Zagazig University,

Faculty of Computer and Informatics

Department of: IT, DS

B. Eng. Final Year Project:

Student Service Zone

By:

Mohamed Emad Abd El-Samie

Mohamed Fathy Abd El-Kamel

Amira Salah Ahmed

Aya Mohamed Saleh

Aya Hassan El-Sayed

Aya Hassan Fouad

Supervised by:

Prof. Ehab Rushdy

Eng. Heba Khater

ACKNOWLEDGEMENT

We would like to thank everyone who contributed to this project and giving us this great opportunity to learn and try new technologies and work on real project.

We would also like to specially thank **Dr Ehab Rushdy** for inspiring us with this great idea, encouraging us to even enhance it and putting us under the supervision of one of the best teaching assistants in the university **Heba Khater** whom we also want to thank for the great support and guidance through the whole year . we really learnt a lot thanks to them, and we wish them the best for what they have done for us.

DECLARATION

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Bachelor of Computer Science is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: ______________________________

Signed: ______________________________

Signed: ______________________________

Signed: ______________________________

Signed: ______________________________

Signed: ______________________________

Registration No.: ___________

Date: 11, 08 2020.

ABSTRACT

Hand in hand is based on two applications. a web application and an android application. Combined, these two applications will provide a social solidarity platform for students to show the spirit of kindness and help each other. Both applications will offer the same functionality and they are both connected via a RESTful API to ensure the consistency of data so the user can reach it no matter what device is being used. Mainly, hand in hand has 4 core resources which are items, hand made products, services and events and two types of users a normal user and a trusted user.

The purposes of these resources are as follow: -

- **Items**

  An item represents a used study material like a book or a lab equipment the purpose of this resource is for students to donate the materials they do not need any more to younger students for free or cheaper price.

- **Products**

  A product represents a handmade object the purpose of this resource is for students who have a hoppy or like making small business in creating something like accessories, paintings , etc. to find a place where they can share what they create and earn money from it.

- **Services**

  A service is like a Facebook post the purpose of it is for students to help other students by for example volunteering to explain a subject or making a workshop or providing general advices for other students.

- **Events**

  An event can be a seminar or a specific class where an advanced topic can be taught or even a concert the purpose of this resource is to help students find all the useful events near them in the university.

Users can interact with resources by adding, interesting, reporting, etc. the only difference between the two types of users is that the trusted user is the only one allowed to add an event to ensure that only trusted and useful events are added.

Hand in hand has so much potential and future work we will explain it in details in this documentation.

Table of Contents

INTRODUCTION

We are living in the age of science. Each day you hear about new technologies and inventions in many different fields of life. With new technologies come the need to new ways and sources for education. Most of the changes to the way of education happens at the time of the university as there are less restrictions on the learning resources. New ways of learning require materials and for that thousands of books are printed, and tools are made to make the education process easier. As every material has a price this can be a burden for the student or the country if it provides these materials for free or lower prices. Let us take Egypt for example.

Egypt is known for the high number of youths. According to statistics about **3 million** students go to universities each year. Students need materials to go through the college. Some of them can be provided by the university and others the student should get them himself. Providing materials for about 3 million students is huge but **the problem is when these materials are not needed, they are mostly thrown away each year.**

Life in university is different. It is the last stage of education before joining the market and discover how real work and projects are done. That is why in university you will find events made by companies and organizations to teach and attract students, student union for helping students and other student activities and families. It appears that university is a more social place than school **so why all these materials are wasted not donated?** We knew what the problem was, but we wanted to make sure, so we conducted a survey. The survey consisted of questions mostly about the materials they use, the events they attend, the help they get and other basic questions about their life in university. The survey showed that:

- Most of student's pays from **500LE** to **3000LE** each year for materials only and others pays a lot more than that.

- Students are willing to donate unneeded materials, and many accept working with used material even if it is sold with lower cost than the new one.

- **77.4%** said that not all the posts and news like events reach them on social media groups.

- **54.7%** said that even if they asked for something on groups there is no quick response.

