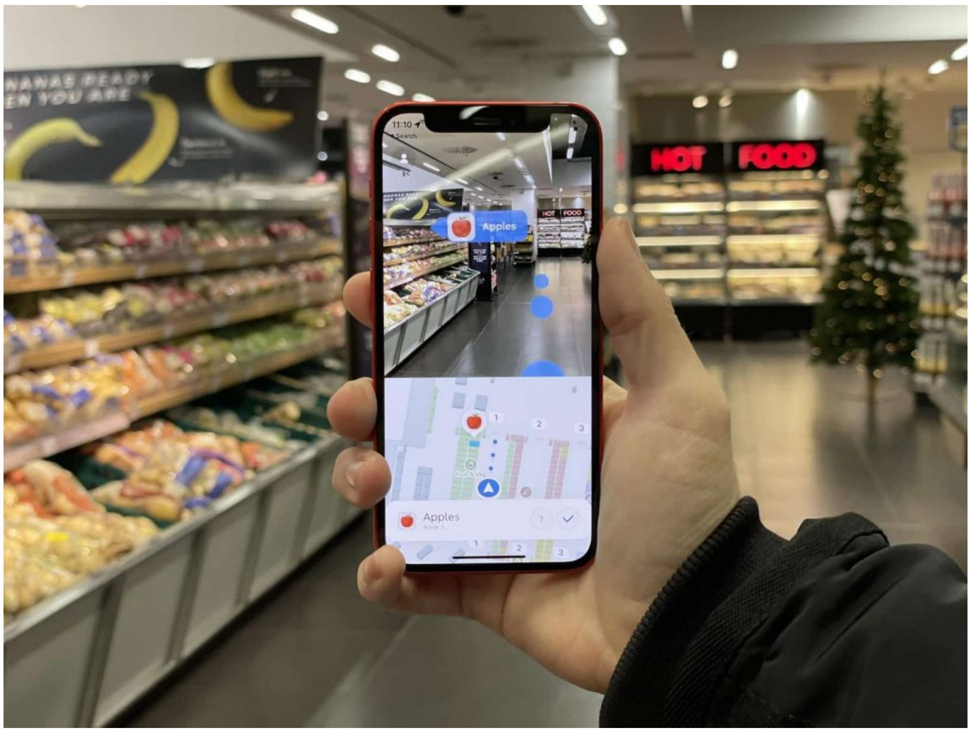


Figure 11: Example of AR guiding users to grocery store sections. Source: `https://poplar.studio/blog/augmented-reality-grocery-shopping-examples/`

AR could also help users organize their lists more intuitively by allowing them to categorize and prioritize items in a visual 3D space. Additionally, in a virtual reality (VR) environment, users could experience a fully immersive grocery store, where they interact with their shopping lists in real-time and categorize items as they ”pick” them from virtual shelves. This would be particularly useful for online shoppers or for planning grocery trips more efficiently.

Immersive technologies could further support multi-user collaboration, allowing family members to share a virtual space and update their grocery lists together. Although not essential for the app's core functionality, AR/VR would enhance user engagement, offering a more interactive and efficient way to manage shopping tasks.

8.2 Explainability in Intelligent Interfaces

Explainability is crucial for building trust in AI-powered systems, especially when users encounter incorrect predictions. In the grocery list app, all users faced some errors, highlighting the need for transparent feedback.

Since the underlying model is a classifier that operates on probabilities, we can provide users with a history section that shows previously added items, along with the list of categories and their associated probabilities. This allows users to check how decisions were made without overwhelming them with information during normal use. Users could access this history to see how their shopping patterns influenced the AI's predictions (e.g., ”This item was categorized as Produce based on a 75% likelihood from previous selections”).

This optional feature ensures that only users who want to understand the AI's reasoning can access the probability details, while keeping the main interface clean and focused on simplicity. Offering this transparency enhances trust and user confidence in the system's accuracy without disrupting the user experience.

A Appendix

A.1 Detailed User Needs and Requirements

The user needs identified from the survey include the following:

- **Quick Item Entry:** Users should be able to add items easily with minimal steps.
- **Automatic Categorization:** Items should be categorized automatically into predefined or user-defined categories.
- **AI-Powered Recommendations:** The app should predict item categories based on previous user input and shopping patterns.
- **Customizable Categories:** Users should be able to create and edit categories to match their preferences.
- **Phone Compatibility:** The app must be optimized for mobile use, as most users shop with their phones.

A.2 Non-Functional Requirements

- **Usability:** The interface should be intuitive and easy to learn for first-time users, while efficient for frequent users.
- **Performance:** The app should load quickly and handle large lists smoothly, with minimal delays during navigation or list creation.
- **AI Responsiveness:** Predictions should be made within 1 second to ensure a seamless user experience.

