UNIVERSITATEA POLITEHNICA BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

PROIECT DE DIPLOMĂ

ShopSmarter
Aplicație Android bazată pe beaconi Nearby ce găsește oferte relevante din proximitatea utilizatorului

Radu-Tudor Paraschivescu

Coordonator științific:
Șl.dr.ing. Radu-Ioan Ciobanu

BUCUREŞTI

2019

UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE DEPARTMENT

DIPLOMA PROJECT

ShopSmarter
Android App That Uses Nearby Beacons
to Find Relevant Promotions in the Proximity of the User

Radu-Tudor Paraschivescu

Thesis advisor:
Radu-Ioan Ciobanu

BUCHAREST

2019

TABLE OF CONTENTS

SINOPSIS

Este cunoscut faptul că industria comerțului cu amănuntul se îndreaptă din ce în ce mai mult către mediul online datorită costurilor reduse și a disponibilității mai mari pentru clienți. În ciuda acestui fapt, există și comercianți care vând numai în magazine, în timp ce mulți clienți preferă "modalitatea tradițională" de a cumpăra direct din magazin. Din cauza acestei situații, există o nevoie reală pentru o aplicație care îmbină cele mai bune caracteristici ale acestor două tipuri de comerț.

Soluția prezentată în această lucrare este o aplicație Android bazata pe context capabilă să afișeze oferte în timp real ale magazinelor din apropiere. Aplicația poate găsi aceste reduceri prin scanarea anumitor dispozitive care proiectează un semnal Bluetooth, numite beaconi. Aceste dispozitive trebuie plasate în magazine și stochează informații despre vânzările disponibile. Utilizatorii aplicației pot stabili anumite preferințe pentru a adapta căutarea la anumite nevoi specifice (spre exemplu, căutarea reducerilor ce depășesc un anumit procent).

ABSTRACT

It is a known fact that the retailing industry is shifting more and more towards online shopping due to reduced costs and higher availability for the customers. Despite this, there are still retailers who do business only in offline stores, while for lots of customers the "traditional way" of buying directly from the shop is still preferred. Because of this situation, there is a real need for an application that merges the best features of these two major ways of shopping.

The solution presented in this paper is a context-based Android application able to display real-time offers from nearby stores. The app can find these discounts by scanning for certain devices that project a Bluetooth signal, called beacons. These devices are to be placed in stores and contain information about the available sales. The users of the app are able to set certain preferences to tailor the search for their specific needs (seeking only discounts that exceed a certain threshold).

ACKNOWLEDGMENTS

Special thanks go to my thesis advisor, Radu-Ioan Ciobanu, for the incredible help and support through all the phases of developing the solution and writing this paper. From outlining the whole project, exchanging ideas and reviewing pieces of work, he had a pivotal role and I enjoyed working with him.

Also, I want to acknowledge the professors, teaching assistants and colleagues with whom I had contact in these 4 years at the Faculty of Automatic Control and Computers. With good and bad, they helped me develop, learn and arrive at one step to becoming an engineer.

Last, but certainly not least, I would like to thank my family, especially my parents, for always being there for me and putting my needs above their own. Always encouraging me and helping every way they could, I could not be here without them.

1 INTRODUCTION

1.1 Context

The retailing industry is one of the biggest in the world, worth trillions of US dollars. The total revenue of only the biggest 250 retailers is 4.4 trillion US dollars [1]. It's safe to say that the customers have the upper hand, because of all the revenue generated by their purchases. Technology is also a big part of it, because of the ability to always be connected. The term "everywhere commerce" appeared [1], meaning that the consumers can shop on their own terms, whenever they want, online or in stores.

A question that raises now is if online shopping, after a steady and continued growth, will come to put an end to the traditional shopping. The answer is that it will not happen at the moment, but it might happen in the future. Online sales are increasing year after year, but there are still plenty of customers that prefer to shop in stores. In the United States, these sales are 10 times bigger than those made online [2]. Nevertheless, the online registers a yearly growth that is 3 times higher [2].

1.2 Problem

In the fast-changing world of retailing, technology is an important pillar. It defines how the customers can engage with the retailers, while the demand for better and cheaper products increases. Despite the technological advancements and improved advertising campaigns, a retailer getting noticed is a big challenge because of a broader competition. This is an even more difficult task for the retailers who sell only in stores or their biggest chunk of profit comes from offline sales.

The proposed solution comes as a useful tool that shall boost the potential customers' eagerness of going into a nearby store, looking at the available products and buying certain items. One of the main features of online shopping is the ease of spotting better prices and discounts. The ShopSmarter Android app is able to display the current promotions available in the stores in the proximity of the user, taking also into account a set of preferences. The search is done by scanning for beacons [3] placed in the partnering stores. A beacon is a hardware device that sends signals to indicate its location while keeping and sending payloads to devices in its proximity [3].

1.3 Objectives

This paper covers only the implementation of the said Android application. The beacons used in testing were simulated by another Android device with a specialized application (Beacon Simulator [4]) that can create and broadcast Eddystone-UID beacons [5].

The Android application can be used only by an authenticated user, either by email and password with a confirmation email or with a Google account. The main objective of the app is to be able to show the sales in the user proximity. An important feature of the app is the possibility of setting certain preferences to filter the search results. The user has the ability to save, see and remove offers as favorites. The favorite products can also be sorted by discount and price. A last important objective was for the user to enjoy the time in the app, seeking for a responsive user interface and an intuitive user experience.

Another main objective of the architecture was the security of the data stored on the simulated beacons. Other apps with beacon searching capabilities shall not able to see the data stored on the simulated beacons.

1.4 Thesis Structure

This thesis presents the implementation of an Android application capable to search for shops and discounts through beacon scanning. Its main purpose is to be a useful tool for the consumers and help find better deals in shops around them.

In the second chapter of the thesis, we will define the features of the app more precisely and analyze how they are beneficial from the perspective of both the consumer and the retailer.

The third chapter covers other related works. We address already existing applications with similar features and use cases. We present apps made for specific shopping centers or retailers, while also taking a look at general apps with shop deals and beacon searching abilities. In the end, we compare all of them to see if our proposed solution is a better fit for the potential users of such apps.

Chapter 4 presents the main technologies used in the implementation, alongside reasoning of why these choices were made instead of other alternatives.

In the fifth chapter, an in-depth presentation of the solution is made, starting from the architecture, continuing with the application flow from the standpoint of a regular user and ending with technical details of implementation.

In Chapter 6 we present the results of the conducted tests for functionality and performance. The functionality of the app is tested by checking whether all the features and specifications are met. For the performance analysis, we cover aspects of reliability, range, detection, and real-life commercial use.

The last chapter presents the conclusions of the paper and the potential subjects for future work.

2 FEATURES

2.1 Analysis From the Consumer Standpoint

Any consumer wants to get the most possible value out of an item bought at the smallest possible price. This may seem an absolute and obvious statement, but is actually one of the 7 biggest trends in retail for 2019 [6]. Paying less for more is a sign of the commitment of consumers to prioritizing value and education on the pricing strategies of the retailers. As technology advances and gets more widely available to everyone, information about convenient products can reach a larger audience and help those in the pursuit of finding the perfect product for their needs.

In order to find out more about the perspective of the consumers, we conducted a short survey to discover further information and opinions on shopping habits, online and offline. At the same time, we wanted to see if the audience is receptive and finds the main idea behind the ShopSmarter app useful. No personal data was collected during the survey, all answers being anonymous.

The survey was completed by a total of 167 persons. It consisted of 7 questions:

- **Q1**: How do you prefer to shop?
- **Q2**: What do you think is online shopping's biggest advantage?
- **Q3**: What do you think is offline (in-store) shopping's biggest advantage?
- **Q4**: To what degree do you search online for sales before going shopping in stores?
- **Q5**: How much would you like to use a mobile app that displays real-time sales and discounts from all nearby stores?
- **Q6**: Which features seem important and useful to you in such an app?
- **Q7**: To what extent do you believe a mobile app with all these features would make you go and shop more in stores?

Questions Q4, Q5 and Q7 asked for a rating between 1 and 5, 1 meaning strongly disagreeing and 5 meaning strongly agreeing.

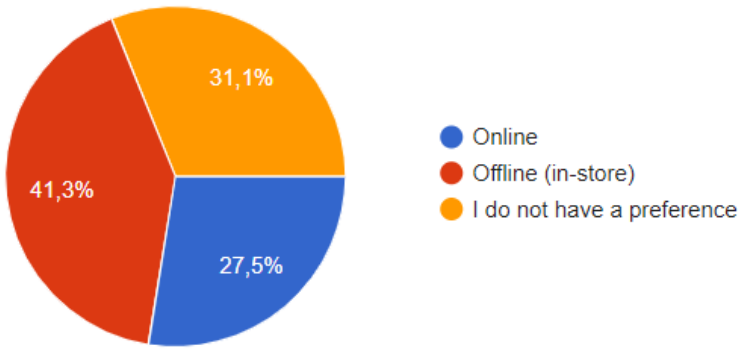

Figure 1. Survey answers to Q1

Question Q1 seeks to find out which way of shopping is preferred by consumers. The question was single choice type and had 3 possible options ("Online ", " Offline " and "I do not have a preference"). By finding out more about the preferred way of shopping, we can understand why the retailing industry is where it is right now.

As can be seen in Figure 1, the most preferred option was offline shopping. This stands in line with the sales statistics because this type of shopping generates the most revenue [2]. The second most picked option was not having a preference, signaling that for lots of those surveyed the means of shopping is not that important as long as they get the wanted results and fulfill their needs. A bit surprising is the last position, occupied by online shopping. Even if it made big strides in the last years and is rising at a much faster pace than in-store shopping [2], online shopping was the most preferred way of shopping only for 27.5% of those surveyed.

Other than the order itself, these aggregated results show us that consumers are getting more and more channel-agnostic. The answers were spread fairly even, the difference between the most and least chosen option being of only 23 votes. Nowadays, shopping becomes more of a journey that distributes over both the online and offline.

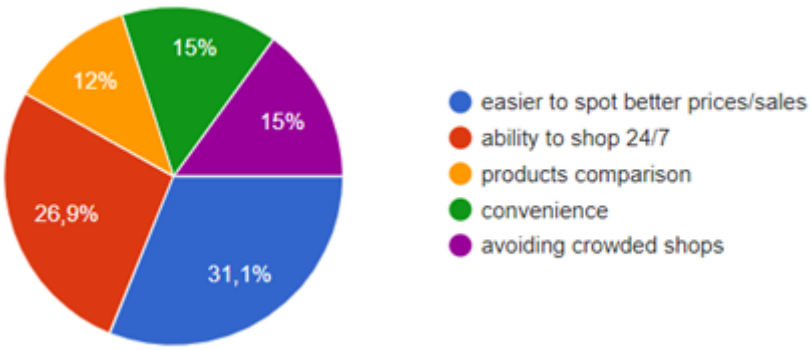

Figure 2. Survey answers to Q2

Question Q2 tries to find what is the biggest advantage of online shopping from the perspective of those surveyed. The options were "easier to spot better prices/sales", "ability to shop 24/7", "products comparison", "convenience" and "avoiding crowded shops". Only a single choice was accepted.

In Figure 2, the aggregated answers can be seen. The biggest advantage, per the survey answers, was the ease of spotting good prices and sales. This is because online campaigns specifically try to bring to the attention of possible customers the biggest available sales as a way to attract more people to the retailer website. If more people come to the website, there is a higher chance of making an acquisition and not only from the discounted products. Also, there are retailers that target the online particularly, by using discounts and offers available only when shopping online. The consumers are compelled by these because they feel like making a great deal that could not be possible in other circumstances.

The second advantage chosen the most times was the ability to shop uninterrupted. This is empowering for the consumers because they do not have to adapt their schedule on the opening and closing hours of one shop if they can place an order at any time on the shop website and just wait for the product to arrive at their doorstep. Convenience and avoiding crowded shops are also important benefits of shopping online, while the ability to compare products is seen as the least important asset.

Because the ease of spotting better sales is considered the main advantage of online shopping, the context for an app like ShopSmarter is favorable. Being able to search for all the available sales in shops near you would help bring an important advantage of online shopping to those who prefer to shop in stores.

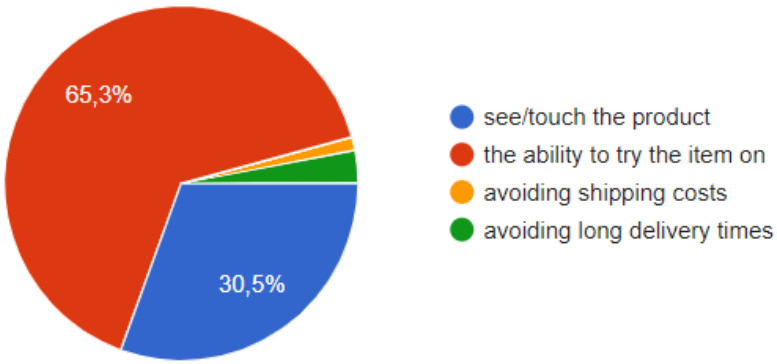

Figure 3. Survey answers to Q3

In Question Q3, the assets of shopping in stores are presented in order to find what is considered the main advantages of this way of shopping. Q3 was a single choice question and had four available options: "see/touch the product", "the ability to try the item on", "avoiding shipping costs" and "avoiding long delivery times".

The most selected answer was the ability to try an item on. This refers mainly to trying clothes and fashion items. The general perception of shopping is for many equivalents with going out in centers with many shops and buying clothes and shoes. This a reason why over 65% percent of those asked about the biggest advantage of offline shopping thought at first at buying clothes and trying them on. The significance of this asset is major since for many fashion items one has to try them in person to see if they fit properly. Buying online a product with the wrong measurements can be really upsetting for the customers because of the inconvenience of returning the item, refunding the money for the acquisition or changing it with another measurement.

Seeing and touching a product is a relevant step in the purchasing process. As the second most voted option, it is clear that consumers consider that you can find a lot more about a product after seeing it, not only in pictures online. There are also cases where the pictures used for some products may be misleading and do not help the consumers make a genuine perception of the item. One can also determine de craftsmanship of a product after feeling the textures and the materials used.

Avoiding shipping costs and long delivery times are not considered important anymore. Because of the small shipping costs perceived, free shipping for orders exceeding a certain amount and possibility to opt for express couriers, only 7 out of the 167 surveyed chose one of these two options as the main advantage of online shopping.

