

SMART REAL TIME BRING LIST APP

**Submitted in partial fulfillment of the requirements for the award
of Bachelor of Engineering degree in
Electronics and Communication Engineering**

By

ZOYA FIRDOUSI

(Reg no 38130248)



**SCHOOL OF ELECTRICAL AND ELECTRONICS ENGINEERING
DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI
CHENNAI – 600119**

MARCH - 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of
ZOYA FIRDOUSI (38130248) who carried out the Project entitled
"SMART REAL TIME BRING LIST APP". Under
the guidance of **DR. A. ARANGANATHAN M.TECH PhD Associate professor**,
during the period NOV 2020 to APRIL 2021

INTERNAL GUIDE

DR. A. ARANGANATHAN,
Mtech PhD Associate Prof.

HEAD OF THE DEPARTMENT

Dr.T.RAVI ,M.E.,Ph.D

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **ZOYA FIRDOUSI (38130248)** hereby declare that the Project report entitled **SMART REAL TIME BRING LIST APP** is done under the guidance of **DR. A.**

ARANGANATHAN Mtech PhD Associate Prof., is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **ELECTRONICS AND COMMUNICATION ENGINEERING**.

DATE:

PLACE:

SIGNATURE OF CANDIDATE.

(ZOYA FIRDOUSI 38130248)

ACKNOWLEDGEMENT

I pleased to acknowledge my sincere thanks to board management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY for their kind encouragement in doing this internship and for completing it successfully. I am grateful to them.

I convey my thanks to Dr.N.M.NANDHITHA,M.E.,Ph.D., Dean, school of Electrical and Electronics engineering and Dr.T.RAVI,M.E.,Ph.D., Head of the Department, Electronics and Communication engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide DR. A. ARANGANATHAN Mtech PhD Associate Prof., for his valuable guidance, suggestions and constant encouragement paved the way for the successful completion of my Project.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Electronics and Communication Engineering who were helpful in many ways for the completion of the project.

ABSTRACT

In this report, I present a Hybrid App (iOS and Android platform) BringLixt that focuses on bringing items that are needed or shopped cooperatively within family members. Previously some work has been done on a native app for making shopping lists that influences healthy shopping habits and a useful smartphone solution for the process of making food lists. I have improvised these apps and built an app that manages item lists within family members which are going to be bought by a family member. I have improvised it by making a Hybrid App that can be installed in both iOS and Android both smartphones, no need to build separate apps for iOS and Android. I have also created a very user friendly interface so users can use it in a very easy manner in less time as compared to other apps. I have used Flutter technology to achieve this goal which helps us to build this Hybrid App.

TABLE OF CONTENT

CHAPTER	TITLE	PAGE.NO
	ABSTRACT	iv
	LIST OF FIGURES	vii
	LIST OF TABLES	ix
1	INTRODUCTION	9
1.1	ORIGIN OF IDEA	10
1.2	OBJECTIVES AND SCOPE	11
1.3	ORGANIZATION	11
2	LITERATURE SURVEY	13
2.1	A COLLABORATIVE SHOPPING LIST ANDROID APP	13
2.1.1	METHODOLOGY	14
2.1.2	IMPLEMENTATION	15
2.2	A COLLABORATIVE SHOPPING LIST ANDROID APP	15
2.3	LIST IT: A SHOPPING LIST APP	17
2.4	IMPROVING FOOD SHOPPING BEHAVIOR	18
2.5	CROLIST: A MOBILE APP FOR COLLABORATIVE GROCERY LIST MANAGEMENT	20
3	OVERALL APPROACH	24
3.1	PROPOSED METHOD	24
3.1.1	AUTHENTICATION	24
3.1.2	LIST OF USERS	25
3.1.3	LIST OF ITEMS	25
3.2	IMPLEMENTATION	26
3.3	DESIGN	34
3.4	SCOPE	37
4	TOOLS AND TECHNOLOGY USED	38
4.1	VISUAL STUDIO CODE	38
4.2	XCODE 13	38
4.3	ANDROID STUDIO (IDE)	39
4.4	iOS SIMULATOR	40

4.5	ANDROID STUDIO EMULATOR	41
4.6	TECHNOLOGY USED	42
5	RESULT	49
6	CONCLUSION	51
7	REFERENCES	53

LIST OF FIGURES

FIGURE.NO	TITLE	PAGE. NO
3.2.1	Source code for signup process	28
3.2.2	Firebase Authentication using email and password	29
3.2.3	Source code for Sign in process	30
3.2.4	Source code for collecting current user info and connected user list	31
3.2.5	Source code for collecting other user info	32
3.2.6	Source code for logout	32
3.2.7	source code for collecting all Items from database	33
3.2.8	Source code for adding new item into Bring list	33
3.2.9	Source code for updating Bring List table in Database	34
3.2.10	Source code for adding new item into Item list table	34
3.2.11	Source code for deleting an item from Bring list	34
3.3.1	Users table with required attributes	35
3.3.2	bring_list table with required attributes	35
3.3.3	Items table with required attributes	36
5.1	Results with user journey	49-50

CHAPTER 1

1 INTRODUCTION

In the ubiquitous computer environment, technology such as mobile phones have great potential to ease human everyday living activities toward a healthier, easier, and more pleasant existence. People's lifestyles have become more reliant on online buying. A Bring List app is an essential means of bringing items. Every family necessitate an online listing app for essential items that should be purchased from the market by a family member, and the rest of the family members can add items to same list such that the one member of the family who is in the market can easily purchase all of the items without forgetting any of the members of the family who have added items to the list.

In today's pervasive computer environment, technology such as mobile phones has enormous potential to alleviate human daily living activities toward a healthier, easier, and more pleasant creation. People's lives have become more reliant on online purchases. A Bring List app is a must-have for bringing items. Every family requires an online listing app for essential items that a family member should purchase from the market, and the rest of the family members can add items to the same list so that the one member of the family who is in the market can easily purchase all of the items without forgetting any of the members of the family who have added items to the list.

The users can connect with each other by installing the apps on their mobile devices, and can simultaneously operate on the lists by sharing the lists with each other. Also the app will be connected to a backend service- Firebase, which will be used for storage as well as user authentication at the same time.

Devices can subscribe with one another by installing the apps on their mobile devices and operating on the lists at the same time by sharing the lists. In addition, the app will be linked to a backend service called Firebase, which will be used for both storage and identity management.

In every daily life there are so many essential items like vegetables, fruits, groceries and street food etc that are needed by members of a house and in those

members of a house some members require these items and few members who are buying those items from the market. A person can have so many required items and he/she could forget some items after some time. A person has to buy so many items from the market and he/she could also forget some items or might be confused which items are remaining to buy and which items are already purchased.

In life, there are numerous essential items such as vegetables, fruits, groceries, and street food that are required by members of a household, and among those members of a household, some members require these items while others purchase them from the market. A person may have a large number of required items, but he or she may forget some of them after some time. A person must purchase a large number of items from the market, and he or she may forget some items or become confused as to which items remain to be purchased and which items have now been bought.

1.1 ORIGIN OF IDEA

We prefer to contact or text someone if we need to request that they bring a list of items such as vegetables, fruits, and all daily usage goods. With the bringlixt app, we can immediately click on the thing we want from the list that is incorporated into the app, which will alert the person from whom we want the item. This real-time application encourages folks who do not know how to write or speak by allowing them to inform what product or object they want with a single click on the app.

We tend to call or text someone if we need to request that they bring a list of items such as vegetables, fruits, and everyday necessities. With the bringlixt app, we can immediately click on the item we want from the app's list, which alerts the person from whom we want the item. This real-time app encourages people who do not know how to write or speak by allowing them to tell the app what product or object they want with a single tap.

BringLixt App provides a better and easy solution for these problems by giving a very user friendly interface to manage a list of bringing items . A common app for all the members of a house so that every member can add items in the list and manage and

cross check items which have been purchased or to be purchased from the market. BringLixt App provides a promising and easier solution to these challenges by providing a very user pleasant interface for maintaining a list of bringing products. A shared app for all members of a household, allowing each member to add goods to the list and monitor and cross-check products acquired or to be bought from the shop.

1.2 OBJECTIVES AND SCOPES

It is human nature to forget things from time to time. If this occurs, other parties will be forced to waste time and wait. This real-time application motivates those who do not know how to write or talk with this they may notify what product or thing they desire with a single click on the app.

It's natural instinct to forget things now and again. Other parties will be compelled to lose time and make sure if this occurs. This real-time application encourages folks who do not know how to write or speak by allowing them to inform what product or object they want from a single tap on the app.

The application will provide a user-friendly interface for users to shop for various things, add people, and share lists with them. The application's real-time management will allow the user and family member to monitor each other's activity while purchasing various things in a convenient manner. Moreover, the software will save time while purchasing multiple items.

Users will be able to shop for various items, add others, and share lists with them using the application's user-friendly interface. The application's real-time administration will allow the user and family member to keep track of each other's activities while shopping for various items in a simple manner. Furthermore, the programme will save time while buying many things.

1.3 ORGANIZATION

A Collaborative Bring List Hybrid (Android and IOS) App. The Application will give a user-friendly interface for users to list various things, add people, and share lists with them. After shopping for a specific item is done, a delete button will be applied to remove that item from the list. With a click, an item is removed from the list, assisting

the user in avoiding confusion for the remaining items.