A.3 Requirement Questionnaire

The requirement questionnaire used for gathering user needs is provided at the following link: Requirement Questionnaire Sheet.

A.4 Manual for the Application

The manual for the application includes detailed instructions on how to use all features of the grocery list app. You can access the manual by following this link: Manual for the Grocery List App.

A.5 User Testing Data and Questionnaires

This section contains all the data collected during user testing, including completed questionnaires and raw data used for analysis. link.

A.6 Training Dataset

The items are distributed between 15 pre-defined categories. The dataset is available on the hugging face datasets library using this link.

A.7 Shapiro-Wilk Test Formula

The Shapiro-Wilk test statistic (W) is calculated using the following formula:

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

Where:

- $x_{(i)}$ are the ordered sample values,
- $\bar{x}$ is the mean of the sample,

- a_i are constants that depend on the sample size n.

The constants a_i are derived from the expected values of order statistics of a standard normal distribution. These constants are used as weights in the formula to ensure sensitivity to deviations from normality.

A.8 Paired T-test Formula

The formula for the paired T-test is as follows:

$$t = \frac{\bar{d}}{\frac{s_d}{\sqrt{n}}}$$

Where:

- $\bar{d}$ is the mean difference between paired observations (AI vs. non-AI task completion times),
- s_d is the standard deviation of the differences,
- n is the number of paired observations.

A.9 AI Training Data and Model Details

A.9.1 Training Data

For the classification task, data was gathered through:

- **User Survey Data**: This provided pairs of items and categories commonly used by participants.
- **Synthetic Data**: Generated using ChatGPT to supplement the user-provided data, cleaned up manually for accuracy.

A.9.2 Initial Model

The logistic regression model used Word2Vec embeddings to map items to vectors, capturing semantic relationships between items for improved predictions.

A.10 Links to Multimedia, Videos, and Supplementary Materials

The provided links can access additional materials such as demonstration videos and supplementary documents. You can find the materials in this Google Drive folder: Link.

Here you can find the google colab notebook that was used for the data analysis with the results in the cells: Link.

A.11 Code Snippets for Data Analysis

The following code snippets were used for analyzing the task completion times and Likert scale data during the user testing.

A.11.1 Normality Testing and Paired T-Test for Task Completion Times

We used the Shapiro-Wilk test to check for normality in the task completion time data. If both rounds were normally distributed, a paired t-test was performed to determine if there was a significant difference between the two rounds.

```python
import scipy.stats as stats

def analyze_completion_times(round_1, round_2):
    # Shapiro-Wilk normality test for both rounds
    shapiro_round_1 = stats.shapiro(round_1)
    shapiro_round_2 = stats.shapiro(round_2)

    print(f"Shapiro-Wilk Test for Round 1: W={shapiro_round_1.statistic}, p-
        value={shapiro_round_1.pvalue}")
    print(f"Shapiro-Wilk Test for Round 2: W={shapiro_round_2.statistic}, p-
        value={shapiro_round_2.pvalue}")

    # Check if both rounds are normally distributed (p-value > 0.05 means
        normal)
    if shapiro_round_1.pvalue > 0.05 and shapiro_round_2.pvalue > 0.05:
        t_test_result = stats.ttest_rel(round_1, round_2)
        print(f"Paired t-test: t-statistic={t_test_result.statistic}, p-value={
            t_test_result.pvalue}")
```

Listing 1: Code for analyzing task completion times

A.11.2 Wilcoxon Signed-Rank Test for Likert Scale Data

For the analysis of Likert scale data, we used the Wilcoxon signed-rank test, which is appropriate for ordinal data.

```python
import scipy.stats as stats

def analyze_likert_data(round_1, round_2):
    wilcoxon_test_result = stats.wilcoxon(round_1, round_2)

    print(f"Wilcoxon signed-rank test: statistic={wilcoxon_test_result.
        statistic}, p-value={wilcoxon_test_result.pvalue}")

    if wilcoxon_test_result.pvalue < 0.05:
        if np.mean(round_1) > np.mean(round_2):
            print("Round 1 values are greater.")
        else:
            print("Round 2 values are greater.")
```

Listing 2: Code for analyzing Likert scale data

A.11.3 About the scipy Library

The code snippets above make use of the `scipy` library, specifically the `scipy.stats` module, which provides a wide range of statistical functions. We used the Shapiro-Wilk test for normality checks and the paired t-test for normally distributed data. For ordinal Likert scale data, the Wilcoxon signed-rank test was applied.

A.12 First Survey Data

The survey data can be accessed here: Survey Data.