- **51%** of them did not go to a single event and many of them said that they do not even know if there is an event or not.

The problem can be seen as a communication problem. **Students cannot find the right person who needs the materials and the needing student cannot find the donator.** Also, students cannot reach for the events made by organizations and student activities despite all the existing social media applications. So, they need a way of communication dedicated and customized for students only.

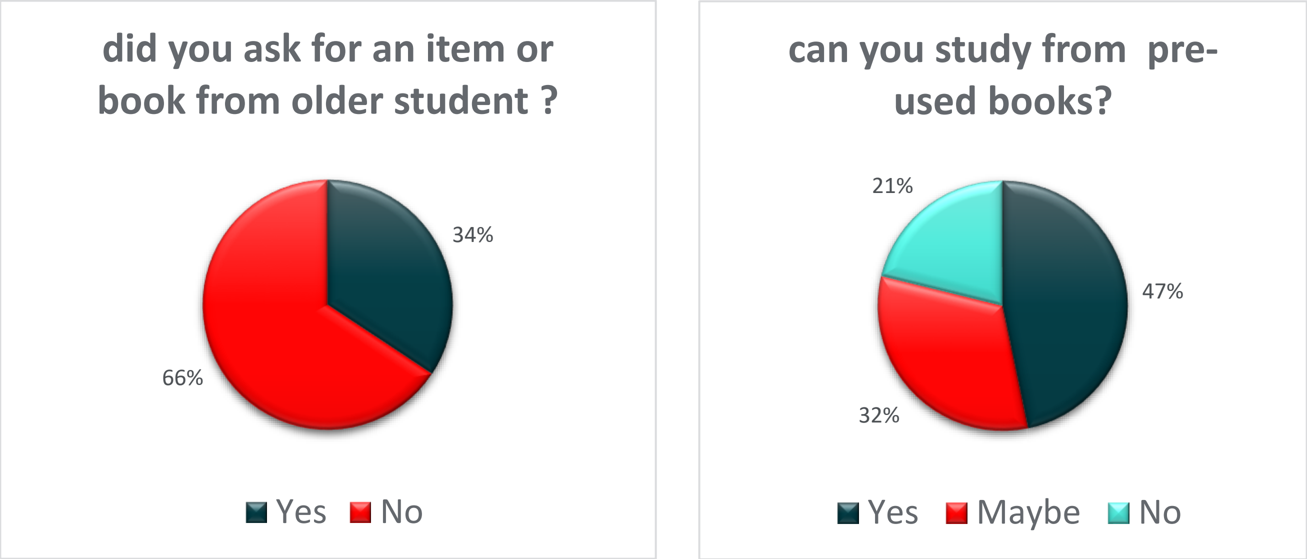


Figure 1: Survey results

The project consists of a website and an android application that provide a platform for students to find what they need and share their tools, books, knowledge and many other things with each other.

The platform will provide four major services: -

1. A way to donate, sell, exchange and search for study materials like books and tools.

2. A way to find, offer and ask for services (questions - teaching or explaining a subject, conducting workshops – trips).

3. A way for students to know about all the important events happening soon.

4. A way for students who have a talent in making hand-made products to come to light and show their talent. It can be a start for their future career.

It is a highly scalable project with huge potential for more future wo

BACKGROUND

As we mentioned before the problem is mostly about communication and delivering the right service to the right user. Social media applications and E-commerce can be a part of the solution but not a complete one.

Due to the large amount of users with different interests and needs it is hard for these solutions to be dedicated for only one category of people which is students in our case so Let us give some examples on the difficulties the students may go through.

- ## Epay, Amazon, Souq, Facebook (marketplace) and other e-commerce companies: -

For selling products like books and other material these companies sure can do it but for students to get what they want there will be some difficulties like: -

- These companies sell for all the world and there may be shipping cost even if it is in the same country our solution can provide the items in the same university so the student can get the item with a simple meeting between lectures

- Many of these companies do not sell pre-used products meaning the student will buy a new item so nothing is solved.