A Hybrid Bring List (Android and iOS) App for Collaborative Bring Lists. The application will provide a simple interface for users to list various items, add individuals, and share lists with them. After you've finished shopping for a certain item, a delete button will appear to remove it from the list. An item is eliminated from the list with a click, facilitating the user in preventing complications for the rest of the items.

"Bringlist" is an hybrid application that focuses on bringing item that be needed or shopped on cooperatively and jointly by a group of users. The application uses the Firebase Real-time Database for listing items and Authentication services for user login sign up. The data is saved as a json format on the Firebase Console, which is accessed using the user account.

CHAPTER 2

2 LITERATURE SURVEY

2.1 A COLLABORATIVE SHOPPING LIST ANDROID APP

The Internet is the fastest growing media during the past decade. Online shopping is a rapidly growing e-commerce area. Online shopping has become a major part of people's lives. A core component of e-commerce is having a shopping cart. To establish an e-commerce website, every online entrepreneur needs to have an online shopping cart. According to the records there are approximately 300 million smartphone users.

During the last decade, the Internet has been the fastest expanding medium. Online shopping is a fast expanding e-commerce industry. People's lifestyles have become more reliant on online buying. A shopping cart is an essential component of e-commerce. Every online entrepreneur needs an online shopping cart in order to set up an e-commerce website. There are around 300 million smartphone users, according to data.

The Internet has been the fastest growing media during the previous decade. Online shopping is a rapidly growing segment of the e-commerce business. People's lives have become more reliant on internet purchases. A shopping cart is an integral component of online purchasing. To set up an e-commerce website, every online entrepreneur needs an online shopping cart. According to estimates, there are around 300 million smartphone users.

Our application "Carts" is an advanced shopping cart system, in which a user can shop along with their friends and family in a real-time environment simultaneously. The users can connect with each other by installing the apps on their mobile devices, and can simultaneously operate on the lists by sharing the lists with each other. Also the app will be connected to a backend service-Firebase, which will be used for storage as well as user authentication at the same time.

Our programme "Carts" is a sophisticated shopping cart system that allows

users to shop alongside their friends and family in a real-time setting. Users can connect with one another by installing the applications on their mobile devices and operating on the lists at the same time by sharing the lists. In addition, the app will be linked to a backend service called Firebase, which will be utilized for both storage and user authentication.

2.1.1 METHODOLOGY

"Carts" is an android application that basically focuses on the shopping carts that are collaboratively and collectively worked or shopped on by a group of users. The Application uses Firebase Real time Database and Authentication services. The data is stored as a hashmap on the Firebase Console that is logged onto via the account created on Google. In order to be a user ,the person needs to be registered on the app via email/password registration that is maintained by firebase API or the google Accounts system. Once you are an authenticated user or member of this application you are given two tabs one of which is the shopping ,the other being the meals part .

On the main page via a floating action button at the bottom right of the page a new list could be created simply by giving it a name. Once the list is created list items can be created easily by giving them a name , items on being created are displayed in the list with a delete option on the right side.List menu contains a green button which serves as an indicator whether the person logged into the app is currently shopping in that list or not .Just below the name there are currently active shoppers shown in the Lists Screen.

On the main screen there is a way to sort the given lists by order time , alphabetically , or by publish time. The list screen also contains a feature such that when an item has been bought , the item name gets a strikethrough and moves that item to the bottom of the list separating the bought and unbought items .

Another feature brings a sharing list option among the friends that are already registered on the application . To share a list use the menu in the List Screen to get to the Sharing screen in which on entering your friends email , you

can share the list with your friend who gets notified about the list you share.

2.1.2 IMPLEMENTATION

The application consists of the following Wireframes:

- Version 0.1 – This is where I will decide what kind of stack we are going to use. This version will be responsible for writing and reading one piece of data from the online data storage that I just set up.
- Version 0.2 – Consists of other screens such as the all shopping list screens and shopping list details screens to get the data that is stored in the database and decide how to visualize it to connect to the database.
- Version 0.3 – Consists of the signin and creates an account screen to the application that will allow for authentication and user accounts.
- Version 0.4 – we could cover how to use all the user data that I just grabbed to allow for sharing user lists. Also , since we are accumulating all the user data, I would probably want some way to sort and filter it.
- Version 0.5 – To address security and database permissions.

2.2 A COLLABORATIVE SHOPPING LIST ANDROID APP

Software solutions based on mobile platforms have become very popular in today's technology world on a massive scale and are used in a vivid range of world domains. Not only mobile applications have become a lifestyle but also the technology related is evolving by the day and demonstrating the promising potential to undertake most of the human technological necessities in a comprehensive manner. Similarly, data mining is also a broad concept which is employed in almost every field in the world, heavily in data sciences. Whilst applying the practicality of data mining on sales and consumer data is prevalent, in identifying purchase and interest patterns, association and association rules, is widely taken into consideration.

Software solutions based on mobile platforms have grown extremely popular in today's technological world and are employed in a diverse variety of world disciplines. Not only have mobile applications become a way of life, but technology

in general is advancing at a rapid pace and displaying the promising ability to meet the majority of human technological needs in a complete manner. Similarly, data mining is a wide notion that is used in practically every industry throughout the world, particularly in data sciences. While the usefulness of data mining on sales and consumer data is common, association and association rules are commonly used in determining purchase and interest trends.

Although a substantial amount of applications have been produced in various mobile platforms for online grocery shopping and comprehensively data mining techniques have been applied on sales data for interest mining, the requirement has always been present for a solution to enhance static grocery shopping experience of people. Up to the present time people use traditional ways of carrying out these activities such as writing on a piece of paper or memorizing items to be bought which are not reliable and eventually leads to time and money loss. Coming up with a solution which has a capacity of properly addressing the above issues would be much beneficial for the users in terms of both financially and lifestyle improvement.

Although a significant number of applications for online grocery shopping have been developed for various mobile platforms, and extensive data mining techniques have been applied on sales data for interest mining, the need for a solution to improve people's static grocery shopping experience has always existed. People still utilize old methods of carrying out these actions, such as writing on a piece of paper or memorizing goods to be purchased, which are unreliable and result in time and money waste. Developing a system capable of appropriately resolving the aforementioned difficulties would be extremely advantageous to consumers in terms of both financial and lifestyle improvement.

This research presents our work on a mobile-assisted software application namely 'The Smart Shopping List' for enabling the users to enrich their grocery shopping experience to a new level. It allows the users to create shopping lists, find the ideal shop using geolocation services, receive item recommendations filtered using pattern matching recognition with higher level of accuracy using association algorithms, and share shopping lists via text-to-app service. The

experimental runs and the comparisons with Weka software have shown promising results in recommending items to the user. Section discusses human grocery shopping behaviors while section III and IV provides an overview of related work followed by sales datamining. Section presents the Smart Shopping List design and the implementation while section describes experiments for evaluating its algorithmic accuracy and performance, and section concludes the paper.

2.3 LIST IT: A SHOPPING LIST APP

Over two billion adults and children worldwide are overweight or obese [3]. Many of these people suffer from serious diseases such as diabetes and heart-related diseases such as Ischaemic heart disease, which has been identified by the World Health Organisation as the number one cause of death worldwide¹. Living an active life and eating healthy foods have been identified as the two major ways of improving one's health, reducing weight and preventing weight gain [2]. The last few years have seen an increase in research promoting active lifestyles such as the development of games that influence people to be more active [9],[5] and the development of fitness apps and devices such as the Fitbit² that encourages people to move around and exercise more. However, more can be done in the area of persuading people to eat healthy, in particular, influencing consumers to shop for healthier foods in order that they will eat healthier foods. I hypothesize that if consumers are influenced to shop for healthy foods, they will likely eat healthy because people typically cook and eat what they buy or shop for. Thus, shopping for healthier foods could aid people in improving their health.

One way consumers can be influenced to shop for healthy foods is by using a grocery list. The use of a grocery list has been identified as a useful tool in improving one's diet [4], [6] and decreasing one's body mass index especially for low-income households [4]. Dubowitz et al. [4] suggest that shopping with a grocery list may limit impulsive buying and could act as a "shield against the availability of unhealthy foods".

Using a shopping list is one technique to convince people to buy healthier items. The usage of a grocery list has been recognised as a valuable strategy for

improving one's diet [4, 6] and lowering one's body mass index, particularly for low-income families [4]. According to Dubowitz et al. [4], shopping with a grocery list may prevent impulsive purchases and function as a "shield against the availability of harmful items."

Writing a shopping list is one thing, but actually taking it to the store or not losing it is another challenge. Because people will likely take their mobile phones with them everywhere they go, in this position paper, I present the idea of a shopping list, List It, in the form of a mobile application. With List It, shoppers do not need to type out the items to be included in their shopping list, instead, they can simply select the products they want to include from a list of products. To further persuade consumers to shop for healthy foods, the calorie content of each product is visible in the shopping list app. Furthermore, consumers can add friends to their network and view the weekly statistics of the healthiness of their shopping lists compared to that of their friends.

Making a shopping list is one thing; actually carrying it to the store and not losing it is another. Because individuals are likely to carry their mobile phones with them everywhere they go, we offer the concept of a shopping list, List It, in the form of a mobile application in this position paper. List It eliminates the need for buyers to type out the goods they wish to include in their shopping list, instead allowing them to just choose the products they want to add from a list of products. The calorie amount of each product is shown on the shopping list app to further motivate people to buy healthy meals.