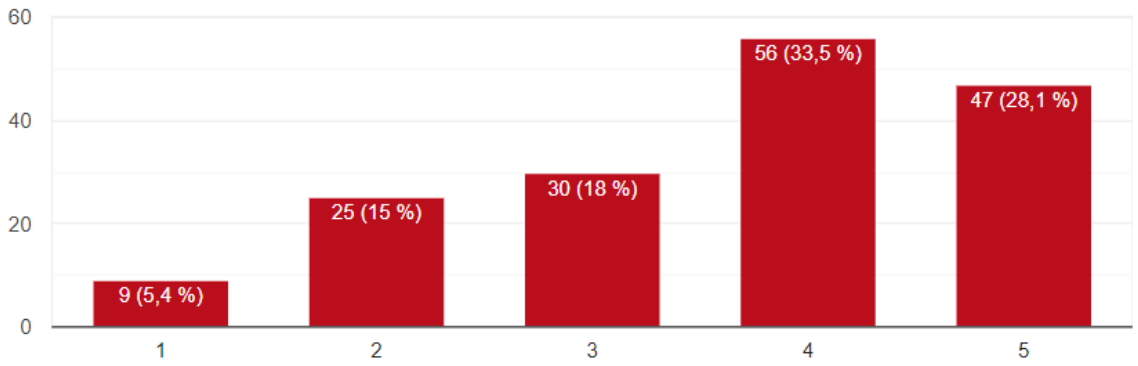

Figure 4. Survey answers to Q4. Oy axis shows the number of responses.

Question Q4 tries to determine the level of research and effort people put in before going to shop in stores. The question had a linear scale from 1 to 5 to determine this level.

The majority of those questioned (61.6%) answered with 4 or 5. This highlights the current trend of consumers wanting to be more informed about prices, available shops, products, and alternatives. Some time ago, the path to purchasing an item did not include much research, however that has changed dramatically. 81% of shoppers acknowledged that before making

a purchase they do online research, while 89% of them begin this process with a search engine [6].

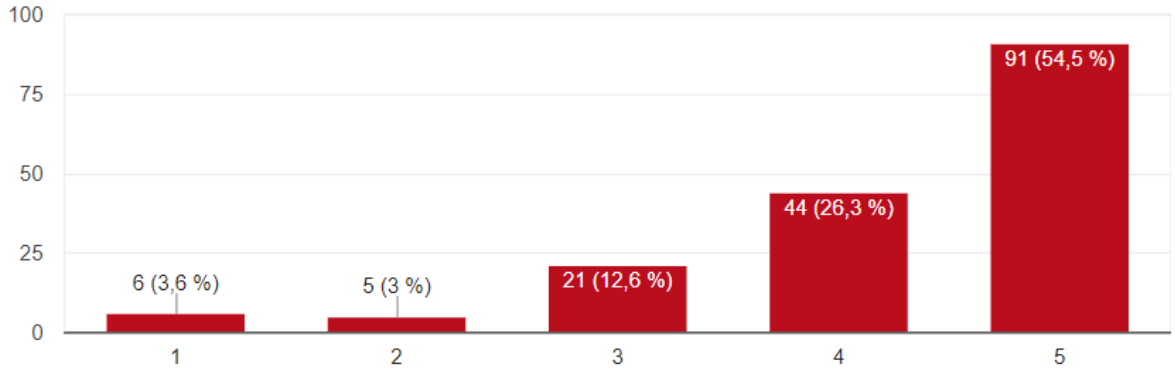

Figure 5. Survey answers to Q5. Oy axis shows the number of responses.

The fifth question tried to find if there is a need on the market for a mobile application that can display real-time offers from the nearby shops. The question had a linear scale from 1 to 5 to determine the level of desire for such an app. As it can be seen in Figure 5, over half of those surveyed strongly agreed on the need for an app with this feature. More than that, 80.8% of the participants in the survey agreed and are aware of this demand by answering with 4 (Agree) or 5 (Strongly Agree). These results secure the foundation of the solution presented in this thesis and highlight a real need in the market for better and cheaper products, something that the ShopSmarter app could help achieve.

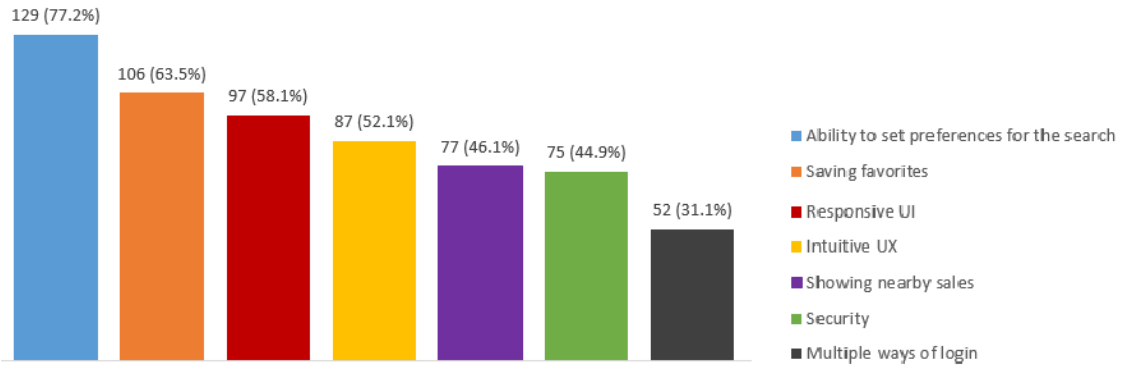

Figure 6. Survey answers to Q6. Oy axis shows the number of responses.

Question Q6 queries the users about the features that would be the most useful for them and would like to see implemented in a mobile app. This question was multiple choice, allowing to mark all the features considered helpful.

Setting preferences for the search enables the users to have more control in the discount lookup and because of this, it was the most wanted feature, being put in the feature list of over 77% of those surveyed. In second place came the ability to save favorites. 63.5% of the participants in the survey see the benefit of saving sales and products to lookup later. There might come up moments when one cannot enter a store right away, so the option to save it for later might come useful.

UI[1] and UX[2] define a set of guidelines and workflows important in the design and use of a technological product that users can interact with [7]. These concepts are at the core of the popularity gained by smartphones and mobile apps in recent years because of their accent on functionality and getting things done fast. It comes as no surprise that a responsive UI and an intuitive UX are next in the list of most wanted features.

The ability to find the best sales in the proximity in the shortest time possible and the security of the data stored in the platform are relevant features, both selected by around 45% of those queried. The last place on this list comes to the possibility of authenticating in multiple ways. In a mobile app, once the user is logged in, one can expect to rarely log out of the account. The importance of having more ways of sign up at the beginning is overlooked after the user has an account created and is logged in.

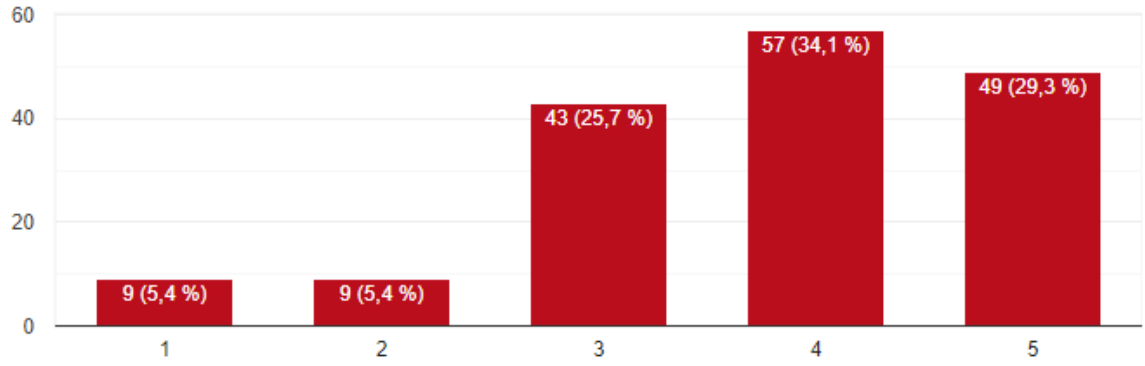

Figure 7. Survey answers to Q7. Oy axis shows the number of responses.

The last question in the questionnaire asks the participant to say to what extent a mobile app with all the features presented at Q6 would make one go shop more in stores. This measured on a scale from 1 to 5.

[1] User Interface
[2] User Experience

As can be seen in Figure 7, the answers are encouraging, with more than half of the participants agreeing that the app would lead to more shopping in stores. 63.4% of those surveyed gave a rating of 4 or 5. The percent of disbelief (rating of 1 or 2) is under 11%.

These results look favorable for the retailers as well, showing the general desire of consumers to hold on to the traditional shopping as long as the experience is getting better with more tools to help them.

2.2 Analysis From the Retailer Standpoint

Globally, the retailing industry had a revenue of 18.8 trillion US dollars in 2018 [8]. This record number is not generated only by the increase in number of retailers, but also by the steady growth of the biggest retailers in the world. In 2001, the minimum revenue required by a company to be in the Top 200 retailers in the world was 2.4 billion US dollars [1]. This number increased steadily until the present day, to 3.36 billion US dollars in 2006, 4.5 billion US dollars in 2011 and 4.73 billion US dollars in 2016 [1].

The global economy is in a period of steady growth. This expansion has increased in Europe, while also stabilizing in two of the main markets of the world, China and the United States [1]. Additionally, there are signs of economic growth in many other emerging markets. For retailers, strong economic growth is an encouraging and welcomed sign. However, hidden risks such as slow growth in evolved markets or political dysfunctions still exist.

Revenue and growth are important factors of the success of one retailer, but probably the most important metric remains the net profit margin[3]. Essentially, this metric illustrates how much of the revenue collected by a company translates into profit. The Top 250 retailers in the world posted a composite net profit margin of 3.2% in 2016 [1]. This comes as a 6% increase from the 3.0% composite net profit margin posted in 2015 [1]. These statistics emphasize the growth of the retail industry from a profit standpoint as well.

Retailers are focusing more and more on online shopping because of the request of the consumers and higher costs in sustaining a chain of stores. In the United States, a record number of over 6,885 stores closed during 2017 [1]. This is a result of retailers focusing on productive locations and opting to close unprofitable venues. If some of the better

[3] The net profit margin equals to how much profit is generated as a percentage of revenue

advantages of online shopping, like the ease of spotting better sales, could be available in stores, the number of closed shops might be smaller.

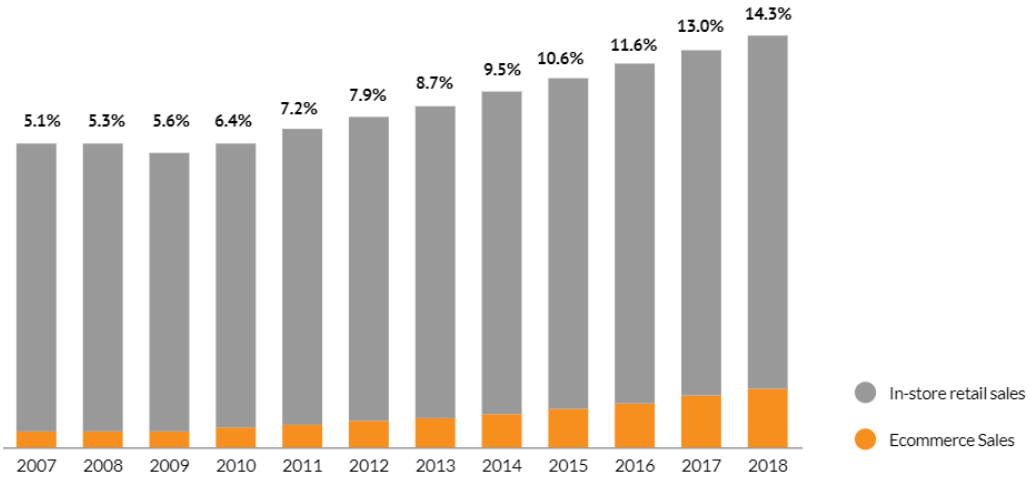

Figure 8. Online sales penetration in the US retail market[4] [9]

The percentage of online sales out of the total sales has increased in each year of the last decade. As can be seen in Figure 8, the percent reached an all-time high of 14.3% in 2018. However, the online is not the perfect general recipe that works for all retailers. The optimal channel structure of a retailer is dependent on the acceptance rate of the customers for the online channel. If this rate is low, medium or high, it means that the most advantageous structure is a pure offline channel, dual channels or a pure online channel [10].

Retailers will try to attract consumers to any of their available channels, either offline or online. Over 85% of the retail sales are made in shops (Figure 8) and even though online shopping is making big strides year after year, shopping in stores will remain the favorite way of the consumers in the near future. Because of this, brands are aware that a shift only to online is risky and must try to improve the experience of traditional shopping. Important trends to follow in 2019 are hyper-personalization and the planet-friendly movement [6].

Hyper-personalization can be condensed as the use of real-time data to produce contextual and relevant services. This technique can be used in a more general manner or be tailored for specific individuals and situations. The introduction of hyper-personalization in the biggest

[4] Sales of items not normally purchased online (e.g., fuel, sales in bars) are not taken into consideration as in-store retail sales

trends of 2019 proves that the solution presented in this paper has the potential of having a big impact in the retail industry. ShopSmarter is giving contextual information based on the position of the user and its preferences, helping pinpoint the best offers available in real time.

The planet-friendly movement refers to buying goods that do not harm the environment, having sustainability as the focus point. Over half of the consumers take sustainability into consideration when making a purchase. 33% make purchases with sustainability in mind, while other 21% do not buy on this principle but would like to [11]. Brands are trying to appear as responsible and environmentally friendly as possible and make use of marketing and social campaigns to promote that.

2.3 List of Features

After analyzing the standpoints of consumers and retailers, we came to the conclusion of the usefulness of the proposed solution for both parts. A rigorous set of features must be defined to fulfill the needs of the consumers. The solution will have the next features:

- authentication by email and password with confirmation email
- authentication with Google account
- showing the sales in the user proximity (50-meter range)
- ability to set certain preferences to filter the search results:
    - minimum price
    - maximum price
    - minimum sale percent
    - product category
    - expired offers
- possibility of saving offers as favorites
- possibility of seeing and removing favorite offers
- ability to sort the favorite offers by discount, ascending or descending price
- fast and responsive user interface
- intuitive user experience.

Authentication is needed in most Android apps because of the need to know the identity of the user. In our solution, the need for signing in is explained by the need of securely saving user data to the cloud. This data is going to be about the saved offers of each user, providing a personalized experience. As options of sign-in, we opted for two methods: with an email and password or with a Google account.

The first sign-in method requires the name of the user that will be shown in the application, a valid email address and a password of at least 6 characters. After signing in, the user will receive a confirmation email and must enter the link provided to verify the email address. Only after verifying the email address, the user will be able to login in the app with the email and password completed in the sign-in step.

The second sign-in method is by using a Google account. Android devices rely on at least one Google account to be able to use the Google Play store or any other apps in the Google Suite. This option of signing in is the fastest and more straight-forward out of the two methods. After clicking the Sign with Google button, the user will be prompted to choose the account he wants to sign in with. After making this choice, the user will log in into the application.