- No student will go to a large worldwide company and go throw their laws, policies and contracts if they exist just to end up selling a simple pre used study material for them it will be an overkill.

- ## Facebook and social media

While it is true that social media websites especially Facebook already have all these features so why would anyone use our solution?

The answer will be they will use it for the dedication. Social media purpose is to simplify communication among all the people, which is awesome, but problems will start to show up due to lack of dedication, here are some examples on Facebook: -

- After finishing an academic year there will be some books and tools that you don’t need any more **if you want to donate them where will you post this?** Of course you can post on your timeline but it won’t reach all the people or you can search for groups for younger students to post but even if you posted there you won’t know if the students who need the material will see the post. While in our solution all you have to do just publish the material and anyone who wants it will just search for the name of the book or tool, no need to search in groups, ask your friends to share or mentioning at all.

- Anyone can make events on Facebook that’s why some of them are just spam, Facebook will notify you about events that your friends are interested in or you may see someone sharing an event by chance, other than that **you have to depend on yourself on finding the events that may suit your need and passion which isn’t the way things should work.** Users shouldn’t search for events instead events should reach the right users and that’s what we can provide by eliminating the chance of spam events and focus the event subjects on learning and students we believe we can easily connect students to organizations and help prepare them for the future.

- if you are good at making hand-made products and you want to tell people about it, **sell your products and start a small business you will probably go to marketplace** or the simple post and share way while market place is free to use it isn’t just for handmade projects or simple products and student materials. You will find cars, buildings, mobiles, etc… and for people to know you it will be by chance when they see your products because it happens that they are near.

The point is **Facebook is too general** so it lacks the dedication and focus on a specific field, it was built for that but our solution focuses on the students, you don’t have to pay for advertisement for your small business to come to light, you don’t have to go through many posts, groups and pages to find what you need we will take care of all of that so you can just focus on helping others and make your own future.

Our solution purpose is to **facilitate social solidarity among students,** and it is hard to find other solutions for this specific purpose, existing solutions lack the dedication. They are not built for the use of students only they are built for everyone.

The solution is available on web and android. It is a real time solution that will notify users once an important change is made. We used html, CSS, php and MySQL for the website and java for the android.

BUSINESS MODEL

The project purpose is to act as a platform for solidarity among the students as the name suggests **"Service Zone"**. That means we cannot charge students for money to let them help others. This project will not charge students anything at all. Our income can come from advertising to keep the project running and progressing without the need for students' money. This is the simplest way to make this a **win-win situation**. The steps we walked through to make this project were also simple and they are as follow: -

1. **Searching for an idea:**
    - We tried to look for a problem that affects the same community we are part of. We were able to get many ideas, but we found this to be the best one.

2. **System analysis:**
    - After thinking about the models, actors, processes and all other needed information, we refined the idea and added more functionality to the system, then everyone was assigned his roles and we moved to the technical work.

3. **Prototyping:**
    - At this step, the project began to take shape. The purpose of this step was to try all possible designs that can fit the idea. We walked through the **best practices** for using colors, dimensions, structures, fonts and applying user experience without affecting the quality of user interface. This resulted in many designs for both the web and android before using this one. We used tools like Photoshop, illustrator and invasion studio with the help of Google material design.

4. **Implementing the prototype:**
    - Following the guidelines of the prototype members who are responsible for the design started to implement it. For the front end we used **HTML** – **CSS** – **JavaScript** – **bootstrap** and for android we used **XML** and **Java.**

5. **Implementing the logic:**
    - After finishing the design what is left for the project to come to life was to send the data between the pages and for that we used **MySQL** as the database, **Laravel (PHP)** for back end development and **java for android.**

6. **Testing:**
    - Testing was a **part of each step** we did not move from a step to another until we made sure it is done perfectly but that was individual testing. After finishing all five steps we are now talking about testing the project. We wanted to make sure every part of the project is interacting well with other parts.