2.4 IMPROVING FOOD SHOPPING BEHAVIOR

Grocery shopping is an essential part of living a healthy and sustainable lifestyle. The interplay between individual-level and environmental factors determines food choices and eating habits (Brug, 2008). For this reason, not only the (mal-)adaptive behavior but also the context in which the behavior occurs needs to be taken into account to develop evidence-based health behavior change interventions (Davis, Campbell, Hildon, Hobbs, & Michie, 2015).

There is evidence that grocery shopping situations in particular can be very

stressful and demanding for many, because of countless temptations and distractors (Aylott & Mitchell, 1999). Countering these temptations requires intensive planning to overcome unwanted behavioural habits. Supporting people in adopting behavior with a healthier outcome, has proven to be challenging. While incentives make people act in a more desired way as long as the incentive is given (Abrahamse, Steg, Vlek, & Rothengatter, 2005), motivating people to change their behavior also in the long term remains an unresolved issue across domains including dieting, energy efficiency, or reduction of waste to name a few.

Personalized approaches and tools like smartphone applications constitute an increasing contribution to health interventions (Metz et al., 2000; Lowe, 2003). The WHO has coined the term mHealth since 2009 (Codyre, 2014) to address the growing health challenges in the world. Smart phones have gained a lot of attention recently through their almost endless possibilities and the fact that they have become so ubiquitous and pervasive (Spruijt-Metz & Nilsen, 2014).

The uses of mobile devices have penetrated almost every aspect of our lives. There is, therefore, significant potential for both monitoring and influencing human behavior utilizing this technology. Nevertheless, neither health interventions (Davis et al., 2015) nor smartphone applications (Riley et al., 2011) are commonly designed with reference to theoretical concepts, or use only isolated components of a behavior theory. While behavioral change theories predominantly focus on interventions and their outcomes (e.g. (Davis et al., 2015; Head, Noar, Iannarino, & Harrington, 2013)), practical considerations for designing smartphone applications have led to a field of user experience (UX) research, which aims at analyzing and improving usability and effectiveness of products and systems (ISO 9241, 2019).

Both perspectives have been developed in parallel but thus far little synergies between the approaches have been exploited. UX evaluation methods can be broadly categorized into those that have to be integrated into the application development, for instance Visitor Behavior Analysis (Wingify, 2021), and those that can be applied without having access to the application back-end. As an example of the latter Heuristic Evaluation (Wilson, 2014) has been proposed as a methodology for analyzing application features with respect to achieving

certain predefined desirable properties.

It is based on defining suitable heuristics (Quinones, Rusu, & Rusu, 2018) and evaluating application features with respect to support of these. Various heuristics have been proposed (Schohn, Thoma-schewski, & Bader, 2017), a popular choice being suggested by Nielsen (Nielsen, 1994; Nielsen Norman Group, 2021), although these are often adapted and augmented as required (e.g. (Reis, Pa'ris, & Gomes, 2020; Suzianti, Minanga, & Fitriani, 2017)). A shortcoming of these approaches is that the definition of heuristics is ad-hoc and is lacking a theoretical underpinning, which behavioral theory has the potential to provide.

In this paper I aim to address this shortcoming by presenting a behavioral model derived from established psychological and social behavioral theories that provides guidelines for defining suitable heuristics to analyze application features accordingly. The presented approach is applied to the most popular recipe and grocery shopping applications available on the Google Play store and an evaluation with respect to heuristics support in a model-derived context is presented. Putting application features into a theoretical context allows for a better understanding of the ways how these features individually contribute to achieving the goal of behavior change towards healthy and sustainable grocery shopping.

We want to overcome this deficiency in this research by offering a behavioral model drawn from recognised psychological and social behavioral theories that gives instructions for creating appropriate heuristics to examine application aspects appropriately. The technique given here is applied to the most popular recipe and grocery shopping apps accessible on the Google Play store, and an evaluation of heuristics support in a model-derived context is presented. Placing application features in a theoretical framework gives for a better understanding of how these elements work together to achieve the objective of behavior change toward healthy and sustainable grocery purchasing.

2.5 CROLIST: A MOBILE APP FOR COLLABORATIVE GROCERY LIST MANAGEMENT

In the ubiquitous computing environment, technologies such as mobile phones have high potential to facilitate human daily life activities towards a

healthier, easier, and more comfortable life [1],[2]. Smartphones are referred to as the modern mobile phones that provide basic communication functions for voice calls and text messaging as well advanced functions such as music player, video player, digital camera, mini personal computer, and navigation tool [3]. Many applications (apps) have been developed to facilitate human daily life activities [4], such as food ingredients recommendation that suggests that people with recipe lists instantly [5]. Further, a grocery list is automatically generated from the recipes [6], which helps people during the grocery shopping process at a grocery store. Many consider grocery shopping a mundane and less exciting task; however, it is an important activity that is a necessary part of human life.

. Shopping for groceries is a routine that may take place daily, weekly, or monthly. The previous study suggests that there are three phases associated with a grocery shopping list, known as a development stage, a fulfillment (or usage) stage, and an outcome (or post-purchase) stage].

The development stage involves the process of generating a grocery list that contains a list of items that need to be purchased by people, where this stage determines how successful a grocery shopping process is accomplished. Grocery lists act as a memory aid during grocery shopping, and a guide to avoid impulse purchases. Further, it also plays a role as a planning tool that manages household planning for meals as well as home expenditure on groceries and foods

A grocery list is also considered as a supporting tool for a grocery shopping process. It can be created by writing the items on a piece of paper or mobile phone's text-processing apps or simply remembering the items in mind [8]. The common practice of generating a grocery list is by writing the items to be bought on a piece of paper. Although the method has been used for ages; nevertheless, a paper grocery list could be easily contaminated by liquids and dirt, difficult in reading handwriting, and may be left at home or even misplaced.

The development in smartphones and mobile technology has been a solution to overcome the limitations in paper-based grocery lists where mobile apps can support the process of creating and managing grocery lists more

conveniently and flexibly. It is the most relevant solution nowadays as smartphones serve many purposes, and they are increasingly becoming an integral part of human lives.

Mobile-assisted grocery shopping has been an important area in consumer behavior studies, which leads to the development of hybrid, intelligent, and multimodal digital shopping lists that are intended to support consumers' shopping processes [7]. The digital grocery lists are created instantly and stored in individual users' smartphones, in which they can be easily retrieved back for updating the items or as a reference during the grocery shopping process. Although the digital grocery lists align with the current human lifestyle and technological trend, they are unable to support a collaborative environment.

The current mobile apps for creating digital grocery lists do not provide the facility to share a grocery list among household members. The lists are not able to be updated instantly by other household members who generally consume grocery products in the house and wish to assist in creating a grocery list. It is the main limitation of the existing mobile apps for creating and maintaining grocery lists in which it is unable to support household members' needs in managing a shared grocery list together using their own smartphones.

The grocery items in a grocery list may range from cleaning products, pet supplies, meat, dairy products, fruits, vegetables, and personal care products. The products may be consumed by family members (such as the daughters and sons) other than the one who prepared the list (usually the mother). The creator of a grocery list must check the grocery inventory and the current state of grocery usage during the process of generating a list. This process is tedious; hence, it may cause the grocery items to be missed and omitted from the list.

However, the family members who directly consume a product could add it to a shared grocery list when it runs out, which makes the process of creating the grocery list more effective, faster, and easier. Managing a shared grocery list between the members of the family is a missing feature in the mobile app for creating a grocery list. Therefore, this paper aims to design and develop a mobile

app that can facilitate household members in creating and managing shared grocery lists. This study proposes a collaborative feature in a mobile app for creating a grocery list that is expected to enhance the overall process and experience of creating a grocery list.

The next section discusses the background and related studies on grocery lists. Then, in Section 3, the methodology for designing and developing the mobile app is described. Section 4 explains the detailed design and development of a mobile app named CROLIST to support collaborative grocery management. After that, CROLIST has undergone a usability evaluation, as reported in Section 5. In Section 6, the work is concluded, and potential future works are listed.

CHAPTER 3

3 OVERALL APPROACH

3.1 PROPOSED METHOD

In the previous paper they made an android app using Android studio and Firebase. I have created a Hybrid app (iOS and Android) using Visual Studio Code, Android Studio and Firebase mainly.

3.1.1 AUTHENTICATION

First I have created an auth screen (Login & Signup) to manage users in our app so that we can differentiate among users. If a user is using our app for the first time then he/she has to register in the app. In the registration process we ask for name, email and password from the user. After entering name, email and password it has used Firebase Auth to register a new user and simultaneously we are adding a new user into Firebase Realtime Database to manage the user list in our app.

Initially, I established an authentication page (Login & Signup) to handle users in our app so that we can distinguish between them. If a user is using our app for the first time, he or she must enroll. We request the user's name, email address, and password during the enrollment process. After providing a user's name, email, and password, we utilized Firebase Auth to register a new user while also adding a new user to the Firebase Realtime Database to manage the user list in our app.