Showing the sales in the proximity of the user is the main feature of the app. The user will be only one button press away from finding what are the best offers nearby. After pressing the search button and waiting for 8 seconds, the found results will come up. The offers will be sorted by shops. One can see the shop logo and name, along with additional details (website, location). Under each shop, the list of available sales will unfold, with details about price, sale percent, availability and informative pictures.

The ability to set preferences for the search was the most requested feature in the survey presented in Chapter 2.1. All shown search results must respect all the set preferences.

The user can set an integer value for the minimum price. The minimum value is 0, the maximum 9,999 and the default 0. One will be able to set a maximum price as well. The value must be an integer in the range between 1 and 99,999, with the default value being the maximum value. The value for the maximum price must be strictly greater than the value of the minimum price.

The discount percent of the offers is another filter for the search. The percent will be an integer between 0 and 99, with the default value of 0. All offers shown will have a discount percent of at least the set minimum value.

Filtering by the product category is another option. By default, a toggle will be switched off, meaning this filter is turned off and all the products will appear in the results. If the toggle is switched on, an additional dialog can be opened and one can select all the categories desired from the possible options. There are 21 product categories available: Accessories, Books,

Cinema, Coffee, Cosmetics, Dessert, Electronics, Fashion (Kids), Fashion (Men), Fashion (Women), Food, Games, Gym, House, Jewelries, Market, Optics, Shoes, Sport, Stationery, Toys. These categories will be displayed to the user in alphabetical order.

The last search filter is a toggle that, when is switched on, shows the expired offers as well. By default, it is switched off.

Saving offers as favorites was another highly requested feature, standing at number 2 in the list of most wanted features in the survey. In the results screen, all the offers displayed will have a button attached to save an item as a favorite. If the item was already in the favorites list, the button will be already activated and pressing it will remove the item from the favorite list. Besides this, a separate screen in the app exists that displays all the items saved as favorites. Here, the user will be able to remove items by swiping, pressing the delete button and confirming the action. All the offers in the favorites screen can be sorted by an option in the preference screen. The available options are discount and ascending or descending price. The default sorting criteria is discount in descending order.

The UI and UX follow the Android Design guidelines [12]. The app has a light theme, a primary and a secondary color. UI elements are to be easily distinguishable, with an accent on clickable elements. The layout will be consistent throughout the app, all screens containing the bottom navigation and toolbar. The only exceptions to this rule will be the login and signup screens. Through the bottom navigation bar, the user can access the 4 main screens of the app: Search, Preferences, Favorites, and Profile. The bar will emphasize the current screen the user is in. The toolbar does not change through the app, always consisting of primary color background and the centered ShopSmarter logo. The responsiveness of the app will come from always showing the progress by either loading animations or progress bars. In this way, the user will feel in control and know that the app is loading the wanted content.

The MVP[5] of the app would consist of a tool that can search for discounts in the proximity of the user based on certain preferences. The feature of saving favorite items is not in the MVP because it might not lead to consumers going to shop more in stores. Due to the absence of favorites, authentication would not be necessary. This is very important because it can reduce the cost of development from the smaller need of cloud storage. The intuitive UX must be a major point of the MVP for realizing if the users understand the main flow of the application. The UI may not be as important because it can be easier to improve than the UX.

[5] Minimum viable product

3 RELATED WORK

In this chapter, we present other similar existing applications and compare them to the proposed solution. We will start with apps of shopping centers and retailers, continue with apps based on beacon searching and finish with other alternatives to find discounts.

3.1 Shopping Center Apps

Shopping centers are great venues to shop because of the multitude of stores available. All types of stores are present and the competition is at the highest in this environment. Because of the large number of customers and stores, it is difficult and time-consuming to go through all the available opportunities to seek what is needed. To help both the casual and experienced shopper, some shopping centers offer mobile apps to help visitors find relevant stores and promotions. Two examples of such apps in Romania are AFI Cool [13] and CITY iLOVE by Baneasa [14].

AFI Cool [13] is the official app of AFI Palace Cotroceni, a shopping center from Bucharest, Romania. The app can show information and the location on the map of all the shops in the center. There is a special section for the cinema, where one can see available movies, screenings, and tickets. There are also sections for promotions and events. At the time of the writing, these sections were scarcely updated, containing information about events from 5 months back. Also, there were only 7 promotions presented in the app, a number way too small for the over 180 stores in the center.

CITY iLOVE by Baneasa [14] is the official app of Băneasa Shopping City, a shopping center from Bucharest, Romania. The app is able to show a list of all stores, restaurants, and services. Like AFI Cool, there is the possibility of seeing the cinema schedule and buying tickets. An interesting feature of the app is the ability of scanning bills and promotion codes to win extra discounts in the shopping center. Besides this, there is also the ability to scan the parking ticket and a promotions tab. At the time of the writing, the promotions tab contained only 13 promotions, 3 of which were expired. On top of that, after entering the page of a retailer, a location icon is visible, but it redirects to an empty Play Store page with an "Item not found" disclaimer.

Table 1 - Comparison with shopping center apps

	AFI Cool	*CITY iLOVE*	*ShopSmarter*
Authentication with email	Yes	Yes	Yes
Authentication with social media account	Facebook	Facebook	Google
Show context-based promotions	No	No	Yes
Filters for promotions	No	No	Yes
Save promotions as favorites	No	No	Yes
Good UI/UX	Yes	Yes	Yes
Cinema section	Yes	Yes	No
Bills and promotion codes scanning	No	Yes	No

In Table 1, we break down the features of the two apps presented in this chapter compared with the proposed solution. All three apps offer similar authentication methods and have good UI and UX. Despite that, the main features of ShopSmarter do not appear in the two alternatives, not even in a basic form. Context-based promotions, filters, and favorites do not appear in any of the two implementations. Both have only a minimal implementation of a promotions section, where users can only see a few offers. However, both shopping center apps have a specialized section for buying cinema tickets, while CITY iLove offers scanning of bills and promotions. These features could be useful in the context of the ShopSmarter app and may be the subject to future works.

3.2 Retailer Apps

Retailers are trying to promote their brand to mainly attract new customers, but at the same time to keep the loyal clientele. There are plenty of benefits for loyal customers who purchase from the same store time and time again. Fidelity cards, special discounts, and bonuses are just a few of the tricks to maintain a loyal clientele. Most of the time, these benefits are unlocked through a custom mobile app of the retailer, where users can keep everything in check in their account. We will further present apps of two fashion retailers: H&M and Bershka.

"H&M - we love fashion" [15] is the official app of H&M, a Swedish clothing retailer company. The app shows the full catalog of products and has a separate section for promotions. The promotions can be filtered by product category and other factors like type, color,

measurements or concept. There is a shop locator that shows all the shops in the residence country on a map for seeing which is closer to the current location. Users can save products as favorites and place orders directly within the app. The app has an overall great structure and it is easy for users to find what they are looking for. Other than finding the nearest store, there are no more context-based features.

"Bershka - Fashion and trends online" [16] is the official app of Bershka, Spanish clothing retailer company. Exactly like the H&M app, it can show the full catalog of products and has a section for promotions that can be filtered by color, size, and price. The shop locator searches for nearby shops and displays them ascendingly after the distance from the current location. Favorite products can be saved and there is the option for online shopping as well. An interesting feature is the ability of scanning QR codes[6] to see in-store availability, share or buy a product. Like its counterpart app from H&M, the option for context-based searches does not exist.

Table 2 - Comparison with retailer apps

	H&M	*Bershka*	*ShopSmarter*
Authentication with email	Yes	Yes	Yes
Authentication with social media account	-	Facebook	Google
Show context-based promotions	No	No	Yes
Filters for promotions	Yes	Yes	Yes
Save promotions as favorites	Yes	Yes	Yes
Good UI/UX	Yes	Yes	Yes
Option of purchase within the app	Yes	Yes	No
QR code scanning	No	Yes	No

In Table 2, the breakdown between the two retailer apps and the proposed solution can be found. All three apps check most of the boxes that make a great shopping app. The main difference remains that only ShopSmarter can offer context-based promotions and make use of the beacons placed in the proximity of the user. Both retailers are aware of the power of online shopping and integrated it into the respective apps. The option of scanning products with the Bershka app could come handy, although the use cases are fairly limited.

[6] Quick Response Code = machine-readable label that contains information about an item

3.3 Beacon-based Apps

Beacons can be used in retail to create contextually enhanced shopping experiences for customers through a smartphone app. There is a wide variety of possible use cases, such as proximity adverts, indoor navigation, and analytics. There are also plenty of other scenarios where beacon broadcasting can help both companies and consumers. Environments that can deploy such technology are places like shopping centers and markets, but also museums [17] and libraries [18]. An app that is based on beacon scanning and helps consumers in finding better offers is NearBee [19].

"NearBee - Discover what's buzzing around you" [19] is an Android app that notifies users of existing offers and informational messages broadcast by beacons near the user. It can detect Eddystone [5] campaigns and push notifications for signals coming from Beaconstac [7] beacons. These notifications appear on the lock screen of the smartphone. The app has also the capability to read NFC[8] campaigns and scan QR codes. NFC campaigns work by tapping the smartphone on an NFC tag for receiving notifications. The QR code scanner can display information and actions embedded into scanned codes. At the time of the writing, the QR scanner worked only with codes generated with the Beaconstac online generator. When scanning another type of valid QR code, the app crashes. Another limitation of the app is represented by the strict specifications of the beacons compatible with the app. The documentation states that a beacon must transmit both Eddystone-URL [5] and Eddystone-UID frames for it to be detected by the NearBee app. Also, the URL [9] broadcast in the Eddystone URL frame must be HTTPS[10] [20].

Table 3 - Comparison with beacon based app

	NearBee	*ShopSmarter*
Authentication with email	No	Yes
Authentication with social media account	-	Google
Show context-based promotions	Yes	Yes
Filters for promotions	No	Yes
Save promotions as favorites	Yes	Yes

[7] Platform that offers hardware and software solutions for beacon management
[8] Near Field Communication
[9] Uniform Resource Locator
[10] HyperText Transfer Protocol Secure

Good UI/UX	Yes	Yes
Push notifications	Yes	No
NFC campaign detection	Yes	No
QR code scanning	Yes	No

In Table 3, it can be seen how the proposed solution stacks up against its alternative. The NearBee app does not have any authentication method for the user and has no options for filtering the promotions. Showing context-based promotions is the main feature of both apps. Besides that, the option of saving promotions and an intuitive user interface are both present. The NearBee app edges ShopSmarter with push notifications and other context-based campaign detections (NFC, QR). One main disadvantage of NearBee is the lack of personalization for promotions. These are composed only of a URL, displaying additional information being available only in the browser after entering the URL. NFC detection and QR scanning are interesting features that may be considered for future works, as these can improve the contextual-based experience granted by the app. The main downside of these approaches is the extra cost related to buying the NFC keycards and QR code stickers.

3.4 Other Alternatives

The United States is one of the major retail markets in the world. The total revenue generated in retail in 2017 reached 3.1 billion US dollars only from offline shopping [2]. American consumers still tend to make more purchases in stores. 64% of the budget for shopping is spent in stores [2]. For this market, there are custom solutions tailored for the American audience. We further present and compare two shopping apps that are available only in the United States: Groupon [21] and Slickdeals [22].

"Groupon - Shop Deals, Discounts & Coupons" [21] gathers deals and discounts from shops in cities from the United States. The deals can be used immediately after they are redeemed. All the selected vouchers can be tracked by location and expiry date. There is also a section where one can search for deals near the current location and receive notification for nearby deals. The app has also a social aspect because discounts can be shared with friends.

Slickdeals [22] hosts a deal-sharing community, where the deals are shared by existing users on a forum. The deals are voted and the best ones reach "Popular Deals" or "Frontpage" forums. There are editors that ensure the reliability of promotions posted. The app lets the user set alerts for future deals and the alerts are configurable by category, store, and

keywords. A shortcoming of the app is the nature of deal finding. Once a great deal appears, it might end before it reaches the masses on the popular tabs.

Table 4 - Comparison with other alternatives

	Groupon	*Slickdeals*	*ShopSmarter*
Authentication with email	Yes	Yes	Yes
Authentication with social media account	Google, Facebook	-	Google
Show context-based promotions	Yes	No	Yes
Filters for promotions	Yes	Yes	Yes
Save promotions as favorites	Yes	Yes	Yes
Good UI/UX	Yes	Yes	Yes
Push notification	Yes	Yes	No
Community based	No	Yes	No

In Table 4, the breakdown between the two alternatives and the proposed solution can be found. All the apps can save promotions and have several filters available. The main advantage of the proposed solution is the ability to present context-based promotions using the beacons placed in proximity. Groupon offers a similar search where one can see a map with all discounts. Slickdeals has a very different approach than all the apps presented in this chapter. Letting the community find and share the discounts is profitable, as long as there is a large enough community.

4 TECHNOLOGIES

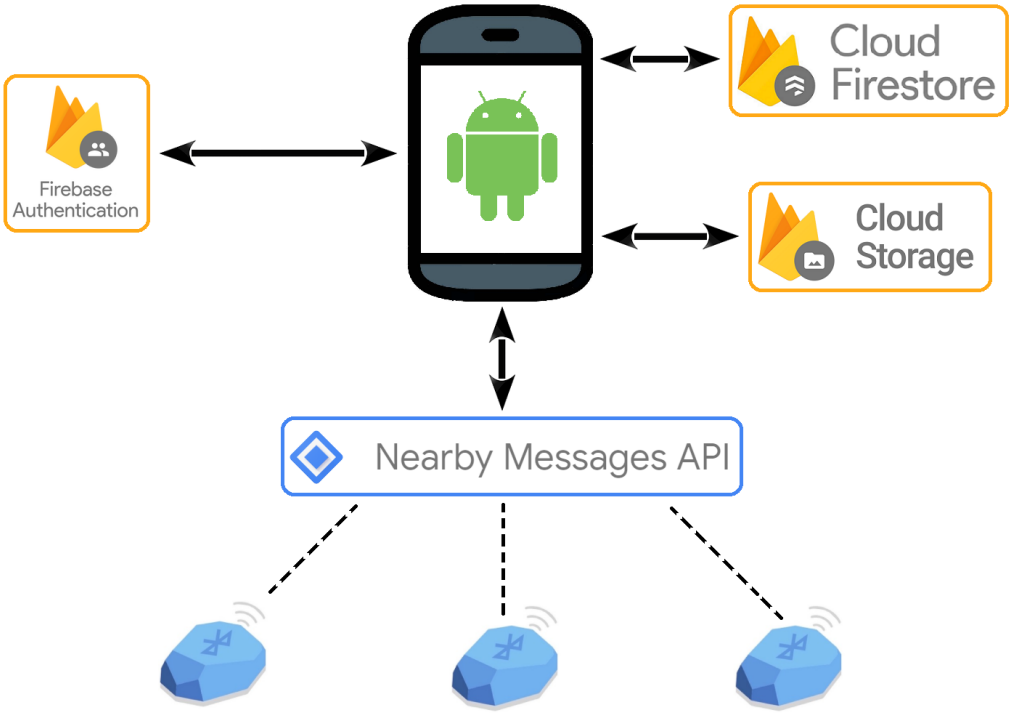

Figure 9. The general outline of the technology stack

In this chapter, we present the main technologies used in the implementation and how they work within the app context. A general outline of these technologies and how they interact with each other is presented in Figure 9. For each technology we will present alternative options that could have been used instead, comparing them to justify the selections made.

