CS5041 Practical 2, Group Report: Expo application

Submitted by 190020857 and 220034672.

Summary

This application is a shared shopping list, where users can collaboratively add to the list, update information, tick off purchased items, manage shopping finances, track their five-a-day and meal plan. It was built with Expo, using the React Native Paper component library, and stores data in a Google Firebase Realtime Database. Information about setting up and running the application is in the README file.

Key activities

Our key activities are presented as user stories.

1. As a user, I want to add things to my shopping list so that I do not forget to buy something I need.
2. As a person who lives with others, I want to see what other people have added to my shopping list in real-time, so that I know what the household needs.
3. As a person doing the shopping, I want my list divided into categories, so as I walk around the supermarket, I know what I need in each section.
4. As a user, I want to be able to specify information like quantity or additional notes about the items, so that I remember exactly what I need.
5. As a person doing the shopping, I want to be able to tick off items as I go round, so I know what I've still to find.
6. As a user I want to plan my meals, and then create a shopping list based on that.
7. As a user I want to split the bill for the shopping in the same space where I have my shopping list.
8. As a user I want to keep track of my nutritional intake alongside my shopping list, making it clear whether I am buying enough fruit and vegetables.

Design

The database layout is described in the `schema.json` file with some descriptions about what each field does. Generally, items in the database map very closely to something a user interacts with. The `list-item` `lastBought` field is always transformed before it is shown to a user: if it is blank, the item is un-ticked and still to buy. Otherwise, it contains the timestamp the item was last purchased at, which is used for list sorting. Handling this transformation adds a little more complexity in the code, but also adds a lot of flexibility and helps to keep the database simpler. It also reduces the risk of anomalies, where an item could (for example) be marked as ticked but without a timestamp.

The application opens on the shopping list screen, as that is the area which the user is likely to interact with most frequently and they will want quick access to it when they realise they are running out of some grocery item or when they walk into a shop. The list and all its metadata updates in real-time as changes are made on other devices.

An input is presented at the top of the screen for the user to type in things they need to buy. When the user adds an item using this field, the text they enter is fuzzy-searched (using the Fuse.js library) against a catalogue of built-in known items and everything else they have put on their list to try and guess an appropriate icon and category for the item. If a potential match is found, the item is added with the bonus metadata. Otherwise, the item is added to the Uncategorised section with a generic icon. Items in the Uncategorised section can be re-categorised by the user, which will help the system do a better job next time. This means that a user can enter "Digestive biscuits" and see the right icon and category applied automatically, even though the system doesn't know "digestive biscuits" – it will match it to the more generic term of "biscuits". The prototype project has a relatively small catalogue as a demo (in `app/dev.tsx`) – a full production application would have a larger catalogue.

The list is sorted by category. This keeps similar items together, so it is easier to navigate the supermarket. Un-ticked items are sorted alphabetically; ticked items are sorted by last purchase time so it is easy to find staples you need to buy often (e.g. bread) and un-tick them when you next run out (this workflow is why ticked-off items stay visible on the list). This type of sorting was challenging to implement, since the relevant metadata is stored as a string in Firebase instead of a JSON object due to the security rules on the realtime database, so we cannot use the Firebase filtering tools very much. Instead, we download all the list items and group/sort them on-device with the lodash library.

The list displays the item name, an icon if available, the item notes, the item quantity (editable), and a tick box to mark an item as done. This is a reasonably compact layout which shows all the key information you need as you shop. Tapping on an item opens a detail pane which lets you edit all information about the item, including re-categorising it, seeing when it was last purchased (displayed as a relative timestamp with Day.js), recording the item's price, and deleting it completely. The deletion action has a confirmation step to prevent accidental activation.

We also stick to using `useState` and `useEffect` hooks in the Receipts screen for state management and potential side effects as this aligns with best practices for functional components as this led to cleaner code.

The application supports dark mode and will switch automatically to match the user's device setting.

Analysis

We've leveraged React Native Paper for a cohesive design and Firebase for real-time updates, focusing on front-end user experience. Going forward, deeper integration with React Native Paper will be key to unlocking its full potential.

The app boasts intuitive navigation and is accessible to a diverse user base. To improve, we aim to create a more integrated flow between features like bill splitting and the shopping list for a seamless user experience. Additionally, these tabs are still basic calculations that could add much greater functionality, such as bill splitting by selected users. Furthering our integration, we could have integrated the web such that our application acts as a much greater sorter of information to potential users.