Once a user has completed the registration process he/she is able to Login in our app using Firebase Auth by entering the email and password given in the registration process. After login the user will land on the dashboard in which the main action will come. Once a user is logged in then he/she does not have to login again in our app until the user clears app data from device settings or user changes their device. In the dashboard I have given an option menu button 'Logout' and using this button the user can logout from our app using Firebase Auth.

After successful registration, a user may log in to our app using Firebase Auth by using the email and password provided during the signup process. After logging in, the user will be sent to the dashboard, where the primary activity will take place. Once a user has checked in, he or she does not need to log in again until the user clears app data from device settings or switches their device. I have provided a 'Logout' option menu button on the dashboard, and the user may logout from our app utilizing Firebase Auth by clicking on this button.

3.1.2 LIST OF USERS

After login the user will land on the home screen in which he/she will be able to see the user's list. In this list a user can add or remove another user. user can be added from all users list in Firebase RealTime Database and can be removed only from a users person list. This user list contains those users who have to purchase items for you or who require some items to be purchased from you.

In other words, after registering in, the user will be sent to the home page, where he or she will be able to access the user's list. A user can add or remove another user from this list. In Firebase RealTime Database, users may be added from the entire user list and removed solely from a specific user's person list. This user list includes users who must purchase products for you or who must purchase things from you.

3.1.3 LIST OF ITEMS

When a user clicks on any other user card from the users list then he/she will land on a new screen in which We are showing a list of items to be purchased from the market. I named it the bring list. I have created a bring list table in Firebase Realtime Database and whenever users land on this screen we are showing items from the bring list table. A user can add items into this list by clicking on the add button at the bottom of the screen. When the user clicks on the add button then we are shown a bottom sheet in which a list of all items will appear on the screen from the item list table. A User has to select items from those items to add a new item into the bring list. We have also given a search bar where users

can search for items from all the items present in the items list. If a user searches for an item and the item is not present in the list then we are shown an add item button in the search bar. When the user clicks on add item then I have added item to bring list as well as item list of Firebase Realtime Database.

When a visitor clicks on another user profile in the users list, he or she will be sent to a new page where I will display a list of things available for purchase from the market. It was given the moniker "bring list." In Firebase Realtime Database, I developed a bring list table, and once users arrive on this screen, I display things from the bring list table. By pressing the add button at the bottom of the screen, a user can add things to this list. When the user hits the add button, we are sent to a bottom sheet where a list of all things from the item list table appears on the screen.

To add a new item to the bring list, a user must pick things from those items. I've also included a search box so users may look for goods from the entire items list. If a user searches for an item and it is not found in the list, we are provided with an add item button in the search bar. When the user clicks on an add item, we add it to the brought list as well as the Firebase Realtime Database item list.

3.2 IMPLEMENTATION

I have created an authentication system in which we build the Login and Signup process using Firebase Auth. I have integrated Firebase into our project and we have used two firebase services Firebase Auth and Firebase Realtime Database. Firebase Auth provides an authentication system in which users can login and signup. Firebase Realtime Database provides a database system in which all the data management like create a new table or read data from existing table or update data into existing table or delete data from existing table.

I developed a verification system in which the Login and Signup processes are built with Firebase Auth. I used Firebase Auth and Firebase Realtime Database in our app, and we incorporated Firebase into it. Firebase Auth is an authentication system that allows users to login and sign up. Firebase Realtime Database provides a database system for all data management tasks such as

creating a new table, reading data from an existing table, updating information in an existing table, and deleting data from an old table.

I have used the email password login method in Firebase Auth. In the registration process, we collect information from users like name, email and password and using Firebase auth we have called a signup method to create a new user and side by side we have also added user information in users list from Firebase RealTime Database. After registration the user will land on the home screen.

In Firebase Auth, we utilized the email password login approach. In the registration process, we gather information from users such as name, email, and password, and we utilize Firebase auth to execute a signup function to create a new user while simultaneously adding user information to the users list from the Firebase RealTime Database. The user will be sent to the home screen after authenticating.

```

25 Future<void> signUp() async {
26     if (_passwordController.text != _confirmPasswordController.text) {
27         _showSnackBar();
28         return;
29     }
30
31     var name = _nameController.text.trim();
32     var _email = _emailController.text.trim();
33     var _password = _passwordController.text;
34     final User? user = (await _auth.createUserWithEmailAndPassword(
35         email: _email,
36         password: _password,
37     ))
38         .user;
39     if (user != null) {
40         setState(() {
41             _success = true;
42             _userEmail = user.email;
43             _addUser(Users(name, user.uid, _userEmail!, ''));
44             Navigator.of(context).popAndPushNamed(Hompage.routeName);
45         });
46     } else {
47         setState(() {
48             _success = false;
49         });
50     }
51 }
52
53 void _addUser(Users user) {
54     _userDbRef
55         .child(user.id.trim())
56         .set({'email': user.email, 'id': user.id, 'name': user.name});
57 }

```

Fig 3.2.1: Source code for signup process

For login users have to provide email and password and using Firebase auth we have called a login method that will return a User object of user email, and user id. If the user is not present in the Firebase Auth list then the login method will return null. After login we get user information from the user list table using his id then the user will land on the home screen.

Users must supply an email address and a password to login, and utilizing

Firebase auth, we have called a login function that returns a User object including the user's email address and user id. The login method will return null if the user is not found in the Firebase Auth list. After logging in, we retrieve user information from the user list database using his id, and the user is sent to the home screen.

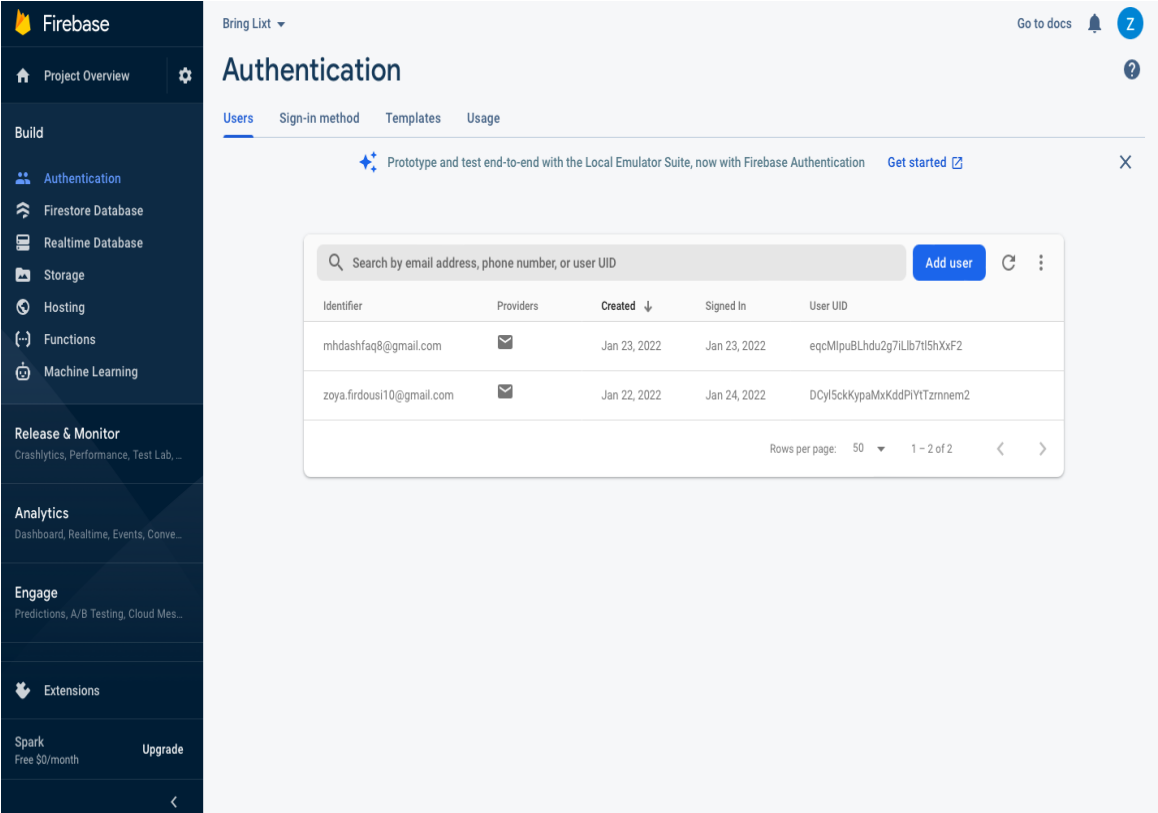


Fig 3.2.2: Firebase Authentication using email and password

```

16   User? _user;
17
18   void signIn() async {
19     try {
20       UserCredential userCredential = await _auth.signInWithEmailAndPassword(
21         email: _emailController.text, password: _passwordController.text);
22       setState(() {
23         _user = userCredential.user;
24         if (_user != null) {
25           Navigator.of(context).popAndPushNamed(Hompage.routeName);
26         }
27       });
28     } on FirebaseAuthException catch (e) {
29       if (e.code == 'user-not-found') {
30         print('No user found for that email.');
```

Fig 3.2.3: Source code for Sign in process

Once the user lands on the home screen first we collect information of the current user from Firebase Realtime Database with the help of user id and change name text from user name. We have collected bring lists for current users by matching user id in every collection in the bring list table. Every list is combined with another user so we need to show list items by another user's name, for that we have collected another user information from users list table from Firebase Realtime Database. We have also added a logout feature in the home screen, for logout we have called Firebase Auth logout method. After logout User object value will be null and User will land on the Auth screen.