4.1 Android

Android is an operating system developed by Google and used by mobile devices. The operating system is based on a Linux kernel and adds multiple other features, like wakelocks, low-memory killer, logger and many others [23]. A main advantage of choosing Android as the platform to develop a mobile app is the market share in terms of units sold and apps. Android is the most used operating system for mobile devices in today's market. In terms of market share, as of 2018, 88% of all sold smartphones are powered by Android [24]. Google Play Store, the official app market for Android devices, has 2.6 million apps available [25]. Another advantage is the process of publishing an app, as any developer can easily upload

apps to the Google Play Store. The only prerequisite is paying a one-time fee before the publishing of the first app.

The main competitor of Android is iOS, the mobile operating system developed by Apple. It is made exclusively for Apple hardware and it is the second most popular operating system for mobile devices, after Android. In 2018, Apple sold over 200 million iOS smartphones [25], an impressive number, but still 6 times less than Android. App Store, the official app market for iOS devices, has 2 million apps available [25]. The number of apps is 23% smaller than its counterpart from Android, the Google Play Store. The smaller market share was one of the reasons Android was preferred as the platform of choice for the solution proposed.

Another reason is the increased costs. iOS apps are developed using Xcode, an IDE[11] available only on macOS, the operating system for Apple's Mac line of computers. There are no other alternatives for developing iOS apps on a Windows computer other than virtualization, renting or building a Mac computer. Google, on the other hand, offers Android Studio, the official IDE for Android app development, available on Windows, macOS, and Linux.

4.2 Google Beacon Platform

Google Beacon Platform was designed by Google for context-based applications. It can enable apps to react to the beacons in the environment and it has three components: Eddystone, Proximity Beacon API[12] and Nearby API [26].

The beacons are essential for the Google Beacon Platform. These transmitters are able to broadcast information to nearby devices and allow context-based use cases. Eddystone is the protocol created by Google for BLE[13] messages transmitted by beacons. The Eddystone format can be detected by both Android and iOS devices. The frame format can include several types of payloads, such as Eddystone-UID, Eddystone-URL, Eddystone-TLM, and Eddystone-EID [5]. The proposed solution uses only Eddystone-UID payloads. This type of payload only consists of a unique ID formed of a 10-byte namespace ID and a 6-byte instance ID [5]. It can be formatted as a code of 32 hexadecimal digits, where the first 20 digits represent the namespace ID and the other 12 digits the instance ID.

The Nearby platform is able to discover other devices in the proximity of the user and establish connections with them. It consists of 3 components: Nearby Connections API,

[11] Integrated development environment
[12] API stands for Application Programming Interface
[13] Bluetooth Low Energy

Nearby Messages API and Fast Pair [27]. Out of these 3, only Nearby Messages API was used in the proposed solution. In the early drafts of this paper, another component was also considered to be used in the solution - Nearby Notifications. This feature was supposed to help users discover interesting things to do around them. It was meant to push useful notifications and there was no need to install an app for these searches made in the background. The notifications could trigger app intents or provide URLs to open in a browser. But due to "a significant increase in locally irrelevant and spammy notifications that were leading to a poor user experience", Google decided to close Nearby Notifications in December 2018 [28]. At the time of the writing, there is no alternative to Nearby Notifications in the Google Nearby ecosystem. Additionally, Google warns developers to use Nearby features sparingly and only after an action is invoked by the user [29]. These cautions are meant to prevent a high battery drain.

The Nearby Messages API enables the passing of small payloads between Internet-connected mobile devices. For the communication, it uses a mix of Bluetooth, Bluetooth Low Energy, Wi-Fi and ultrasonic audio [30]. Devices send tokens to the Nearby Messages server, as it can be seen in Figure 10. The server tracks the messages of publishers and tokens of active subscribers. When either the publisher or subscriber detects another device, the server facilitates the message exchange.

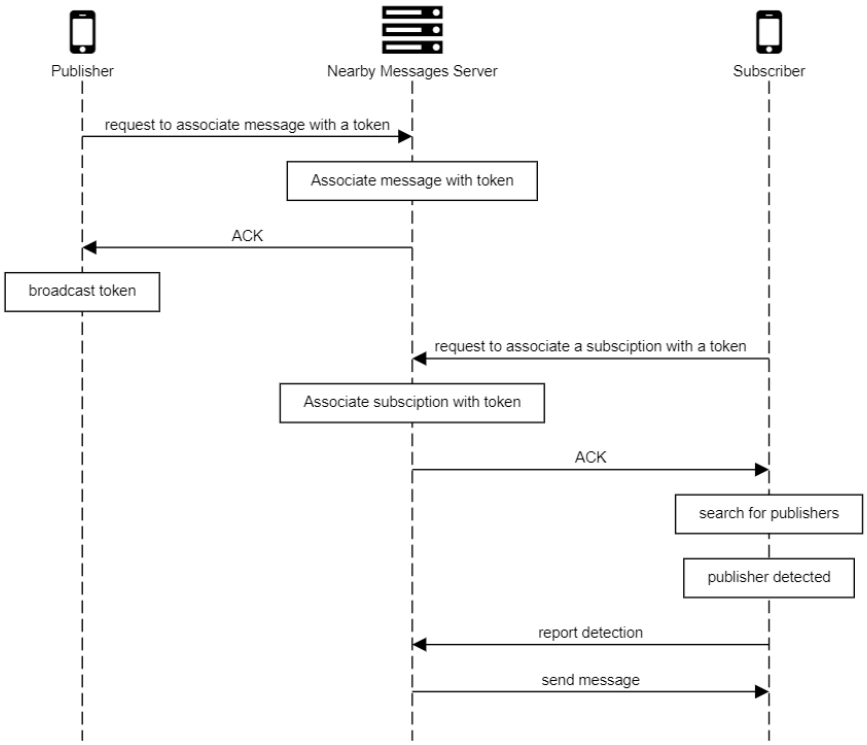

Figure 10. The sequence diagram of a general Nearby Messages communication

Android Beacon Library [31] is an open-source library for beacon interactions. It can notify apps when beacons appear and disappear in the range of the phone. The library also has features of sending updates about the range of the beacons and allowing smartphones to send beacon transmissions. By default, the library detects beacons corresponding with the AltBeacon standard[14], but Eddystone is also fully supported [31]. Android Beacon Library is considered the best alternative to Google Nearby.

Nearby Messages API was preferred for the proposed solution because of its design and ability to resolve high-level beacon data. When a beacon is discovered, the API returns the attachments of that beacon from the cloud, with no need for low-level processing. The quotas of the API are more than sufficient for the scope of the paper and even for a limited release.

4.3 Firebase

Firebase is a mobile and web application development platform owned by Google. It consists of 17 products [32] that provide tools for developing and analyzing mobile and web apps. Firebase has 3 pricing options: free, fixed and "pay as you go" [33]. For the implementation of the proposed solution, the free pricing plan was used because it contains all the needed tools for developing the app. In the proposed solution, we used 3 of the platform products: Authentication, Cloud Firestore, and Cloud Storage.

Firebase Authentication is an authentication service where developers have to implement only the client side. In the ShopSmarter app, the service is implemented with 2 options for sign-in: email address with password or Google account. In the email and password based authentication, the user must verify the email address by entering a link sent to the provided email.

Cloud Firestore is a NoSQL[15] cloud-hosted database designed to work cross-platform through its multitude of SDKs[16]. Data is stored in documents formed of field to value mappings. There are several data types available, such as strings, numbers, timestamps, arrays, and maps. The documents are stored in containers called collections for a better-structured view of the data. In the ShopSmarter app context, the database stores all the information about shops, sales, and favorites.

[14] A protocol for beacon advertisement messages
[15] Not Only SQL (refers to a non relational database)
[16] Software development kit

Cloud Storage is the service that provides file uploads and downloads for Firebase apps. Any objects, such as image, audio or video files, can be stored there. This service is used to store the logos of all the shops and sales used in the ShopSmarter app.

Table 5 - Comparison between Firebase and Amazon Web Services

	Firebase	*Amazon Web Services*
Collaboration	closed-source	closed-source
Hosting	cloud	cloud, dedicated servers
Free tier	Yes	Yes
Authentications / month	10,000	50,000
Database - Stored GB	1	25
Storage - Stored GB	5	5
Storage - Requests / day	70,000	22,000

An alternative to Firebase is Amazon Web Services (AWS). It has more than 90 services available [34], including services for authentication, database, and storage. The authentication service from AWS is called Amazon Cognito and the free tier permits 50,000 monthly active users [35]. Amazon DynamoDB is a serverless NoSQL database and Amazon S3 is a service for object storage. Both are considered counterparts of Cloud Firestore and Cloud Storage from Firebase. The details for the free tier of both Firebase and AWS are presented in Table 5. The biggest disadvantage of AWS is the steep learning curve, especially because of the high number of available services. Firebase has the benefit of more advanced technology and a simple, yet powerful set of services that are easier to deploy.

5 PROPOSED SOLUTION

5.1 Architecture

From a hardware standpoint, the architecture of the proposed solution includes two main components. As has been previously shown in Figure 9, these 2 elements are the Android device and the beacons.

The Android device is the central component of the architecture. It must have the ShopsSmarter app installed to enable all the features presented. There are also special requirements for the Android device for the app to work.

The app was developed for devices with Android API 21 or higher. This means that the ShopSmarter app is available for all Android devices with Android Lollipop 5.0 or newer. As of May 2019, that covers more than 89% of all Android devices [36].

One of the technologies used by the Nearby Messages API is Bluetooth Low Energy. BLE provides a more reduced battery consumption than the classic Bluetooth. Android 4.3 (API level 18) included full support for BLE [37]. Also, for the API to work, Google Play services 7.8.0 or higher are needed [38]. This does not constitute a problem for the potential users, as all devices running Android 2.3 or higher automatically update the Google Play services.

In conclusion, the architectural requirements of the Android device are:

- Android Lollipop 5.0 or newer
- Google Play services 7.8.0 or higher
- A working Internet connection (WiFi, mobile data).

For the purpose of this paper, the beacons were simulated by an Android smartphone using the Beacon Simulator app [4]. The simulator app can broadcast Eddystone-UID frames needed for the ShopSmarter app. This type of frame consists of a unique ID formed of a 10-byte namespace ID and a 6-byte instance ID [5]. For every shop used in the testing and simulation of the proposed solution, a beacon was created in the Beacon Simulator app. The beacon was then registered to the ShopSmarter project and had the shop ID attached. The shop ID is a 20-characters case-sensitive string used to uniquely identify the shop in the database. After registering the beacon and attaching the shop ID, for as long as the beacon is enabled and in the range of the user, the proposed solution finds the shop and can load the available offers.

For beacons used in commercial releases, Google works with beacon providers to help them release hardware devices that are compatible with the Google Beacon Platform and the Eddystone format [5]. At the time of the writing, Google features a list of 18 manufacturers that provide beacons with full support for Eddystone [5]. There is a wide variety of beacon hardware available, with ranges of up to 500 meters, battery lives of up to 10 years and prices starting from 15 US dollars.

In the ShopSmarter app, the Nearby Messages API is used for determining what shops are in the proximity of the user. Each shop would need to have one or more beacons placed inside and/or outside the shop. These beacons must be powered on and set up to emit Eddystone-UID payloads. The attachments are unique for each store and they are saved in the cloud. The beacon itself emits only the Eddystone-UID payload. In order to link an attachment to a beacon, the latter must be registered, which is going to be presented in detail in Section 5.3.1. After the attachment is added, the beacons are ready to be used.

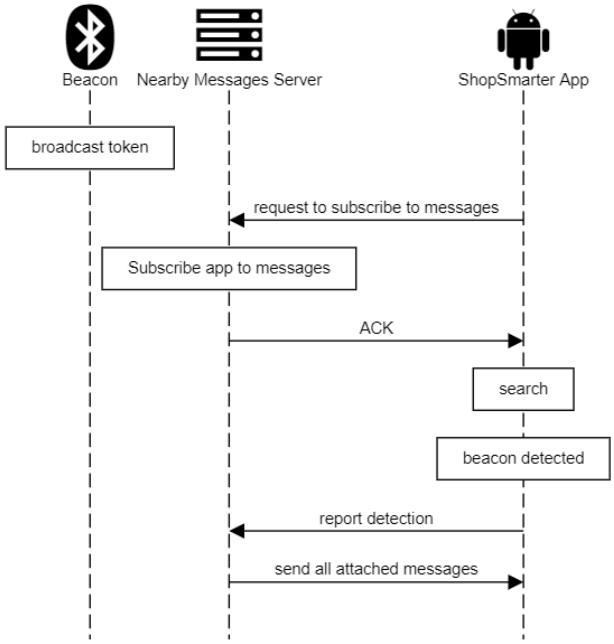

Figure 11. The sequence diagram of communication between a beacon[17] and ShopSmarter

The ShopSmarter app sends a request to the Nearby Messages API and subscribes a listener able to process found messages, as it can be seen in Figure 11. The Android phone is scanning the environment for nearby beacons. When a beacon is found, the Eddystone-UID broadcast by the beacon is sent to the Nearby Messages server. If the found beacon is registered, all the

[17] It is assumed that the beacon presented in the diagram is registered and emits an Eddystone-UID payload

attachments linked to that beacon in the cloud are sent to the Android client. These attachments reach the app as separate messages.

All the Google services used for developing the proposed solution (Firebase, Nearby Messages) do not need us to provision any hardware to use them. These services are provided as BaaS[18] and just need to be integrated with our application codebase, without the need for any servers or additional hardware.

5.2 Application Flow

In this section, the features and possible flows of the application are presented from the perspective of the end user. There are 8 scenarios presented: sign in with email and password, login with email and password, search, add favorite, see saved favorites, remove saved favorite, set preferences.