We utilise the Garrett et al (2016) description for simplicity within our design, using simple subject headings with easy to use and understand tabs which reduce the cognitive load to the user. To go beyond a basic design, the meal planner could store set weeks' worth of meals or even individual favourites and each meals component. Additionally, the current array data structure and methods within the meal planner would not scale well.

Future Work

Our vision for the application's evolution encompasses a suite of enhancements designed to enrich user interaction and functionality:

1. **User Account and Grouping**: We plan to introduce a robust account system, enabling users to form groups for collaborative shopping. This system will facilitate item assignment among group members, preventing redundant purchases. Our database already lays the groundwork for this feature.
2. **Intelligent Item Input**: We aim to refine the item input process with predictive text features that suggest familiar items and auto-populate frequent notes, streamlining the list-making experience.
3. **Diverse Iconography**: Recognizing the limitations of current icon sets, we intend to incorporate a specialized food icon library, enhancing visual appeal and user navigation. An initial icon picker prototype highlighted the need for a more tailored approach.
4. **In-Shop Display Customization**: In response to user feedback, we'll implement a toggle feature to manage the visibility of completed items, minimizing in-store distractions and optimizing the shopping flow.
5. **Automate:** The shopping list items, and nutrition tracker based on meal plans can be automated to make the user perform less activities for the same value.
6. **Gamification and Persuasive Design**: We're exploring gamification strategies, such as completion incentives and personalized UX settings, to engage users and foster a sense of achievement and ownership as gamification increases user engagement (Loovyestn et al 2017).
7. **Conversational Agents**: The integration of AI-driven conversational agents will offer a more interactive and responsive shopping experience, capable of understanding and assisting with natural language queries. Chatbots have also been shown to increase upselling and marketing (Lee 2020)
8. **Iterative Design and Feedback Loop**: We are dedicated to an iterative development process, continuously incorporating user feedback to refine and evolve the application's design and functionality.
9. **Prototype and User Testing**: Future phases will involve developing a working prototype and conducting comprehensive user testing to gather actionable insights, driving informed design enhancements.

These planned features and improvements are aimed at delivering a more personalized, efficient, and enjoyable shopping experience, ultimately driving user satisfaction and engagement.

Conclusion

This project fully meets the project specification including some advanced requirements. The application supports 8 key activities including real-time collaboration features and the ability to calculate key metrics important to food shopping such as bill splitting, and a 5-a-day counter, as well as meal planning. The application supports a variety of content types and uses additional libraries to enhance the user experience with features like advanced sorting.

References

Garett, R., Chiu, J., Zhang, L. and Young, S.D., 2016. A literature review: website design and user engagement. Online journal of communication and media technologies, 6(3), p.1.

Lee, S.B., 2020. Chatbots and communication: the growing role of artificial intelligence in addressing and shaping customer needs. Business Communication Research and Practice, 3(2), pp.103-111.

Looyestyn, J., Kernot, J., Boshoff, K., Ryan, J., Edney, S. and Maher, C., 2017. Does gamification increase engagement with online programs? A systematic review. PloS one, 12(3), p.e0173403.

Individual Reflection - 220034672

Throughout the development of our Expo application for CS5041 Practical 2, my role was multifaceted, involving initial concept development, literature review, data schema design, and front-end implementation across the shopping list, bill splitter, nutrition tracker, and meal planner.
Our initial sessions conceptualized our application, then the next task was to conduct a literature review to inform our design choices, ensuring our application was grounded in user-centric principles - though I was perhaps too optimistic here. Concurrently, I was responsible for designing the data schema, only to find Andrew had done this too.
I pivoted to focus on the front-end, particularly implementing our agreed accordion list feature within our shopping list interface, which was key to a smooth user experience and ease of use. Despite successfully integrating this with real-time database updates, a unilateral change by my partner later in the project discarded much of my work and necessitated a hasty redesign. This limited the depth of functionality I ended up providing.
On a personal level, this project has honed my adaptability and resilience in the face of unforeseen changes. It has reinforced the value of documentation and regular check-ins within team projects. Most importantly, I have gained a deeper understanding of the Expo framework, Firebase integration, and the nuances of collaborative software development such that web development can be much easier and smoother in the future.
In conclusion, while the journey was fraught with challenges, the knowledge and skills I have acquired are invaluable. I look forward to applying these lessons to future collaborative endeavors and believe the end result was not as bad as the journey there.