When the user arrives at the home screen, we first get information about the current user from the Firebase Realtime Database using the user id and modify the name text from the user name. We gathered bring lists for current users by matching the user id in each collection in the bring list table. Because each list

is associated with another person, we need to display list items by that user's name. To do so, we gathered information about other users from the users list table in the Firebase Realtime Database. We've also added a logout feature to the home screen, which we've dubbed the Firebase Auth logout method. After logging out, the User object value will be null, and the user will be returned to the Auth page.

```
45
46 void _getCurrentUser() async {
47   DataSnapshot userSnapshot = await _usersDbRef
48     .orderByChild('id')
49     .equalTo(_auth.currentUser?.uid.trim())
50     .once();
51   Map userMap = userSnapshot.value;
52   userMap.forEach((key, value) {
53     setState(() {
54       _currentUser = Users(value['name'], value['id'], value['email'],
55         value['bring_list_ids']);
56     });
57   });
58
59   DataSnapshot bringListSnapShot = await _bringListDbRef
60     .orderByChild('from')
61     .equalTo(_auth.currentUser?.uid.trim())
62     .once();
63   Map? bringListMap = bringListSnapShot.value;
64   bringListMap?.forEach((key, value) {
65     setState(() {
66       _bringItemList.add(BringListItem(value['list_items'],
67         value['bring_list_id'], value['to'], value['from']));
68     });
69   });
70   bringListSnapShot = await _bringListDbRef
71     .orderByChild('to')
72     .equalTo(_auth.currentUser?.uid.trim())
73     .once();
74   bringListMap = bringListSnapShot.value;
75   bringListMap?.forEach((key, value) {
76     setState(() {
77       _bringItemList.add(BringListItem(value['list_items'],
78         value['bring_list_id'], value['to'], value['from']));
79     });
80   });
81   // print(_bringItemList[0].listItems);
82 }
```

Fig 3.2.4: Source code for collecting current user info and connected user list

```

33
34 Future<Users?> getUserDetail(String userId) async {
35     DataSnapshot userSnapshot =
36         await _usersDbRef.orderByChild('id').equalTo(userId).once();
37     Map userMap = userSnapshot.value;
38     Users? user;
39     userMap.forEach((key, value) {
40         user = Users(
41             value['name'], value['id'], value['email'], value['bring_list_ids']);
42     });
43     return user;
44 }
45

```

Fig 3.2.5: Source code for collecting other user info

```

84 void _logout() {
85     _auth.signOut().then((value) {
86         Navigator.of(context).popAndPushNamed(AuthScreen.routeName);
87     });
88 }
89

```

Fig 3.2.6: Source code for logout

If a user clicks on any item from another user's list he/she will land on the Item list screen. We pass arguments (bring list id) from user list screen to item list screen. In the Items list screen we fetch all the items present in the Items list from Firebase Realtime Database. These items will be used in the bottom sheet when users select items into the list. Bring list id represents a unique list between two users and using bring list id we fetch previous added items into the list. When the user selects an item from the bottom sheet then we add the item into the bring list using bring list id. When a user searches for an item and the item is not present in the bottom sheet and he/she clicks on add new item then we add item into the Bring list as well as into the Items list. When the user clicks on the delete button of any selected item then we delete the item from the Bring list.

When a user clicks on an item on another recipient's list, he or she is directed to the Item list screen. From the user list screen to the item list screen, we

send parameters (bring list id). In the Items list screen, we retrieve all of the items in the Items list from the Firebase Realtime Database. When users choose things from the list, these items will be used on the bottom sheet. Bring list id defines a unique list shared by two people, and we utilize it to retrieve previously added things to the list. When a user picks an item from the bottom sheet, we add it to the bring list by utilizing the bring list id. When an users search for an item and it is not found in the bottom sheet, and he or she clicks on add new item, we add the item to both the Bring list and the Items list. When a user hits the delete button on a chosen item, we remove it from the Bring list.

```
36 void _getAllItems() async {  
37   DataSnapshot allItemsSnapShot = await _allItemsDbRef.get();  
38   (allItemsSnapShot.value as Map).forEach((key, value) {  
39     _allItems.add(Item(value['name'], value['id']));  
40   });  
41 }  
42
```

Fig 3.2.7: source code for collecting all Items from database

```
73 void _getSelectedItem(Item item) {  
74   if (_items.contains(item)) {  
75     return;  
76   }  
77   // setState(() {  
78  
79     // });  
80   _items.add(item);  
81   // _allItems.add(item);  
82   String newListItemIds = '';  
83   _items.forEach((element) {  
84     newListItemIds = newListItemIds + element.id + ',';  
85   });  
86   // newListItemIds  
87   newListItemIds = newListItemIds.substring(0, newListItemIds.length - 1);  
88   _updateBringList(newListItemIds, _bringListId);  
89 }  
90
```

Fig 3.2.8: Source code for adding new item into Bring list

```

55 void _updateBringList(String value, String child) {
56     _bringListDbRef
57         .child(child)
58         .update({'list_items': value}).then((value) => setState(() {
59             print('object');
60             }));
61 }

```

Fig 3.2.9: Source code for updating Bring List table in Database

```

90
91 void _addNewItemFromSearch(String name) async {
92     var key = _allItemsDbRef.push().key;
93     Item item = Item(name, key);
94     _allItemsDbRef.child(key).set({'name': name, 'id': key});
95     _getSelectedItem(item);
96 }
97

```

Fig 3.2.10: Source code for adding new item into Item list table

```

62
63 void _delete(int pos) {
64     _items.remove(_items[pos]);
65     String newListItemIds = '';
66     _items.forEach((element) {
67         newListItemIds = newListItemIds + element.id + ',';
68     });
69     newListItemIds = newListItemIds.substring(0, newListItemIds.length - 1);
70     _updateBringList(newListItemIds, _bringListId);
71 }
72

```

Fig 3.2.11: Source code for deleting an item from Bring list

3.3 DESIGN

This section explains the design and development of database schema that is used in BRINGLIXT app. we have created users table, bring_list table and item list table in Firebase realtime database with required attributes.

This subsection describes the architecture and implementation of the database schema used in the BRINGLIXT programme. In Firebase realtime database, we constructed users table, bring list table, and item list table with

needed properties.

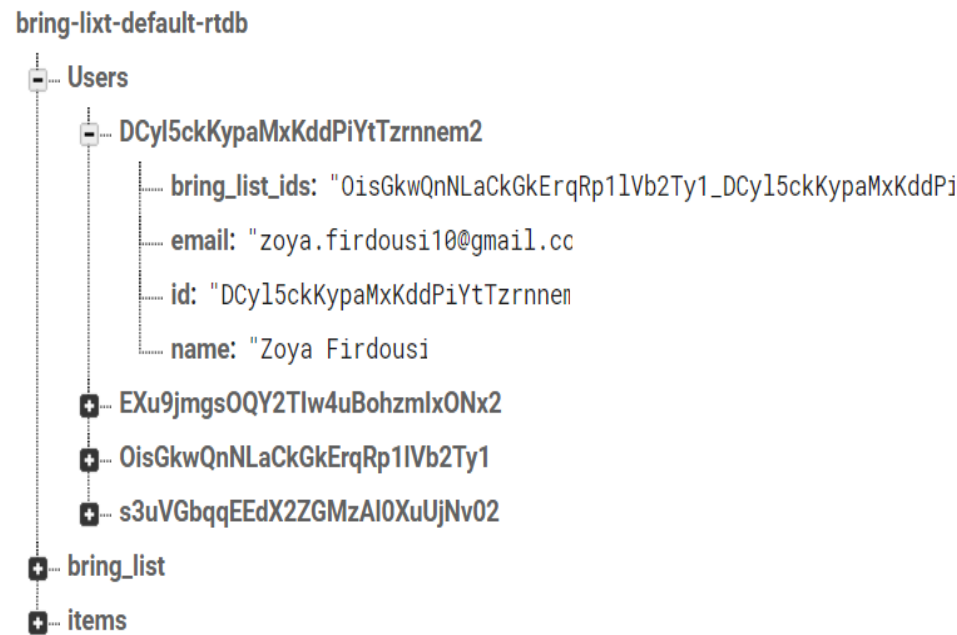


Fig 3.3.1: Users table with required attributes

bring-lix-rtdb



Fig 3.3.2: bring_list table with required attributes

bring-lix-rtdb

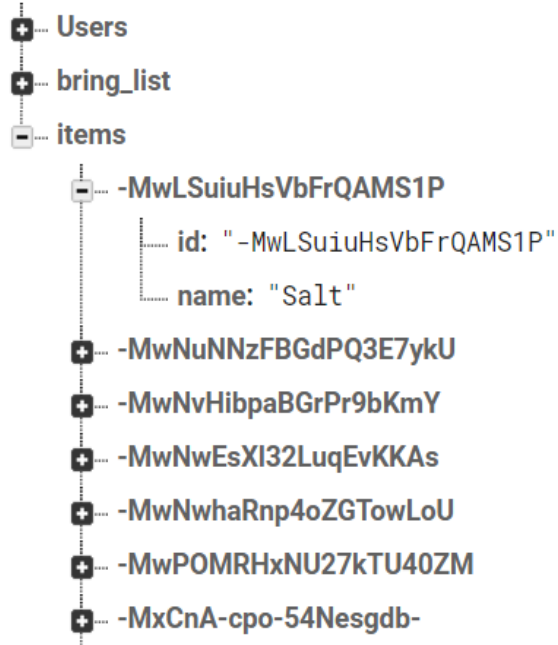


Fig 3.3.3: Items table with required attributes

3.4 SCOPE

Scope of Firebase database tables is to manage users and items. Every user will have a different user list and every user list will have a different bring list. To distinguish between these things we have created tables with proper ids.