7. **Feedback and maintenance:**
    - Further observing and refining for the project based on users' feedback and new ideas to deliver better user experience.

DYNAMIC BEHAVIOR OF THE SYSTEM

After seeing the survey results the requirements **(the user requirements)** for the new system become clearer now.

Our goal is to create a usable website and android application that meet the user requirement. So, we keep the user requirement in our mind while we design or create the behavior and the interactions of the system.

In this step our goal is to convert the User Requirement to functions in the system or interactions to make it easy to use and easy to be developed. In the other hand **we want to avoid the failure parts of other solutions** like Facebook or Olx for example. At first, we will describe the user requirement that we conclude it from the survey result. Then we will show the Main use cases or main functions of the system.

It seems that other alternative solutions like Facebook or Olx or Epay do not behave will because these sites more general and the student want site or app that the whole system depend on students only, so it become easy to search on it.

When the student searches the result that will be shown will be from another students so now **the site is more special or more restricted** and depends only on student. When I search, or I post my items.

We took this point in our mind while we design the behavior of the system. You will see in the coming parts of the documentation words like **"User"** it means the student that will use the system.

After converting the user requirements into interaction parts of the system or convert it to functions included in the system, now we will add more functions that will help the main functions of the system to make it secure or more usable and easier to use.

Now let us show how the system will interact and the main function that our system depends on see the use case diagram in **figure 2.**