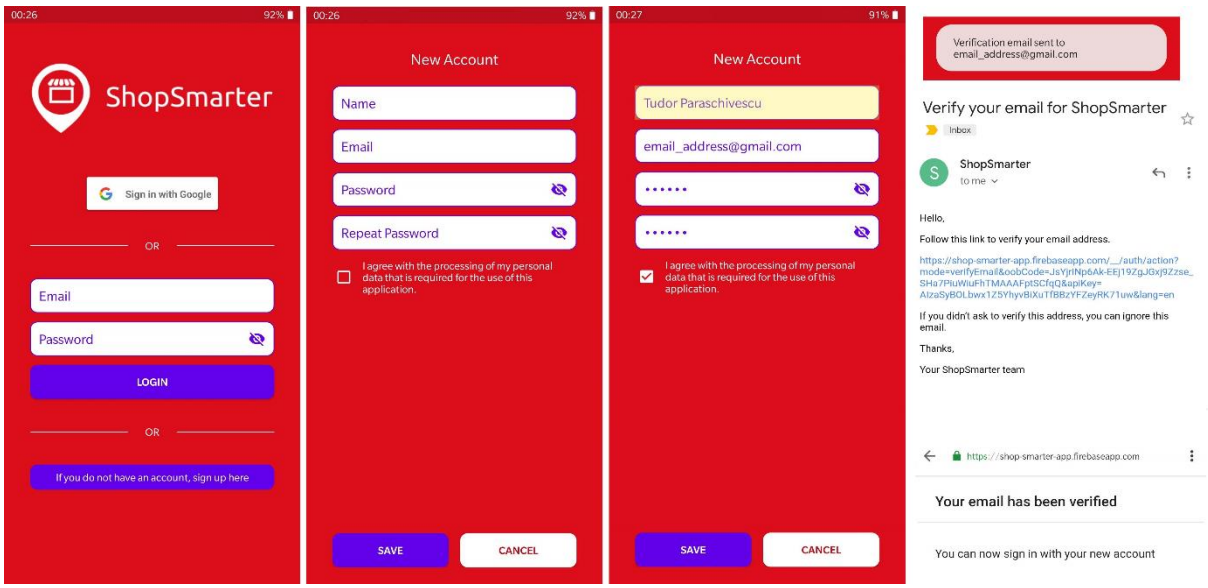

Figure 12. Application flow for sign up with email and password

In Figure 12, the flow for signing up with an email and password can be seen. The first screen that the users see when entering the application is the login screen. Here the user can log in the application with an already existing account. If that is not the case, a new account must be created. Pressing the button at the bottom of the screen opens the second screen from

[18] Backend as a Service

Figure 12. In this screen, the user must complete the name, email address, and password. The password must be written twice, to ensure that is written correctly. There is also a checkbox at the end of the form that must be checked if the user gives consent for storing their personal data (name, email). The account cannot be created without checking this box. After completing all the fields in the form, as seen in the example from the third screen in Figure 12, the Save button must be pressed. The app notifies the user if any problems occurred during the processing of the request. If no problems occur, a Toast[19] will show a message that a verification email was sent to the provided address. The format of the verification email sent to end users can be seen in the last screen of Figure 12. After clicking the link, a confirmation message will be provided and the user can log into the app.

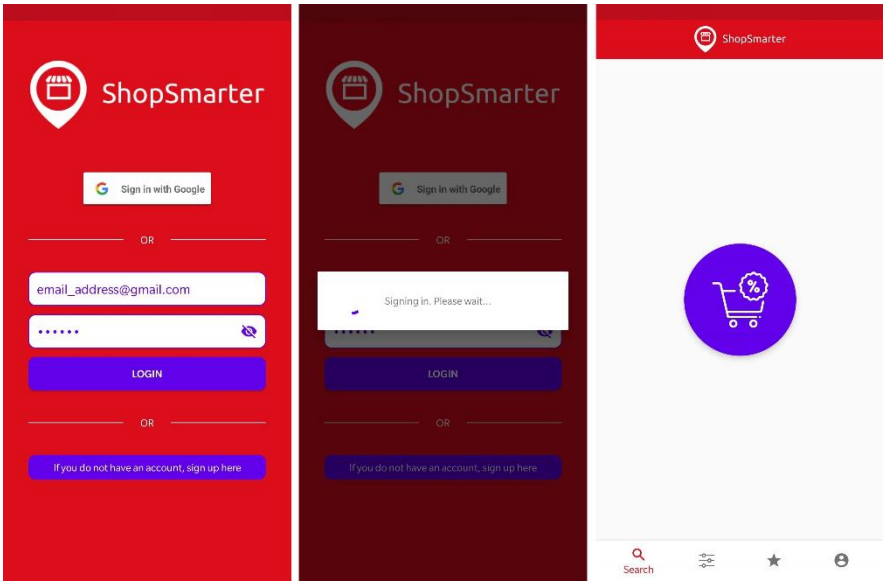

Figure 13. Application flow for login with email and password

After creating an account, the user can log in with the email and password. In the first screen presented in Figure 13, in the middle of the screen there are the 2 fields for email and password that must be completed. The password field has the option to make the password visible to check for mistakes. After pressing the "Login" button, a progress bar will be shown until the request is completed. If successfully, the user is logged in the application and the search screen appears, as it can be seen in the last screen from Figure 13. If not, an error message will be displayed through a Toast. The user must have the email address verified before logging in.

[19] Small popup that gives feedback to the user

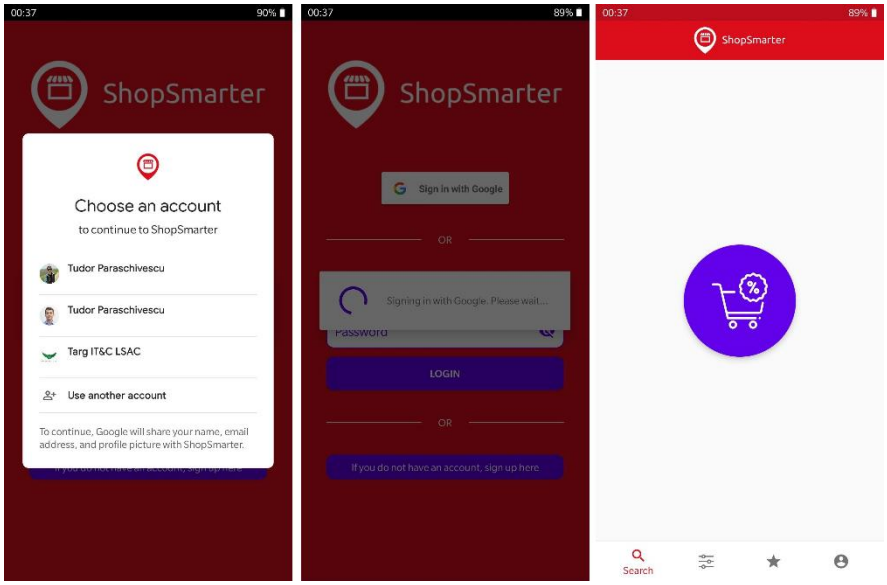

Figure 14. Application flow for signing in with a Google Account

The ShopSmarter app offers a second option for signing in, through a Google account. In the upper part of the login screen, there is a button for this option of sign in. As it can be seen in the first screen from Figure 14, after pressing the button, the app will prompt the user to choose the account to continue the sign in. The app displays all the Google accounts that are available on the Android device used. Alternatively, another Google account can be used by choosing the "Use another account" option. This last option would need the user to complete the email address and password associated with the wanted Google account. If the user chooses one of the available accounts, there is no need for reentering the password for the Google account. After selecting the account, a progress bar will be shown until the request is completed. If successful, the search screen will appear and the user will be logged in the application. If not, a Toast will display the encountered error. Both sign-in options require a working Internet connection.

After creating an account with the email and password sign-in method using a Google account email address, the user can also sign in the app using the Google sign-in method. There will be only one account created with 2 sign in providers available. This is not available by doing it the other way around, signing in with the Google option in the first place. When trying to create an account with the email and password method with an already used Google account, an error message will appear stating that the email is already in use by another account.

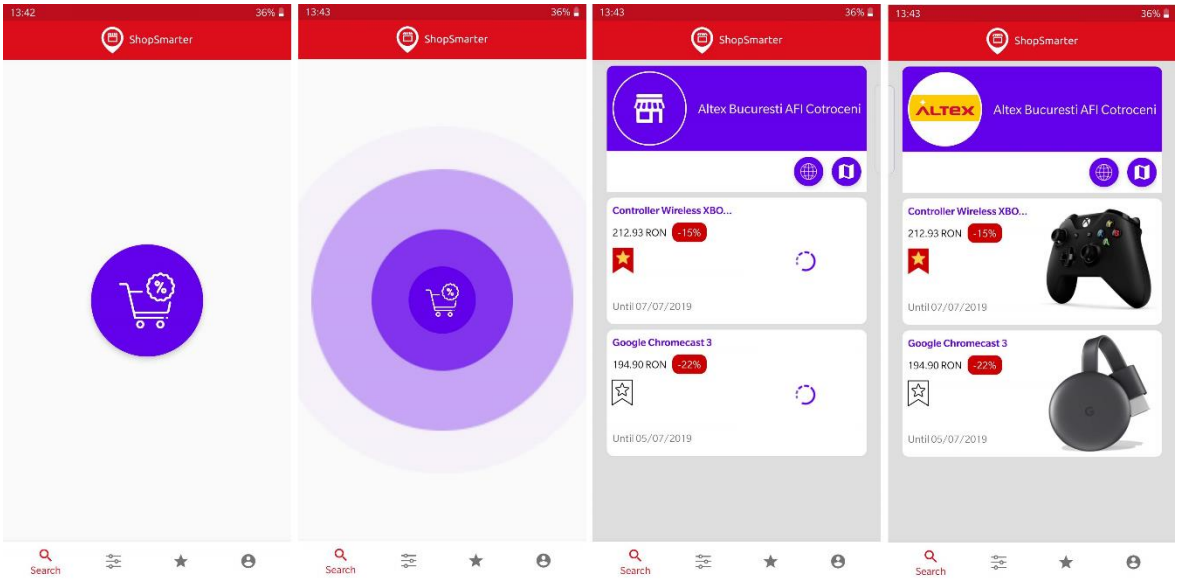

Figure 15. Application flow for search

The search for available offers starts in the search tab. In this tab, there is one round button in the middle of the screen. Pressing the button starts the search for sales in the proximity of the user and activates an animation with ripple effects as seen in Figure 15. The app searches for all the beacons in the range of the device in order to show the corresponding shop and offers. After 8 seconds, the search stops and all the found results are shown. If there were no beacons detected, a message will appear in the middle of the screen explaining that there were no shops found in the area. All the shops and offers will appear instantly after the search ends. The images take longer to load, so for a short time, a placeholder will be displayed instead. The shop details are displayed in the upper part of the screen, including logo, name, website, and location. Underneath, all the available offers are shown, with details like name, price, sale percentage, image, and availability. Vertical scroll goes through all the offers of a specific shop and horizontal scrolling goes through all the shops found.

Prior to starting the search, when the user presses the button, the app checks if all the needed prerequisites are met. For detecting the beacons, Bluetooth and Location access must be enabled. Also, a stable Internet connection is needed for downloading details about shops and offers.

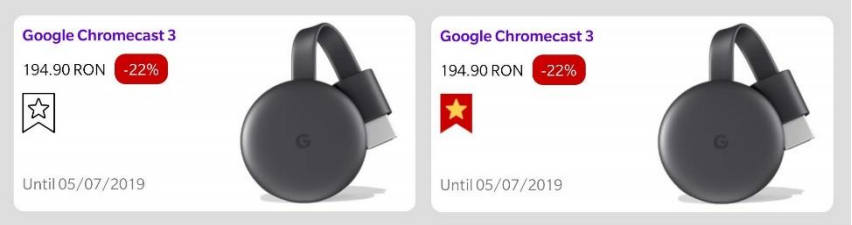

Figure 16. Display of a product before and after saving it to the favorites

All the products that are displayed after the search have an icon with a starred ribbon. That marks if the respective offer is saved as a favorite or not. If the icon is black and white, it means that the product is not saved as a favorite, while a red and yellow ribbon means that it is. Pressing the icon switches the favorite state and updates the UI as in Figure 16.

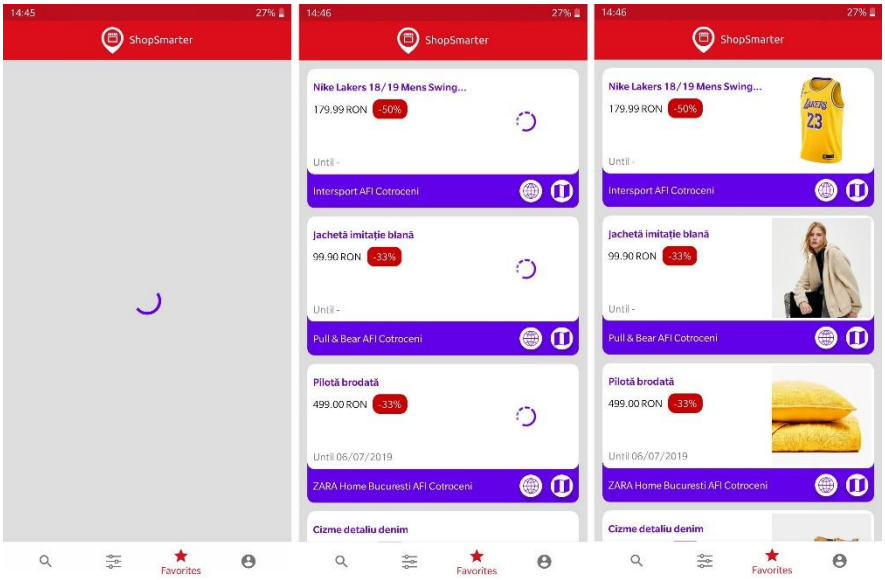

Figure 17. Loading favorite offers

In Figure 17 the process of loading the favorite offers is depicted. In the bottom navigation bar, there is a button for favorites. Tapping it starts loading the saved favorites. After the offers appear, it might take longer for images to load. While this happens, placeholders will be displayed instead. The list with the saved offers is vertically scrollable in order to have enough space for all the offers. In this screen, besides looking at the offers saved and their details, there is also information displayed about the stores that run the save offers. The saved offers can be removed by swiping the corresponding card to the left or to the right. When doing this, a red "Remove" button will appear, like the one in the first screen from Figure 18. By pressing this button and confirming the delete in the next popup, the sale will be removed from the user favorites.

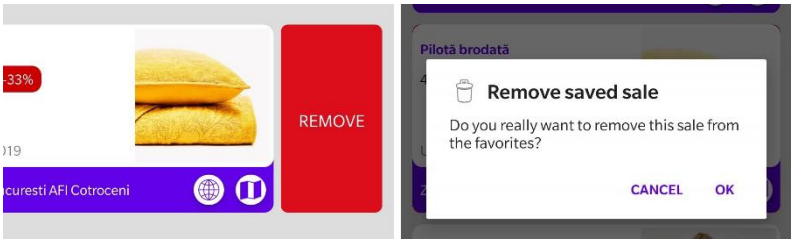

Figure 18. Removing a favorite offer

The user has the ability to set certain preferences for the search. This can be done in the Preferences screen, seen in Figure 19. The user can set values for the minimum price, maximum price and minimum sale percentage of the products that will come up in the search results. Also, by default, products from all the categories are shown in the search results. By switching on the "Do not show all categories" field, only products from selected categories will appear in the search results. The categories can be chosen by pressing the "Categories " button. A list with all 21 categories is displayed and the user can choose all the wanted groups. The design of the list can be seen on the last screen from Figure 19. There is also an option for showing expired sales as well. This option is unchecked by default. If setting multiple preferences, all of them must be met for a product to be displayed in the search results. If a shop was found but has no offers that meet the preferences, it will still appear in the results list with an empty list of offers.