All items list is used in every bring list with required items so we have added item id and added this item into bring_list and every bring list item has their own bring_list_id so that we can identify which user has which bring lists.

CHAPTER 4

4 TOOLS AND TECHNOLOGY USED

4.1 VISUAL STUDIO CODE

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity).

Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

Visual Studio Code includes multiple extensions for FTP, allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software.

Visual Studio Code allows users to set the code page in which the active document is saved, the newline character, and the programming language of the active document. This allows it to be used on any platform, in any locale, and for any given programming language.

4.2 XCODE 13

Xcode 13 adds powerful new team development features, perfect for working with Xcode Cloud as well as with GitHub, Bitbucket, and GitLab collaboration features. Initiate, review, comment, and merge pull requests directly within Xcode. See your teammates' comments right inside your code. And quickly compare any two versions of your code files.

Easy setup: Take advantage of a complete workflow to manage pull requests directly within Xcode. Create new requests, see a queue of pull requests

ready for your review, and quickly view, build, and test results generated locally or by Xcode Cloud.

Team comments inline: Comments that your teammates make on code during review display inline within the code editor, along with the name and avatar of the reviewer.

Quick compare: Jump into a comparison view of any two versions of your source code, in any editor, while retaining your current editing context.

Apps are code-signed using an Apple-hosted service that manages all of your certificates, making App Store submission easier and more reliable. Just sign in to Xcode using your Apple ID, and your Mac is configured for development and deployment based on your membership roles and permissions.

Bottom bar: The always-visible bottom bar shows useful information, such as line and column, and offers direct access to quickly compare different versions of code.

Vim mode: Many common key combinations and editing modes familiar to Vim users are supported directly within the code editor, using the new bottom bar to show mode indicators. Vim mode: Many typical key combinations and editing modes known to Vim users are available directly within the code editor, with mode indications displayed in the new bottom bar.

CarPlay simulator: This new simulator for Mac offers complete control over screen size, resolution, and variable input methods, so you can test your CarPlay experience across many different car platforms.

Instant crash reports and feedback: Crash reports from your TestFlight users appear in Xcode Organizer within just a few minutes, complete with user feedback so your testers can tell you exactly what was happening when they experienced the crash. The crash log displays inline and you can jump to the related code with just a click.

4.3 ANDROID STUDIO (IDE)

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA . On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Apply Changes to push code and resource changes to your running app without restarting your app
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

4.4 iOS SIMULATOR

It's often convenient to develop your app directly on your computer rather than having to physically interact with an iPhone and iPad and load your app over the network, which may be slow under some conditions such as if you need to use a tunnel connection because LAN isn't possible on your network.

This guide explains how you can install the iOS simulator on your Mac and use it for developing your app. It is not possible to install the iOS Simulator on any operating system except macOS; if you want to develop an app for iOS from a Windows machine then you will need to use a physical iOS device.

This step is very easy but it takes a while. Open up the Mac App Store, search for Xcode, and hit install (or update if you have it already). If you're not able

to update because your operating system is out of date, we recommend updating your operating system to the latest version and then updating Xcode. You may run into issues further down the line if your Xcode version is out of date, for example you may not be able to submit your app to the App Store.

4.5 ANDROID STUDIO EMULATOR

The Android Emulator simulates Android devices on your computer so that you can test your application on a variety of devices and Android API levels without needing to have each physical device.

The emulator provides almost all of the capabilities of a real Android device. You can simulate incoming phone calls and text messages, specify the location of the device, simulate different network speeds, simulate rotation and other hardware sensors, access the Google Play Store, and much more.

Testing your app on the emulator is in some ways faster and easier than doing so on a physical device. For example, you can transfer data faster to the emulator than to a device connected over USB.

The emulator comes with predefined configurations for various Android phone, tablet, Wear OS, and Android TV devices.

The Android Emulator has additional requirements beyond the basic system requirements for Android Studio, which are described below:

- SDK Tools 26.1.1 or higher
- 64-bit processor
- Windows: CPU with UG (unrestricted guest) support
- HAXM 6.2.1 or later (HAXM 7.2.0 or later recommended)

The use of hardware acceleration has additional requirements on Windows and Linux:

- Intel processor on Windows or Linux: Intel processor with support for Intel VT-x, Intel EM64T (Intel 64), and Execute Disable (XD) Bit functionality

- AMD processor on Linux: AMD processor with support for AMD Virtualization (AMD-V) and Supplemental Streaming SIMD Extensions 3 (SSSE3)
- AMD processor on Windows: Android Studio 3.2 or higher and Windows 10 April 2018 release or higher for Windows Hypervisor Platform (WHPX) functionality

4.6 TECHNOLOGY USED

FLUTTER: Flutter is an open-source UI software development kit created by Google. It is used to develop cross platform applications for Android, iOS, Linux, macOS, Windows, Google Fuchsia,[4] and the web from a single codebase.

First described in 2015,[citation needed] Flutter was released in May 2017.

On May 6, 2020, the Dart software development kit (SDK) version 2.8 and Flutter 1.17.0 were released, adding support for the Metal API which improves performance on iOS devices by approximately 50%, as well as new Material widgets and network tracking development tools.

On March 3, 2021, Google released Flutter 2 during an online Flutter Engage event. This major update brought official support for web-based applications with a new CanvasKit renderer and web specific widgets, early-access desktop application support for Windows, macOS, and Linux and improved Add-to-App APIs. This release also utilized Dart 2.0 that featured sound null-safety, which caused many breaking changes and issues with many external packages; however, the Flutter team included instructions and tools to mitigate these issues

On September 8th, 2021, Dart 2.14 and Flutter 2.5 were released by Google. The update brought improvements to the Android full-screen mode and the latest version of Google's Material Design called Material You. Dart received two new updates, standardizing lint conditions and marking support for Apple Silicon as stable.

Flutter apps are written in the Dart language and make use of many of the language's more advanced features

DART: Dart is a programming language designed for client development, such as for the web and mobile apps. It is developed by Google and can also be used to build server and desktop applications.

Dart is an object-oriented, class-based, garbage-collected language with C-style syntax. Dart can compile to either native code or JavaScript. It supports interfaces, mixins, abstract classes, reified generics, and type inference.

The Dart SDK contains two Dart-to-JavaScript compilers. During development, `dartdevc` supports quick refresh cycles. For the final version of an app, `dart2js` produces deployable JavaScript.

The first compiler to generate JavaScript from Dart code was `dartc`, but it was deprecated. The second Dart-to-JavaScript compiler was `Frog`. It was written in Dart, but never implemented the full semantics of the language. The third Dart-to-JavaScript compiler was `darts`. An evolution of earlier compilers, Dart is written in Dart and intended to implement the full Dart language specification and semantics.

On March 28, 2013, the Dart team posted an update on their blog addressing Dart code compiled to JavaScript with the `dart2js` compiler, stating that it now runs faster than handwritten JavaScript on Chrome's V8 JavaScript engine for the DeltaBlue benchmark.

The Dart language is type safe; it uses static type checking to ensure that a variable's value always matches the variable's static type. Sometimes, this is referred to as sound typing. Although types are mandatory, type annotations are optional because of type inference. The Dart typing system is also flexible, allowing the use of a dynamic type combined with runtime checks, which can be useful during experimentation or for code that needs to be especially dynamic.

Dart offers sound null safety, meaning that values can't be null unless you

say they can be. With sound null safety, Dart can protect you from null exceptions at runtime through static code analysis. Unlike many other null-safe languages, when Dart determines that a variable is non-nullable, that variable is always non-nullable. If you inspect your running code in the debugger, you'll see that non-nullability is retained at runtime (hence sound null safety).

KOTLIN: Kotlin is a cross-platform, statically typed, general-purpose programming language with type inference. Kotlin is designed to interoperate fully with Java, and the JVM version of Kotlin's standard library depends on the Java Class Library,[4] but type inference allows its syntax to be more concise. Kotlin mainly targets the JVM, but also compiles to JavaScript (e.g., for frontend web applications using React) or native code via LLVM (e.g., for native iOS apps sharing business logic with Android apps). Language development costs are borne by JetBrains, while the Kotlin Foundation protects the Kotlin trademark.

On 7 May 2019, Google announced that the Kotlin programming language is now its preferred language for Android app developers. Since the release of Android Studio 3.0 in October 2017, Kotlin has been included as an alternative to the standard Java compiler. The Android Kotlin compiler produces Java 8 bytecode by default (which runs in any later JVM), but lets the programmer choose to target Java 9 up to 17, for optimization,[9] or allows for more features; has bidirectional record class interoperability support for JVM, introduced in Java 16, considered stable as of Kotlin 1.5.