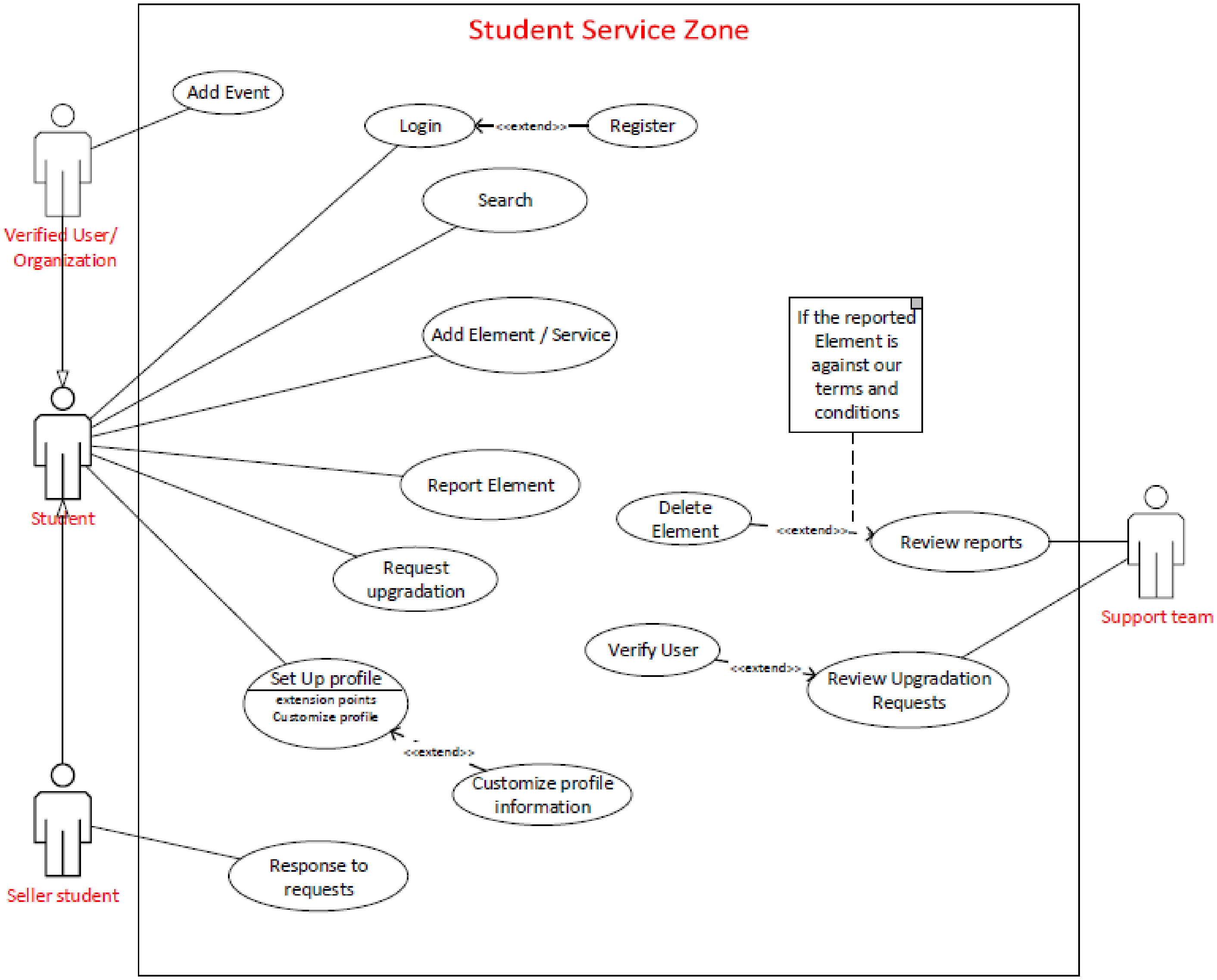


Figure 2: Use case diagram

In Figure 2 the Use case diagram shows what the system does and how the actors will interact with the system. There are three primary actors which are:

1- Student
2- Seller student
3- Verified user / organization

The main actor of the system is the student and for more information and to make the diagram simpler we divide it to another two types.

The Seller student is a normal student with whole system functionality, but we inherit the functionality from the main student to show that the seller student is the student who sell book or donate it for example.

The Verified user is a normal student too but with more special function that let he/she make an event. Here the user must ask to upgrade his account to be able to share or make event.

The only secondary actor of our system is the support team their work is to review the reports elements and delete it or keep it in the system the work is so simple, but it is very important to make the system safe and prevent spam elements.

After we show the main actors and the main functions of the system. Now let us list the description or the scenario of the main functions, we will describe the most important function that will affect our system and our data, and simple functions can be easily understood from its name only:

Use Case Name: Search	**ID Number:** UC - 1	**Priority:** Medium

Actors: Student

Description: This use case describes Student Who Search about elements / services

Trigger: Student at System to search and browse

Type: External Temporal

Preconditions:

- System is available
- Elements database is online

Normal Courses: Search and browse elements		**Information for steps**
1- System displays default home page	←	Username / password
2- Students enters account username / password		
3- Student enters Search request	←	Search criteria
4- System displays elements matching search request	→	Elements matching search
5- Student select elements	←	Choose element
6- Student wants to see a sample	→	Element sample
7- Student reserve elements who selected them	←	New reservation

Alternative Courses:		
2.a student create an account	←	Entered Details

Post conditions:

- Student found the search
- Elements are reserved

Exceptions: Account is not valid

- System display message that username / password is not valid
- System asks student to reenter username / password
- Search request return no results
- System displays message that no results were found for that search

Input	**Source**	**Output**	**Destination**
Username / password	Student	New Reservation	Elements Database
Search criteria	Student		
Elements matching search	Elements Database		
Element sample	Elements Database		

Use Case Name: Add elements / services	**ID Number:** UC - 2	**Priority:** High

Actors: Student

Description: This use case describes Student Who can add elements / services to the system

Trigger: Student wants to Add elements / services

Type: **External** **Temporal**

Preconditions:

- There is internet
- System is available
- Student have an account on the system

Normal Courses:		**Information for steps**
1- Student enter account username / password	←	Username / password
2- Student enter description of elements /services	←	Element Description
3- Student add elements / services	→	New element

Post conditions:

- Elements exist on the page
- Elements may be modified

Exceptions:

- Student does not have device or internet
- Student does not have an account
- Account is not valid

Input	**Source**	**Output**	**Destination**
Username / password	Student	New element	Elements Database
Description	Student		