In the Preferences screen, there is a special section for Favorites. There, the user can choose how the sorting of the saved sales is made. The saved sales can be sorted by discount, ascending price or descending price.

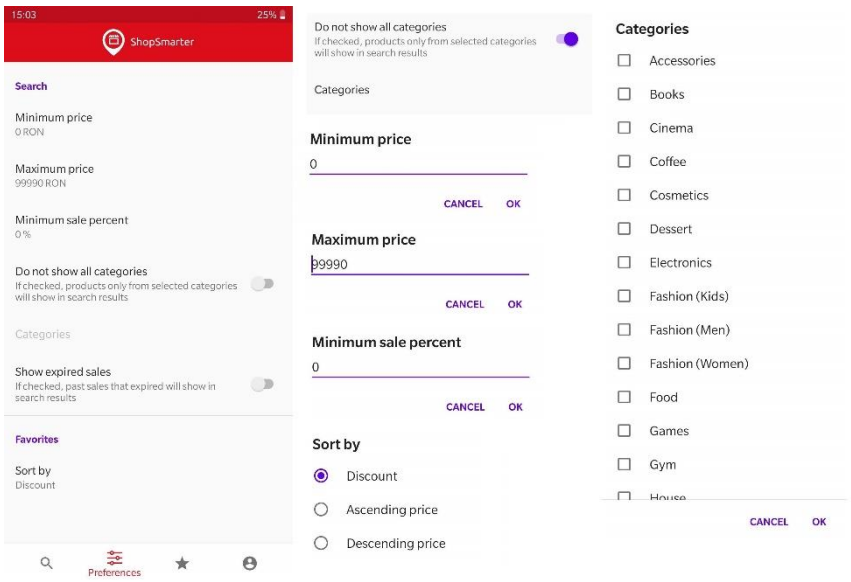

Figure 19. Available preferences and change dialogs

5.3 Implementation Details

5.3.1 Setting Up a Beacon

For the ShopSmarter app to work properly, beacons for the stores must be provided. For the purpose of this paper, the beacons were simulated by an Android smartphone using the

Beacon Simulator app. The process of setting up a beacon has three steps: creating the beacon, registering it and associating an attachment.

The creation of the beacon is done with the Beacon Simulator app on the secondary smartphone used for beacon broadcasting. The type of frame used is Eddystone-UID. To generate a new beacon, the type of the frame must be chosen. As can be seen in Figure 20, the app offers 6 possible frames. After choosing the Eddystone UID option, we are redirected to a screen with the beacon info. Here, we can set up a name that will be displayed in the interface of Beacon Simulator. More importantly, we can set the values that will be broadcast in the Eddystone UID frame: the namespace and instance IDs. We must ensure that the two IDs together uniquely describe the beacon within our network of registered beacons. Two beacons with the same identifiers could not be associated with attachments for two different stores. Other fields that can be set are Tx Power, Transmission Power, and Frequency Mode. Tx Power indicates what the signal power should be when at 1 meter away from the beacon. In general, this field is used to estimate distances. Transmission Power determines how powerful is the signal transmitted by the beacon, while Frequency Mode specifies how often the beacon broadcasts the signal.

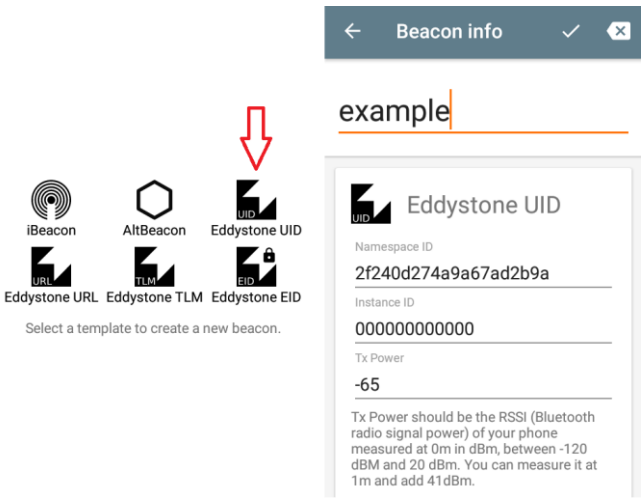

Figure 20. Creating an Eddystone-UID beacon using the Beacon Simulator app

After creating and enabling the beacon for broadcast, the ShopSmarter app would not be able to find the beacon. The app uses the Nearby Messages API for beacon scans, which is searching only for signals from registered beacons in the Bluetooth range of the device. The beacon must be registered in the Google project associated with the ShopSmarter app. This was done using Google's Beacon Tools app [39]. The app has 2 sections, for unregistered and registered beacons, as can be seen in Figure 21. In the unregistered section, we select the beacon we want to register and the beacon info screen appears (second screen in Figure 21). Automatically, the provisioning starts, which is the first step in registering the beacon. At the

time of writing, using the last version of Beacon Tools app, which was version 2.0.178617547, this step never completed successfully and the option for binding attachments never appeared. A workaround we found was to tap the Description option, write the name of the shop and press the tick icon. After doing so, the beacon appears in the registered section and the option to add attachments becomes available, as seen in the second screen from Figure 21. This would conclude the registering step.

For adding an attachment, we use the Beacon Tools app as well. Tapping the Plus button in the Attachments section brings us to the last screen from Figure 21. There we can specify the type of attachment and the data. The type is composed of the namespace ("shop-smarter-app") and type of the data ("shop-id"). The type is common for all the registered beacons. The data corresponds to the 20-characters string that uniquely identifies the shop in the database. Tapping the tick button from the top right corner saves the attachment and concludes the setup of the beacon.

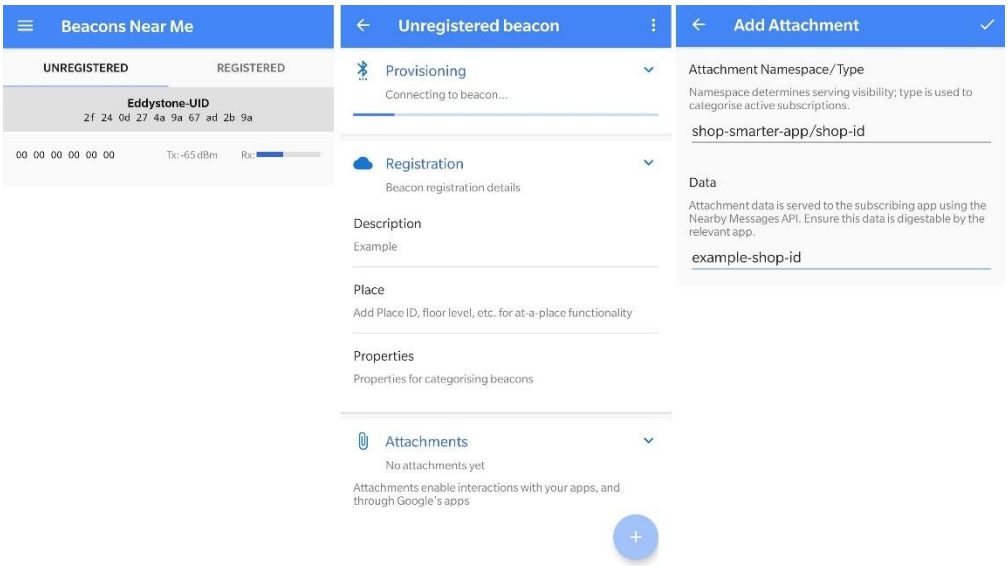

Figure 21. Registering and adding an attachment in Google's Beacon Tools app

5.3.2 Communication with Beacons

For receiving the messages attached to all the nearby beacons, the Beacons API was created. It consists of one public method that starts the beacon search, processes the received messages and proceeds to display the results. In the next snippet, all the fields and methods of the Beacons class can be seen.

```
1   public class Beacons {
2       private static final int TTL_IN_SECONDS = 8;
3       private static Strategy scanStrategy;
4       private static MessageListener messageListener;
5       private static PreferencesHelper preferencesHelper;
6       private static SubscribeCallback scanCallback(MainActivity activity);
7       public static void getBeaconsMessages(MainActivity activity);
8   }
```

The time needed to find all the nearby beacons can vary vastly and depends on the environment, existing interferences and used equipment. A short TTL[20] makes the app more responsive, as the users have to wait less to finish the search. The downside is that a small TTL value could lead to not finding all the beacons in the user proximity. After running several tests and fine-tuning, the final and best value of *TTL_IN_SECONDS* constant was 8.

The scanning strategy prioritizes the scanning. By default, the strategy is to broadcast a pairing code and scan for other devices as well. Also, the strategy sets the TTL for the scan and allows message exchanges over any distance.

The message listener defines what is happening with a message when it is received. The content of the message represents the ID of the shop found and is stored as a String. Immediately after receiving the message, a request is made to the database to download all the data and sales of the shop found. In this way, the user can have the majority of the found shops already loaded when the scanning ends.

The PreferencesHelper instance is used to take into account the preferences set by the user. This variable takes all the values set in the Preferences tab and checks if each offer found respects all the requirements (price, sale percent, category, expiry).

The *scanCallback* method generates a SubscribeCallback instance that is called when the subscription to messages from Nearby Messages API expires. The method redirects the user to the results screen, where all found shops and offers will be displayed.

The *getBeaconsMessages* method starts by clearing the cache used in the database for storing shops data. A client for Nearby Messages is created for the current context and the message listener is subscribed to it. Options for the callback, scan strategy and TTL are made when subscribing. After successfully subscribing, the scanning has started and will automatically stop after the TTL expires. All the found messages will be stored in a cache in the database and will be used in the results screen.

[20] Time to live

5.3.3 Permissions

When using the proposed solution, the user needs access to a stable and working Internet connection. Either by Wi-Fi or mobile data, the user needs the Internet connection for retrieving beacon attachments or favorite offers. When the user tries to use the search functionality, the app prompts the user to turn on Bluetooth, which is needed in the beacon scanning. Additionally, for devices with Android 6.0 or newer, Location must be enabled to discover messages attached to BLE beacons [40]. The location is not automatically enabled, so ShopSmarter requests the user to enable access.

Google states in their documentation [30] that Nearby uses a mix of Bluetooth, BLE, Wi-Fi and ultrasonic audio for communication between devices. As explained in Section 5.3.2, there are several strategies that can be used when a client is subscribing for messages. The default strategy tries to discover nearby devices while also broadcasting a token that other devices can find. With this strategy, during testing, the speaker of the smartphone emits a low ultrasonic sound. We discovered that this sound was used only for broadcasting the device pairing code. The ShopSmarter app uses a strategy that only aims to discover other devices. With this strategy, the sound disappeared, confirming that the near-ultrasonic audio is used only in token broadcasting.

5.3.4 Authentication

For user authentication, the Auth API was created. It consists of methods for setup, getters and user checks, account creation, log in with both the Google and email-password options, sign out and sending of verification email. In the next snippet, all the fields and methods of the Auth class can be seen.

```
1  public class Auth {
2      private static FirebaseAuth mAuth;
3      private static GoogleSignInClient gsClient;
4
5      public static void setup(Context context);
6      public static boolean isUserSignedIn();
7      public static FirebaseUser getCurrentUser();
8      public static boolean isUserVerified();
9      public static void signInWithGoogle(Activity activity);
10     public static void signInWithEmailAndPassword(LoginActivity activity,
                                                     String email, String password);
11     public static void handleSignInWithGoogle(LoginActivity activity,
                                                  Intent data);
```

```
12      public static void createAccount(SignUpActivity activity,
                                         String name, String email, String password);
13      public static void signOut(MainActivity activity);
14
15      private static void sendVerificationEmail(SignUpActivity activity,
                                                  String name);
16  }
```

The implementation needs two fields: *mAuth* and *gsClient*. *mAuth* represents the Singleton Instance of FirebaseAuth and is used in all the calls that have an effect on the state of the Firebase user. *gsClient* is needed specifically for the Google sign-in and sign-out. Both are initialized in the *setup* method. For the initialization of the GoogleSignInClient, the app context and web client ID of the ShopSmarter Google project are needed.

5.3.5 Database

All the app data is stored in Cloud Firestore, a NoSQL database. The database is structured in collections that contain documents. There are 3 collections: *sales*, *saved_sales,* and *shops*. The *sales* collection stores all the sales of all the shops, while the *shops* collection keeps all the details of the shops. The *saved_sales* collection keeps all the favorite sales for each user. The data models of the documents from each collection can be seen in Figure 22.

Figure 22. Data models[21] of the documents from each of the three collections

The Shop model, presented in Figure 22, stores a *sales* array. The datatype of the elements in the array is a reference that points to a Sale element from the *sales* collection. This way, each

[21] Each model contains the name of the fields and the datatypes in parenthesis. The names use the Cloud Firestore naming conventions, where there are 9 data types: string, number, boolean, map, array, null, timestamp, geopoint, reference.

sale can be reached and fetched instantly. The rest of the Shop's fields are strings. The *location* field must be a URL to the Google Maps location of the corresponding shop. The *logo* field is also an URL to the file location in Cloud Storage where the image is stored. The *website* field must be a valid URL that can be opened in a mobile browser.

The Sale model can be seen in Figure 22. The *available_until* field is the timestamp when the offer expires. This field can also be null, meaning that there is no expiry date. The *categories* array stores the IDs of all the product groups in which the product fits. There are 21 categories and each has an ID in the [1, 21] range, corresponding to the alphabetical order. The *id* field keeps the auto-generated id of the document. The *image* field is an URL to the Cloud Storage location of the image. *Price* must be a float number and *sale_percent* an integer.

The SavedSale model keeps only an array of strings corresponding to (sale, shop) tuples. This string is the concatenation of the sale ID, a "-" sign and the shop ID. Each SavedSale document corresponds to one user. The ID of the document is equal with the User UID[22].