Kotlin support for compilation directly to JavaScript (i.e., the classic back-end) is considered stable since version 1.3, while the new Kotlin/JS(IR) is in beta as of version 1.5.30. The new optimized implementations of Kotlin/JVM(IR) and Kotlin/JS (IR-based) were introduced in version 1.4. Kotlin/JVM(IR) is considered stable and enabled by default since version 1.5. Kotlin/Native (for e.g. Apple silicon support) is considered beta since version 1.3.

Development lead Andrey Breslav has said that Kotlin is designed to be an industrial-strength object-oriented language, and a "better language" than Java, but still be fully interoperable with Java code, allowing companies to make a gradual migration from Java to Kotlin.

Kotlin variable declarations and parameter lists have the data type come after the variable name (and with a colon separator), similar to Ada, BASIC, Pascal, TypeScript and Rust. This, according to an article from Roman Elizarov, current project lead, results in alignment of variable names and is more pleasing to eyes especially when there are a few variable declarations in succession and one or more of the types is too complex for type inference or needs to be declared explicitly for human readers to understand.

Variables in Kotlin can be read-only, declared with the `val` keyword, or mutable, declared with the `var` keyword. Variables in Kotlin can be read-only, declared with the `val` keyword, or mutable, declared with the `var` keyword. In addition to the classes and member functions (which are equivalent to methods) of object-oriented programming, Kotlin also supports procedural programming with the use of functions.[29] Kotlin functions and constructors support default arguments, variable-length argument lists, named arguments and overloading by unique signature. Class member functions are virtual, i.e. dispatched based on the runtime type of the object they are called on.

According to Kotlin developers, you can call JavaScript code from Kotlin, e.g. write full, type-safe React applications, or write and maintain full-stack web applications sharing validation logic with the frontend, or you can "generate libraries from your Kotlin code that can be consumed as modules from any code base written in JavaScript or TypeScript".

SWIFT: Swift is a general-purpose, multi-paradigm, compiled programming language developed by Apple Inc. and the open-source community. First released in 2014, Swift was developed as a replacement for Apple's earlier programming language Objective-C, as Objective-C had been largely unchanged since the early 1980s and lacked modern language features. Swift works with Apple's Cocoa and Cocoa Touch frameworks, and a key aspect of Swift's design was the ability to interoperate with the huge body of existing Objective-C code developed for Apple products over the previous decades. It is built with the open source LLVM compiler framework and has been included in Xcode since version 6, released in 2014. On Apple platforms, it uses the Objective-C runtime library, which allows C,

Objective-C, C++ and Swift code to run within one program.

Apple intended Swift to support many core concepts associated with Objective-C, notably dynamic dispatch, widespread late binding, extensible programming and similar features, but in a "safer" way, making it easier to catch software bugs; Swift has features addressing some common programming errors like null pointer dereferencing and provides syntactic sugar to help avoid the pyramid of doom. Swift supports the concept of protocol extensibility, an extensibility system that can be applied to types, structs and classes, which Apple promotes as a real change in programming paradigms they term "protocol-oriented programming" (similar to traits).

Swift is the result of the latest research on programming languages, combined with decades of experience building Apple platforms. Named parameters are expressed in a clean syntax that makes APIs in Swift even easier to read and maintain. Even better, you don't even need to type semi-colons. Inferred types make code cleaner and less prone to mistakes, while modules eliminate headers and provide namespaces. To best support international languages and emoji, Strings are Unicode-correct and use a UTF-8 based encoding to optimize performance for a wide-variety of use cases. Memory is managed automatically using tight, deterministic reference counting, keeping memory usage to a minimum without the overhead of garbage collection.

FIREBASE: Firebase is a platform developed by Google for creating mobile and web applications. It was originally an independent company founded in 2011. In 2014, Google acquired the platform and it is now their flagship offering for app development.

Firebase security applies Google's internal expertise to easily build app sign-ins. Develop simple, free multi-platform sign-in with Firebase Authentication. For Mobile or Web Apps. Boost App Engagement. Monitor App Performance. Build Extraordinary Apps.

Store & sync data between your users in realtime with the Firebase Realtime Database. Optimize for offline use and build strong user-based security

for your app. Monitor App Performance. Customize Your App.

Firebase is expanding to become a unified app platform for Android, iOS and mobile web development. We're adding new tools to help you develop faster, improve app quality, acquire and engage users, and monetize apps. On top of this, we're launching a brand new analytics product that ties everything together, all while staying true to the guiding principles we've had from the beginning:

- Developer experience matters. Ease-of-use, good documentation, and intuitive APIs make developers happy.
- Work across platforms. We'll support you whether you're building for iOS, Web, or Android.
- Integrate where possible. Firebase has one SDK, one console, and one place to go for documentation and support. You can mix-and-match any of our features and, where it makes sense, data flows between them to help you do more, faster.

To help you build better apps, our suite of backend services is expanding.

Google Cloud Messaging, the most popular cloud-to-device push messaging service in the world, is integrating with Firebase and changing its name to Firebase Cloud Messaging (FCM). Available for free and for unlimited usage, FCM supports messaging on iOS, Android, and the Web, and is heavily optimized for reliability and battery-efficiency. It's built for scale and already sends 170 billion messages per day to two billion devices.

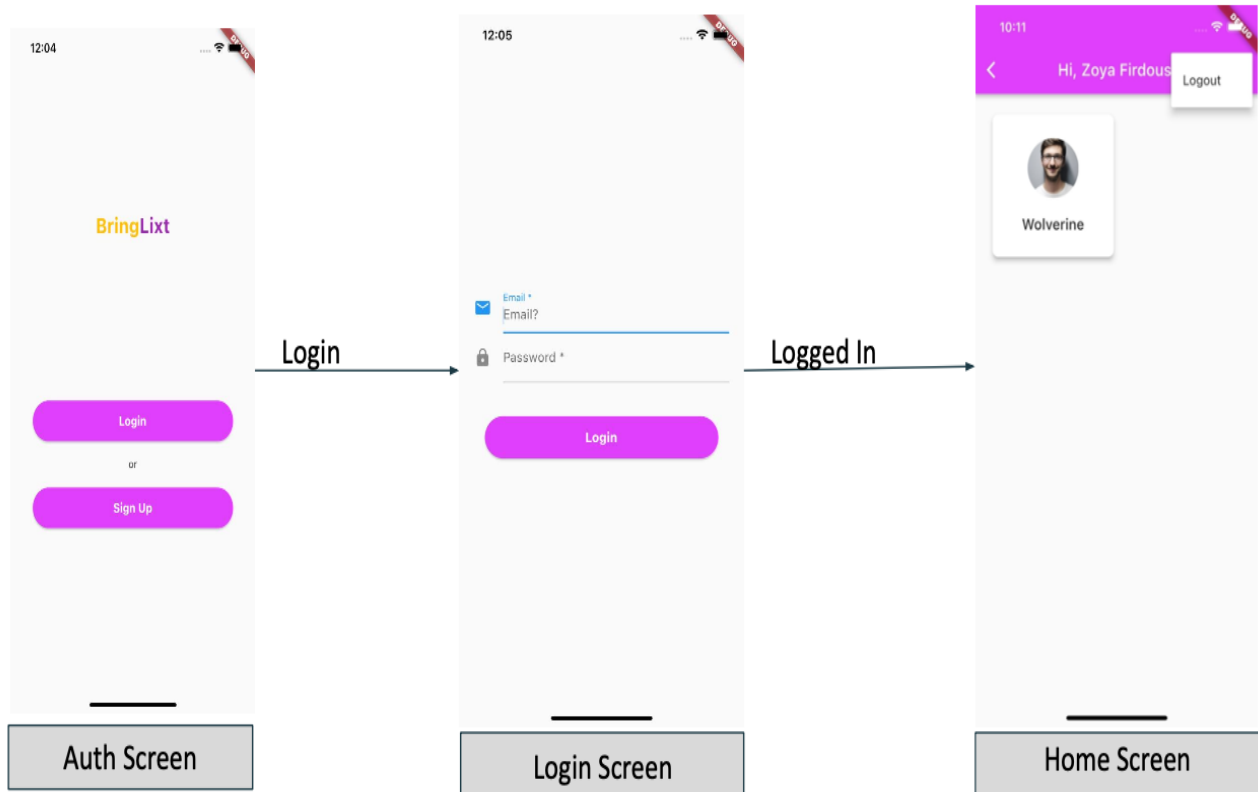
One of our most requested features is the ability to store images, videos, and other large files. We're launching Firebase Storage so developers can easily and securely upload and download such files. Firebase Storage is powered by Google Cloud Storage, giving it massive scalability and allowing stored files to be easily accessed by Google Cloud projects. The Firebase Storage client SDKs have advanced logic to gracefully handle poor network conditions.

Firebase Remote Config gives you instantly-updatable variables that you can use to tune and customize your app on the fly to deliver the best experience to your users. You can enable or disable features or change the look and feel without having to publish a new version. You can also target configurations to specific Firebase Analytics Audiences so that each of your users has an experience that's tailored for them.

In addition, we're continuing to invest heavily in our existing backend products, Firebase Realtime Database, Firebase Hosting, and Firebase Authentication. Authentication has seen the biggest updates, with brand new SDKs, and an upgraded backend infrastructure. This provides added security, reliability, and scale using the same technologies that power Google's own accounts. We've also added new Authentication features including email verification and account linking. For Hosting, custom domain support is now free for all developers, and the Database has a completely rebuilt UI. We're working hard on other great Realtime Database features, stay tuned for those.