The data about shops and sales was manually added in the database for testing. In a commercial release, there would be the need for a method (e.g., web application) where the partnering retailers could manually add, edit and remove the available promotions. This is a subject that could be considered for future works.

```
1  public class Database {
2      public static void setup();
3      public static void getShop(String shopId, PreferencesHelper helper);
4      public static boolean shopsFound();
5      public static void clearShopsData();
6      public static ShopAdapter getShopAdapter();
7      public static void loadSavedSales(SavedSalesAdapter adapter,
                                         View emptyFavoritesView,
                                         View loadingView);
8      public static void removeSavedSale(String saleAndShopId);
9      public static void addSavedSale(String saleAndShopId);
10     public static Set<String> getSavedSales();
11 }
```

The Database API can be seen in the above snippet. Aside from the *setup* method, there are 2 classes of methods, for shop fetching and saved sales handling. The *setup* method gets the Singleton Instance of Firestore and initializes the data structures.

[22] Unique Identifier

From the shop fetching methods, the getShop method is the most important. The method fetches the shop with the ID given as parameter and all the available offers that respect the preferences given as the other parameter. The data about shop and sales is added to a synchronized list that is used in the adapter that displays the results. This list must have the shop as the first element and the sales, if any, next in the list. Only sales accepted by the PreferencesHelper will be added to the list. The data fetching for all found shops is done asynchronously. Because of that, before starting fetching data, the cached shops must be cleared with the *clearShopsData* method. The *getShopAdapter* method returns the adapter used by the custom view that shows the results of the search.

There are several methods used for handling saved sales. The *loadSavedSales* method loads the favorite offers of the user and caches them. The method is called when entering the app because the favorites are needed in the search and in the Favorites screen. All the parameters are used only in the second scenario to display progress using the Views and the favorites using the adapter. The *removeSavedSale* and *addSavedSale* methods receive a string corresponding to a (sale, shop) tuple and do the action both in the cache and in the database.

5.3.6 Deployment

The proposed solution was developed and deployed with Android Studio. The IDE helps with building and running the application. The final format that reaches the Android device is the APK[23]. This package can be deployed and distributed to all Android devices that meet the minimum requirements.

Gradle is an open-source tool for building, automating and managing build processes [41]. It is the build toolkit of choice in Android Studio. Dependencies, repositories for external packages, specifications and compile options are all specified in the Gradle build files.

A problem encountered in the developing process was the migration to AndroidX. AndroidX is the new version of the support libraries that became available after the release of Android 9.0 (API 28) [42]. When trying to integrate an open-source library for the layout of the results screen, only adding the dependency in the Gradle build file made the app crash with a Manifest merger fail. We discovered the cause to be the use of the old support libraries. The library was already using AndroidX and because of that, the incompatibility appeared. The solution was to migrate the whole project to AndroidX, which is also what Google recommends to do in new projects [42].

[23] Android Package

6 RESULTS

6.1 Features Examination

This section covers the testing of all the features presented in the specifications of the application. The full list of features was presented in Section 2.3.

The sign up with an email and password works as specified. The email must be a valid email address, while passwords shorter than 6 characters are not accepted because of the weak security provided by them. The password must be reentered to make sure that the desired password was written. After creating the account, the application notifies the user if a verification email was sent or an error occurred. If an error occurs, the Toast specifies the error and if something can be done to fix the issue. After receiving the verification email, the user can click the link and wait for the confirmation that prompts them to open the app. If the user does not enter the link, the sign-in is not possible and it is prompted to check the email for the verification email. In our tests, Gmail emails were used for sign up and the verification email was received in less than 15 seconds and never reached the Spam folder.

The Google sign-in option prompts the user to select the desired account. All the Google accounts linked with the Android device appear in the options. The progress dialog asks for the user to wait while the status is updated. If any errors occur (e.g., network error, invalid credentials), the progress dialog is dismissed and a Toast shows all the available information about the error.

The sign-in methods were examined with a working and an unstable Internet connection. The proposed solution is always displaying the progress or the loading of elements in both cases. If there are problems of connectivity, the user is notified about the network error. In this way, if changes can be done to improve the connectivity, the user knows to take action.

The next feature on the list was the option of showing sales in a 50-meter range. The detection works as it is supposed to and other metrics about its performance will be discussed in the next section. The range depends mostly on the beacons and how far they can broadcast the signal. In our tests with the beacon simulator, we reached a maximum range of 40 meters, but this number can be further improved with better equipment. A more detailed analysis is presented in Section 6.2.2.

The ability to set preferences to filter the results was the most wanted feature by the end users. The preferences have a separate tab where all the settings are available and easily

configurable. All the fields with input from the user have ranges that prohibit numbers over the maximum limit or negative. The saving of the preferences is done automatically without the need for a "Save" button. The preferences are also displayed when a search starts.

The favorites are displayed correctly in their associated tab and in the results of the search. After adding or removing an offer as favorite, the change is made instantaneously in the Favorites tab as well. Also, after removing a sale from the Favorites tab, the next search is going to take that into account. Removing an offer from the favorites is done by sliding the offer and confirming the action. Without confirmation, no offer is removed. The sorting criteria for the favorite offers can be set in the Preferences tab. For all these scenarios there were also conducted tests for corner cases (e.g., no favorites saved) to assure that all the specifications are met.

The UI/UX was designed to respect the Android Design guidelines [12]. The keyword for the design was consistency. The proposed solution is consistent with dimensions, icons, colors, and design elements. The light-themed app uses red as primary color because of its association with sales. The accent color of the app is purple, which has good contrast with the primary color. The bottom navigation bar emphasizes the current screen by color and stating the name of the opened tab. The clickable elements usually are colored with the accent color and always have an explicit text or icon. This way, the action of a certain element is clarified.

6.2 Performance Analysis

The performance of the proposed solution is important for the prospect of this kind of mobile application. For the commercial use of such a solution, it must satisfy certain standards of robustness, reliability, and speed. In this section, we present the results of the tests challenging these standards.

As previously mentioned, for simulating a beacon we used another Android smartphone and the Beacon Simulator app. The broadcast settings of the beacon include transmission power and frequency mode.

Transmission power determines the power of the transmitted signal. Transmission power had 3 available options in Beacon Simulator: Low (-75 dBm), Medium (-66 dBm), and High (-56 dBm). This values are relevant for the range of the broadcast, but can also have an impact on the battery life of the transmitter because of the need for more power.

Frequency mode specifies how often the signal is broadcast. There were three frequency modes as well: Low power (1 Hz), Balanced (3 Hz), and Low latency (10 Hz). These values

impact how fast a beacon can be found because of the number of broadcasts that happen every second. A higher frequency causes a bigger battery drain.

For analyzing the performance, we varied these two parameters and gathered data about how fast the beacon is found. The testing was done in the open field, with nothing passing between the transmitter and receiver. For each configuration of the two parameters, we started from an initial distance of 5 meters and took 5 measurements. If the beacon was not found within 8 seconds, more measurements were made until the beacon was found or not found 5 times. This way, the number of scans done for a configuration is between 5 (e.g., 5 successful scans) and 9 (4 successful and 5 unsuccessful scans). The distance would increase with 5 meters if the beacon was found successfully 5 times. If this criterion is not met, it is assumed that it would fail for all longer distances.

6.2.1 Reliability Performance

The proposed solution aims to be reliable and operational in real life scenarios. For this to happen, we must be certain that the solution is able to find all the beacons in the proximity of the user time and time again. This depends on the equipment used for transmitting the beacons, but also on the implementation of beacon scanning. The accuracy with which the app can find the beacons is presented in Figure 23. The used method of testing was the one presented at the end of Section 6.2. It can be seen that the success percentage is inversely proportional with the distance. The chart sums up all the configurations and this is the reason for the small percentages. The best configuration (High Power and Low Latency) has a 100% success ratio for all distances up to 35 meters.

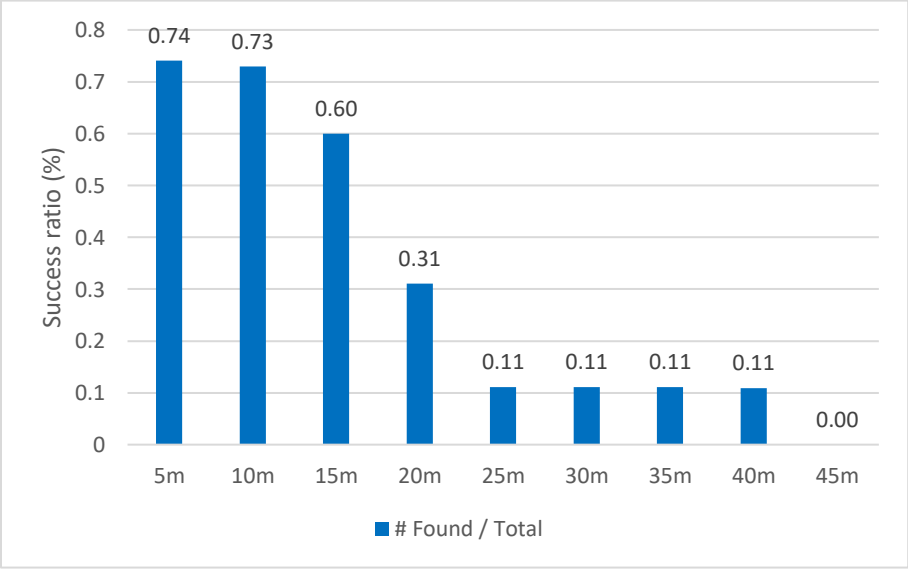

Figure 23. Success ratio of beacon scanning

This type of test is good to find how reliable the beacons are on a wide range of distances. In a real-life scenario, there would be more than 1 beacon in the proximity. In a shopping center, there could be areas with even more than 10 beacons. This is the reason why testing the scanning of multiple beacons is important. We conducted a test where we put the smartphone receiver 1 meter away from the transmitter device and enabled a variable number of beacons. The smartphone used to simulate the beacons can broadcast simultaneously up to 9 beacons. Because of this, we tested for 1, 2, 4 and 8 beacons with default broadcast settings (Medium power and Balanced). The average results of 10 tests can be seen in Figure 24. The first detection is the time needed for finding the first beacon, while the last detection is how much it took to find all beacons. Evidently, these times are the same for 1 beacon.

This experiment helped us realize what are the consequences of placing more beacons in the same area. Firstly, there were no problems with detecting all the available beacons. Secondly, the time needed for the first detection doesn't seem to depend on the number of beacons broadcast in the area. And lastly, as expected, the time needed for detection is directly proportional to the number of broadcasting beacons. Even though the average of the last detection for 8 beacons is almost 7 seconds, there were no values larger than 7.35 seconds.

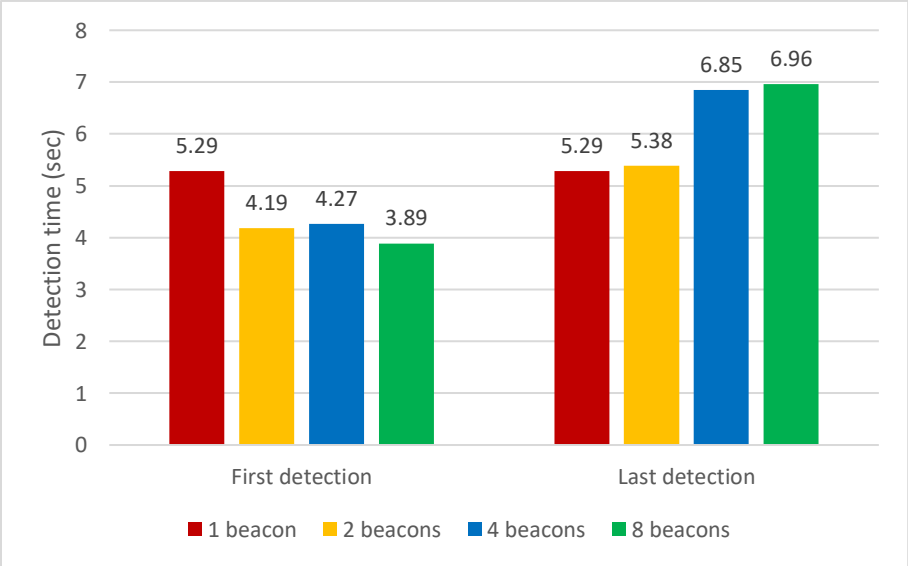

Figure 24. Average time needed for beacon detection in 1-meter range

6.2.2 Range Performance

The range of the solution is represented by the maximum distance between the beacon and the end user that would determine a successful detection. The broadcast settings play a major role in the range value.

For determining the range performance, we used the method presented at the end of Section 6.2. In Figure 25, the number of successful detections for every tested range can be seen. The maximum value at a given range would be 45 (9 configurations, each with a maximum of 5 successes). It can be observed that in small ranges the number of successes is close to maximum. By 20 meters, only the beacons with configurations with high transmission power or high frequency are found. For bigger distances, only the best configuration prevails. By setting the beacon broadcast to High Transmission Power and Low Latency Frequency, the success ratio remains 100% up to 40 meters. This can be seen in Figure 25, where the values between 25 and 40 meters are constantly 5. At 45 meters, even with the best configuration, we are not able to detect the beacon.

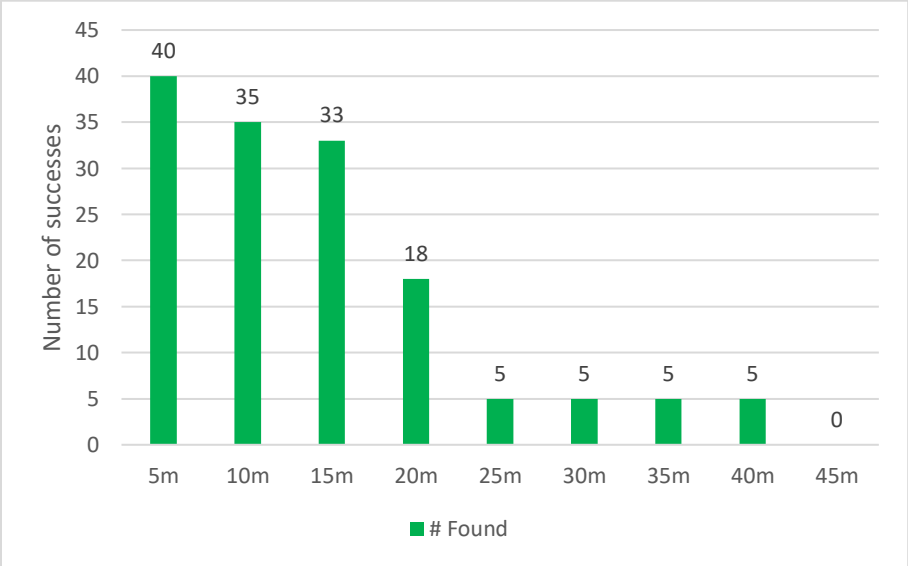

Figure 25. Successful searches

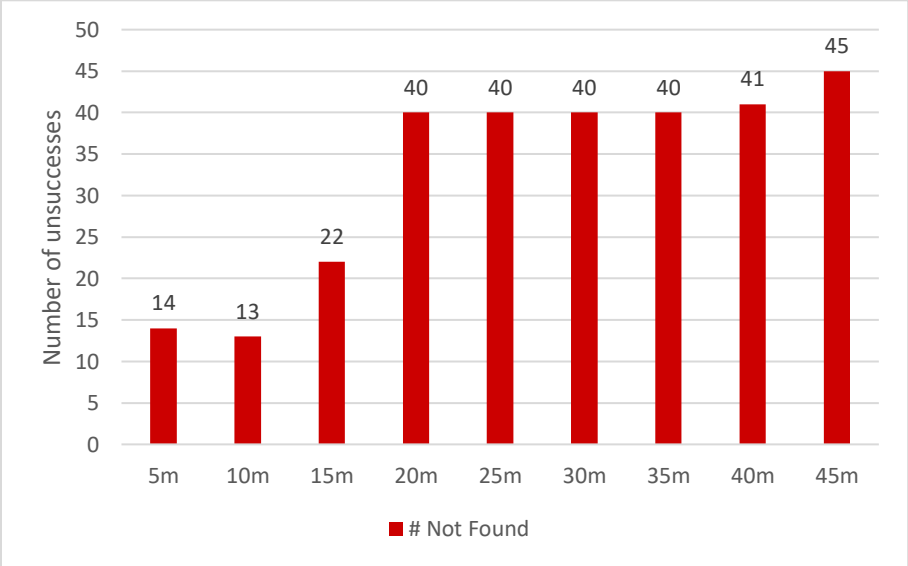

Figure 26. Unsuccessful searches

By presenting only the number of successful detections, we do not get the full overview of the range performance. The testing method can also emphasize how frequently unsuccessful detections occur. The total number of unsuccessful searches can be found in Figure 26. As expected, the numbers increase with distance. The maximum value at a given range would be 45. This value is reached only at the biggest distance after the best configuration fails to find the beacon anymore.

An interesting analysis would consist of the number of beacons needed to cover a whole shopping center. We will further discuss the case of AFI Palace Cotroceni, one of the largest shopping centers [43] in Bucharest, Romania. Its retail zone has a total surface of 90,000 m^2 [43]. For the purpose of this experiment, we will use Estimote's Proximity Beacons [44]. Their advertised range is of 100 meters, but that can decrease in crowded areas with lots of consumers and other beacons. Because of this, we will use a range of 50 meters in the following calculations.

The beacons would be mounted on walls and would broadcast the signal in a circle with the following area:

$$BeaconArea = \pi R^2 = \pi \cdot 50^2 \cong 7{,}850\ m^2 \quad (1)$$

If one beacon can cover 7,850 m^2, for the 90,000 m^2 shopping center a total of about 12 beacons would theoretically cover all the surface of the center. At the time of writing, a kit of 4 Proximity Beacons cost 99$ [44]. That would bring the total cost of equipment to a mere 297$.

In reality, this figure would be much bigger because of the layout of the shops. The center also has a second floor and that could lead to at least doubling the number of beacons. This experiment was intended to prove that the technology is affordable and could be implemented in a real-life scenario with minimal financial costs.

6.2.3 Detection Performance

It is extremely relevant for the proposed solution to be able to detect all the beacons in the proximity of the user in the 8-second threshold. Also, another relevant metric is how fast this detection is done. As can be seen in Figure 27, we aggregated all the results for all ranges to see how the broadcast settings impact the detection times. In the chart, the results were separated into 3 categories based on the transmitting power: Low, Medium, and High. This separation was done because this parameter should not affect the detection times. The frequency of the signal transmission is more important for the detection times.

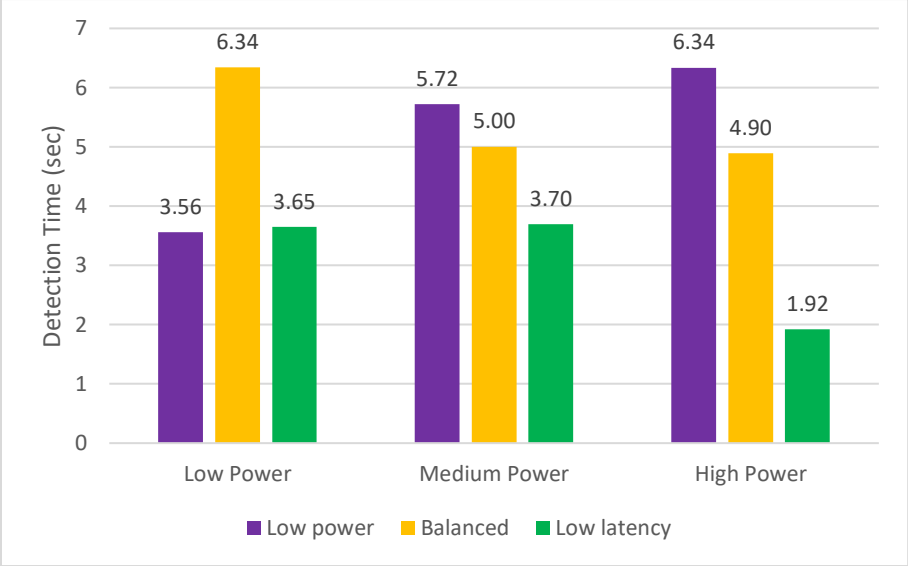

Figure 27. Average detection times for all broadcast configurations

When the transmission power is low, the results are not conclusive. Because of the small range, only a few tests could have been conducted. With Medium or High transmission power, an existing trend line can be observed. The time needed for detection is inversely proportional to the frequency of the transmissions. The best configuration has an average time of fewer than 2 seconds for beacon detection. The chart in Figure 27 proves that with the right settings, a beacon can be used for a wide range of purposes, depending on the application use cases. There is always a compromise to be done between performance and battery life.

7 CONCLUSIONS AND FUTURE WORK

7.1 Conclusion

The retailing industry is a trillion US dollars industry that is trying to reinvent itself as technology advances. Even with important growth in online shopping, the majority of the retail revenue is generated in stores. Retailers are trying to find new ways to engage consumers to enter their stores. One of the most recent trends is context-based applications that are aware of the shops and items in the user proximity.

This thesis provides a solution that can help retailers attract more customers and, at the same time, assist consumers in finding better offers. In this paper, we presented the implementation and benefits of a context-based Android application capable of searching for shops through beacon scanning. We proved the usefulness, from both the consumer and retailer perspectives, and the superiority of the solution compared to other similar already existing applications. After that, a high-level presentation of the technologies used, the architecture and other technical details were presented. In the end, we covered the performance of the solution from reliability, range and detection standpoints and concluded that it is operational for real-life use in commercial centers.

7.2 Future Work

For future work, there are several enhancements that can be made, some of which were already foreshadowed in the past chapters.

The most important improvement would be the addition of a web platform where partnering retailers could add, edit or remove their available offers. Another practical feature, especially for big retailers, would be an automated process for updating offers. Some retailers do already have a strong web infrastructure and display their promotions in a special section on their website or mobile app. Crawling that data and automating the process of updating promotions in the ShopSmarter platform could help attract more big retailers. With this feature, the effort would be minimized for many stores.

Although Android is the most popular mobile platform, iOS is fairly popular as well. For this significant percent of end users, the need for a specialized app is indisputable.

Other potential subjects of work could be categorized as R&D[24] activities. Features like NFC detection, QR scanning or a custom beacon scanning library could play a significant role in the overall usefulness and performance of the ShopSmarter app.

[24] Research and development

8 BIBLIOGRAPHY

[1] Deloitte, “Global Powers of Retailing 2018 Transformative change, reinvigorated commerce,” *Deloitte*, pp. 1–24, 2018.

[2] DAN, “The Future Of The Online Vs Offline Shopping Battle,” 2019. [Online]. Available: https://digitalagencynetwork.com/the-future-of-the-online-vs-offline-shopping-battle/. [Accessed: 24-Jun-2019].

[3] S. Gerasenko, A. Joshi, S. Rayaprolu, K. Ponnavaikko, and D. P. Agrawal, “Beacon signals: what, why, how, and where?,” *Computer (Long. Beach. Calif).*, vol. 34, no. 10, pp. 108–110, 2001.

[4] V. Hiribarren, “Beacon Simulator.” [Online]. Available: https://play.google.com/store/apps/details?id=net.alea.beaconsimulator. [Accessed: 24-Jun-2019].

[5] “Eddystone format.” [Online]. Available: https://developers.google.com/beacons/eddystone. [Accessed: 24-Jun-2019].

[6] K. International, “Retail Trends 2019 Global Consumer & Retail,” no. February, 2019.

[7] R. Roth, “User Interface and User Experience (UI/UX) Design,” *Geogr. Inf. Sci. Technol. Body Knowl.*, 2017.

[8] J. Young, “Global ecommerce sales grow 18% in 2018,” 2019. [Online]. Available: https://www.digitalcommerce360.com/article/global-ecommerce-sales/. [Accessed: 24-Jun-2019].

[9] F. Ali, “A decade in review: Ecommerce sales vs. retail sales 2007- 2018No Title,” 2019. [Online]. Available: https://www.digitalcommerce360.com/article/e-commerce-sales-retail-sales-ten-year-review/. [Accessed: 24-Jun-2019].

[10] P. Zhang, Y. He, and C. (Victor) Shi, “Retailer’s channel structure choice: Online channel, offline channel, or dual channels?,” *Int. J. Prod. Econ.*, 2017.

[11] Unilever, “Making Purpose Pay,” 2017.

[12] Google, “Design | Android Developers,” *Android Developers*, 2017. .

[13] W3MT, “AFI Cool.” [Online]. Available: https://play.google.com/store/apps/details?id=com.evonomix.afi. [Accessed: 24-Jun-2019].

[14] B. Developments, “CITY iLOVE by Baneasa.” [Online]. Available: https://play.google.com/store/apps/details?id=ro.baneasashoppingcity.bsc. [Accessed: 24-Jun-2019].

[15] H&M, “H&M - we love fashion.” [Online]. Available: https://play.google.com/store/apps/details?id=com.hm.goe. [Accessed: 24-Jun-2019].

[16] Inditex, “Bershka - Fashion and trends online.” [Online]. Available: https://play.google.com/store/apps/details?id=com.inditex.ecommerce.bershka.

[Accessed: 24-Jun-2019].

[17] Z. He, B. Cui, W. Zhou, and S. Yokoi, “A proposal of interaction system between visitor and collection in museum hall by iBeacon,” in *10th International Conference on Computer Science and Education, ICCSE 2015*, 2015.

[18] M. Enis, “‘Beacon’ technology deployed by two library app makers,” *Libr. J.*, 2014.

[19] M. Inc., “NearBee - Discover what’s buzzing around you.” [Online]. Available: https://play.google.com/store/apps/details?id=com.mobstac.nearbee&hl=ro. [Accessed: 24-Jun-2019].

[20] S. Choudhary, “How to configure your beacons so that NearBee detects them,” 2019. [Online]. Available: https://docs.beaconstac.com/nearbee/how-to-configure-your-beacons-so-that-nearbee-detects-them. [Accessed: 24-Jun-2019].

[21] I. Groupon, “Groupon - Shop Deals, Discounts & Coupons.” [Online]. Available: https://play.google.com/store/apps/details?id=com.groupon. [Accessed: 24-Jun-2019].

[22] Slickdeals, “Slickdeals: Coupons and Deals.” [Online]. Available: https://play.google.com/store/apps/details?id=net.slickdeals.android. [Accessed: 24-Jun-2019].

[23] K. Yaghmour, “Embedded Android: Porting, Extending, and Customizing,” p. 413, 2013.

[24] statista, “• Mobile OS market share 2018 | Statista,” *statista*, 2019. .

[25] S. Aggarwal, “Android vs iOS: Which Mobile Platform is Best for App Development,” 2019. [Online]. Available: https://www.techaheadcorp.com/blog/android-vs-ios/. [Accessed: 24-Jun-2019].

[26] Google, “Google Beacon Platform - Platform Overview.” [Online]. Available: https://developers.google.com/beacons/overview. [Accessed: 24-Jun-2019].

[27] Google, “Nearby platform.” [Online]. Available: https://developers.google.com/nearby/. [Accessed: 24-Jun-2019].

[28] R. Nayak, “Discontinuing support for Android Nearby Notifications,” 2018.

[29] “Enable interactions between nearby devices and people.” [Online]. Available: https://developer.android.com/distribute/best-practices/engage/nearby-interations. [Accessed: 24-Jun-2019].

[30] “Nearby Messages API - Overview.” [Online]. Available: https://developers.google.com/nearby/messages/overview. [Accessed: 24-Jun-2019].

[31] Radius Networks, “Android Beacon Library.” [Online]. Available: https://altbeacon.github.io/android-beacon-library/. [Accessed: 24-Jun-2019].

[32] “Firebase Products.” [Online]. Available: https://firebase.google.com/products. [Accessed: 24-Jun-2019].

[33] “Firebase - Pricing plans.” [Online]. Available: https://firebase.google.com/pricing. [Accessed: 24-Jun-2019].

[34] “AWS - Cloud Products.” [Online]. Available: https://aws.amazon.com/products/?hp=tile&so-exp=below. [Accessed: 24-Jun-2019].

[35] “Amazon Cognito - Pricing.” [Online]. Available: https://aws.amazon.com/cognito/pricing/. [Accessed: 24-Jun-2019].

[36] “Android - Distribution dashboard.” [Online]. Available: https://developer.android.com/about/dashboards/index.html. [Accessed: 24-Jun-2019].

[37] “Android - Bluetooth low energy overview.” [Online]. Available: https://developer.android.com/guide/topics/connectivity/bluetooth-le. [Accessed: 24-Jun-2019].

[38] “Nearby Messages API - Get Started.” [Online]. Available: https://developers.google.com/nearby/messages/android/get-started. [Accessed: 24-Jun-2019].

[39] Google, “Beacon Tools.” [Online]. Available: https://play.google.com/store/apps/details?id=com.google.android.apps.location.beacon.beacontools. [Accessed: 24-Jun-2019].

[40] “Nearby Messages API - Get Beacon Messages.” [Online]. Available: https://developers.google.com/nearby/messages/android/get-beacon-messages. [Accessed: 24-Jun-2019].

[41] “Gradle User Manual.” [Online]. Available: https://docs.gradle.org/current/userguide/userguide.html. [Accessed: 24-Jun-2019].

[42] “Android - Support Library.” [Online]. Available: https://developer.android.com/topic/libraries/support-library. [Accessed: 24-Jun-2019].

[43] A.-E. Vasiliu, “Analiză ZF: AFI Cotroceni conduce topul mallurilor după suprafaţă. Un metru pătrat ar trebui să genereze vânzări de 1.500 de euro pe an,” *Ziarul Financiar*, 18-Mar-2019.

[44] “Estimote Products.” [Online]. Available: https://estimote.com/products/. [Accessed: 24-Jun-2019].