CHAPTER 5

5 RESULT



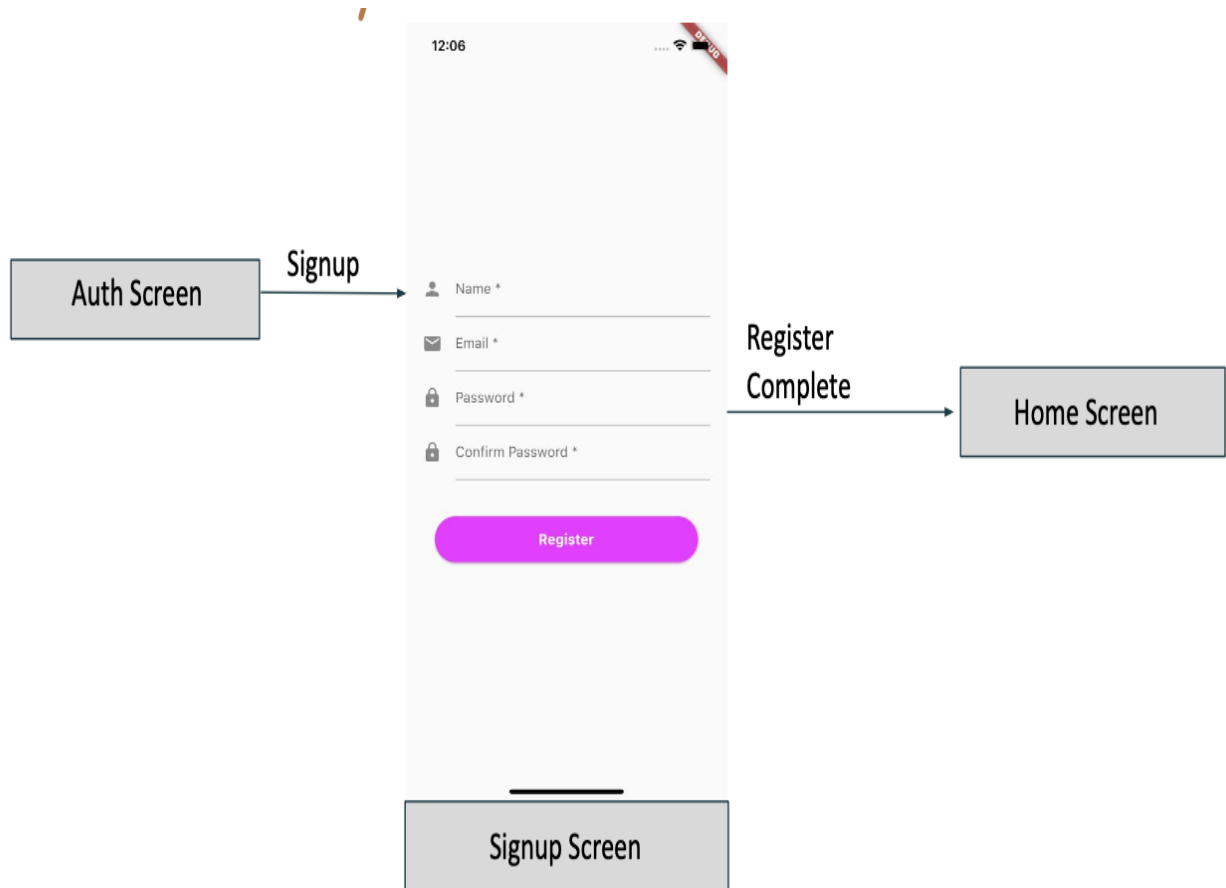


Fig 5.1: Results with user journey

CHAPTER 6

6 CONCLUSION AND FUTURE WORK

This paper described the development and evaluation of the BRINGLIST app, a hybrid app (iOS & Android platform) for creating and managing lists of daily used items that are required to purchase from the market with family members who shared the lists. The evaluation of the mobile app suggested exciting findings that could encourage more studies in improving the design and communication flows of mobile apps for supporting the creation and management of item lists collaboratively among family members.

This study presented the creation and testing of the BRINGLIST app, a hybrid software (iOS and Android platforms) for building and managing lists of daily used things that must be purchased from the market with family members who shared the lists. The assessment of the mobile app yielded promising results that might spur further research into enhancing the design and communication patterns of mobile apps for jointly creating and managing item lists among family members.

In future, App push notification service could be added in this app so that users will be notified whenever a new item is added in the list. action buttons also can be added in notification for removing an added item so that users do not have to open this app for removing purchased items. Item list could be in categorization format while adding items into the list, it will ease the operation time. Every item could have a proper image so that users can easily identify every item.

In the long term, this app might include an app push notification service so that users are alerted anytime a new item is added to the list. Action buttons for deleting an additional item may also be added to notifications so that users do not have to launch this app to remove purchased things. While adding items to the list, the item list might be in categorization format, which would shorten the operation time. Every object might have its own picture so that consumers can readily recognise it.

REFERENCES

- [1] <https://pub.dev> - for flutter libraries and their coding documents.
- [2] <https://firebase.flutter.dev/docs/> - for setting firebase into our app and related coding docs.
- [3] <https://medium.com/firebase-developers/dive-into-firebase-auth-on-flutter-email-and-link-sign-in-e51603eb08f8> - user sign in and sign up related docs and coding
- [4] <https://www.udemy.com/course/learn-flutter-dart-to-build-ios-android-apps/> - Udemy flutter tutorial for hybrid app.
- [5] Navdeep Singh, "Study of Google Firebase API for Android", International Journal of Innovative Research in Computer and Communication Engineering, vol 4, issue 9, september 2016.
- [6] Saurabh Bharadwaj, Priyanka Chouhan, Richa Sharma and Preeti Sharma, "Android Operating system", International Journal of Engineering Technology & Management Research.
- [7] The Smart Shopping List An effective mobile solution for grocery list-creation process -Harsha Jayawilal W.A. Faculty of Information Technology University of Moratuwa Katubedda, Sri Lanka.
- [8] <https://www.statista.com/statistics/467163/forecast-ofsmartphone-users-in-india>, The Statistics Portal.
- [9] Navdeep Singh, "Study of Google Firebase API for Android", International Journal of Innovative Research in Computer and Communication Engineering, vol 4, issue 9, september 2016.
- [10] William Enck, Machigar Ongtang and Patrick McDaniel, "Understanding Android Security", Pennsylvania State University.
- [11] Saurabh Bharadwaj, Priyanka Chouhan, Richa Sharma and Preeti Sharma, "Android Operating system", International Journal of Engineering Technology & Management Research, vol 1, issue 1.
- [12] M. Achouri, A. Alti, M. Derdour, S. Laborie, and P. Roose. Smart fog computing for efficient situation management in smart health environments, Journal of Information and Communication Technology, vol. 17, pp. 537-567, 2018. <https://doi.org/10.1145/3167020.3167022>

- [13] P. Deshwal, P. Khattar, A. A. Srivastava, and N. Kakkar. Mobile learning readiness and time of daily use, *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, 2019.
- [14] N. Katuk, N. H. Zakaria, and K. R. Ku-Mahamud. Mobile Phone Sensing using the Built-in Camera, *International Journal of Interactive Mobile Technologies*, vol. 13, pp. 102-114, 2019. <https://doi.org/10.3991/ijim.v13i02.10166>
- [15] A. Baharum, N. H. Nazlan, N. A. M. Noor, R. Hanapi, I. Ismail, and N. A. M. Noh. Information Visualization for Food Mobile Application: Hangri 2.0, *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, pp. 306-314, 2019.
- [16] K. Yanai, T. Maruyama, and Y. Kawano. A Cooking Recipe Recommendation System with Visual Recognition of Food Ingredients, *International Journal of Interactive Mobile Technologies*, vol. 8, pp. 28-34, 2014. <https://doi.org/10.3991/ijim.v8i2.3623>
- [17] W. Koo. Usage of Smartphone Applications: A Descriptive Study of Top 100 US Retailers, *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 10, pp. 54-58, 2016. <https://doi.org/10.3991/ijim.v10i3.5827>
- [18] H. Jayantilal and S. Premaratne. The smart shopping list: An effective mobile solution for grocery list-creation process, in *The Proceedings of 2017 IEEE 13th Malaysia International Conference on Communications (MICC)*, 2017, pp. 124-129.
- [19] R. Bassett, B. Beagan, and G. E. Chapman. Grocery lists: connecting family, household and grocery store, *British Food Journal*, vol. 110, pp. 206-217, 2008. <https://doi.org/10.1108/00070700810849916>
- [20] A. Stable, "Daily Living Skills," in *Encyclopedia of Autism Spectrum Disorders*, F. R. Volkmar, Ed., ed New York, NY: Springer New York, 2013, pp. 839-840.
- [21] V. Gil, K. D. Bennett, and P. M. Barbetta. Teaching Young Adults with Intellectual Disability Grocery Shopping Skills in a Community Setting Using Least-to-Most Prompting, *Behavior Analysis in Practice*, pp. 1-5, 2019.
- [22] R. Reid and S. Brown. I hate shopping! An introspective perspective, *International Journal of Retail & Distribution Management*, vol. 24, pp. 4-16, 1996