Use Case Name: Send Deal Details	**ID Number:** UC - 3	**Priority:** High

Actors: Seller person

Description: This use case describes Seller Person Who response on student request

Trigger: Seller person makes confirm on student request

Type: External Temporal

Preconditions:

- Student ask a service / send a request
- Seller person accept on request

Normal Courses:		**Information for steps**
1- Student enter search request	←	Search criteria
2- Student select a service	←	Choose elements
3- Student wants to see a sample	←	Element sample
4- Student send a request	→	New request
5- Seller person confirm the request	→	Make confirmation
6- Seller person send deal request	←	Send details
7- Seller person decline the request	→	Rejection request

Post conditions:

- Seller person accepts on request
- Student have deal details that are sent to him / her

Exceptions:

- Seller person make decline on request

Input	Source	Output	Destination
Search criteria	Student	New request	Elements Database
Choose elements	Student	Make confirmation	Elements Database
Send deal details	Seller person	Rejection request	Elements Database
Elements sample	Elements Database		

Use Case Name: Edit profile	**ID Number:** UC - 4	**Priority:** High

Actors: Student

Description: This use case describes Student Who edit profile information or items

Trigger: Student create an account on the system

Type: External Temporal

Preconditions:

- System is available
- Internet is available
- Student have an account

Normal Courses:		**Information for steps**
1- Student enter username and password	←	Username / password
2- Student edit profile information	←	Entered / information
3- Student update items that exist on profile	→	Modified elements
4- Student remove items from profile	→	Modified elements

Post conditions:

- New information exists on profile.
- There is a new detail of profile.

Exceptions:

- Internet is not available
- Student does not have an account
- Account is deactivated
- Student does not have device

Input	Source	Output	Destination
Username / password Entered information	Student Student	Modified elements	Elements Database

Use Case Name: Delete elements	**ID Number:** UC - 5	**Priority:** High

Actors: Support team

Description: This use case describes Support Student can delete elements which student entered against application terms and conditions

Trigger: That elements against application terms and conditions

Type: External Temporal

Preconditions:

- Support team reviewed reported elements
- elements against application terms and conditions

Normal Courses:		Information for steps
1- system displays items / elements with details	←	Elements details
2- student search elements which are wanted	←	Search criteria
3- student report elements to delete them	←	Reported elements
4- support team see these reports to evaluate them	→	Evaluated reports
5- elements are deleted by support team	→	Modified elements

Post conditions:

- Student reported elements
- Elements are deleted

Exceptions:

- Elements aren't against application terms and conditions
- There are no reports

Input	Source	Output	Destination
Search criteria	Student	Evaluated reports	Elements Database
Do report	Student	Modified elements	Elements Database

USER INTERFACE

The main idea of the project is to **encourage students donate to each other** without any emphases. And let the students sell and post their products (homemade). So, we gave the User Interface (UI) much time to let the student see awesome and easy to use UI.

Also, we take care of the fundamental’s parts of the UI:

1. Navigation mechanism
2. Input mechanism
3. Output mechanism

As it known the UI is an art. So, our UI goal is clearly known from the beginning, which is to **let the student please** while browsing the website or the application. We used simple and awesome fonts and colors.

The first and **the core part of the UI is our logo** which includes our message, or our goal and it includes also the two primary colors of our project.

Here is our logo that our Designers team made:

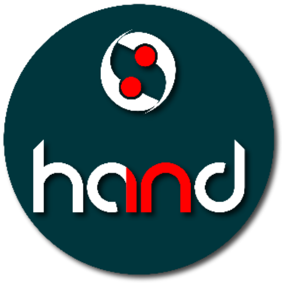


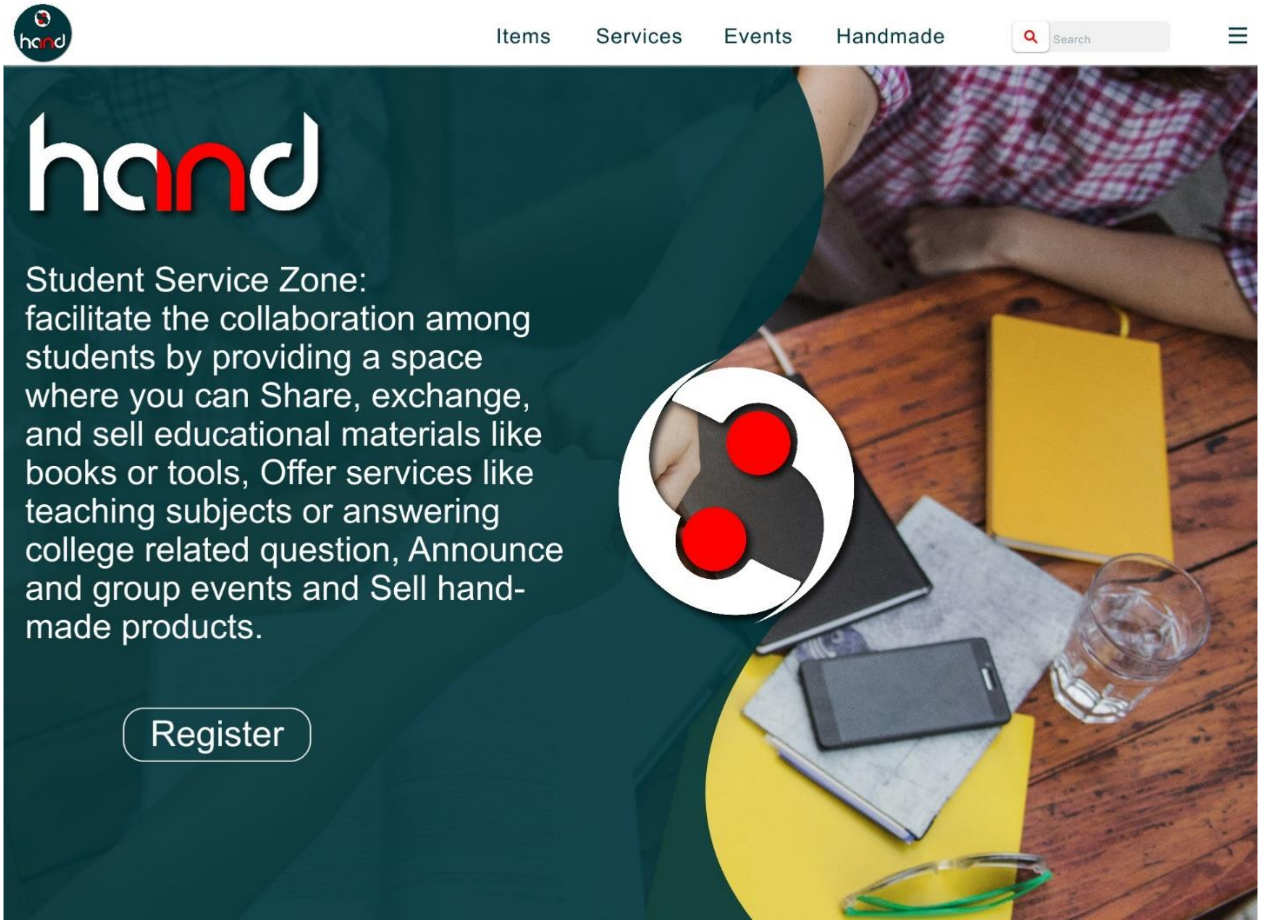
Figure 3: The logo of the project

As it shown in **figure 3**, We put many concepts in the logo and, we chosen a pleasing colors and simple font.

Our project name is **“Student Service Zone”** this name is a formal name, so we choose another name to be our brand name and at the same time give the meaning of helping and donation. So, we choose **“Hand In Hand”** to be **our brand name** and we use it on the logo with the awesome meaning with pretty colors.

Now after we show one of the main parts of our UI let us dive in the main design and pages of the project.

Let us see the landing page of the website which shown in figure 4:



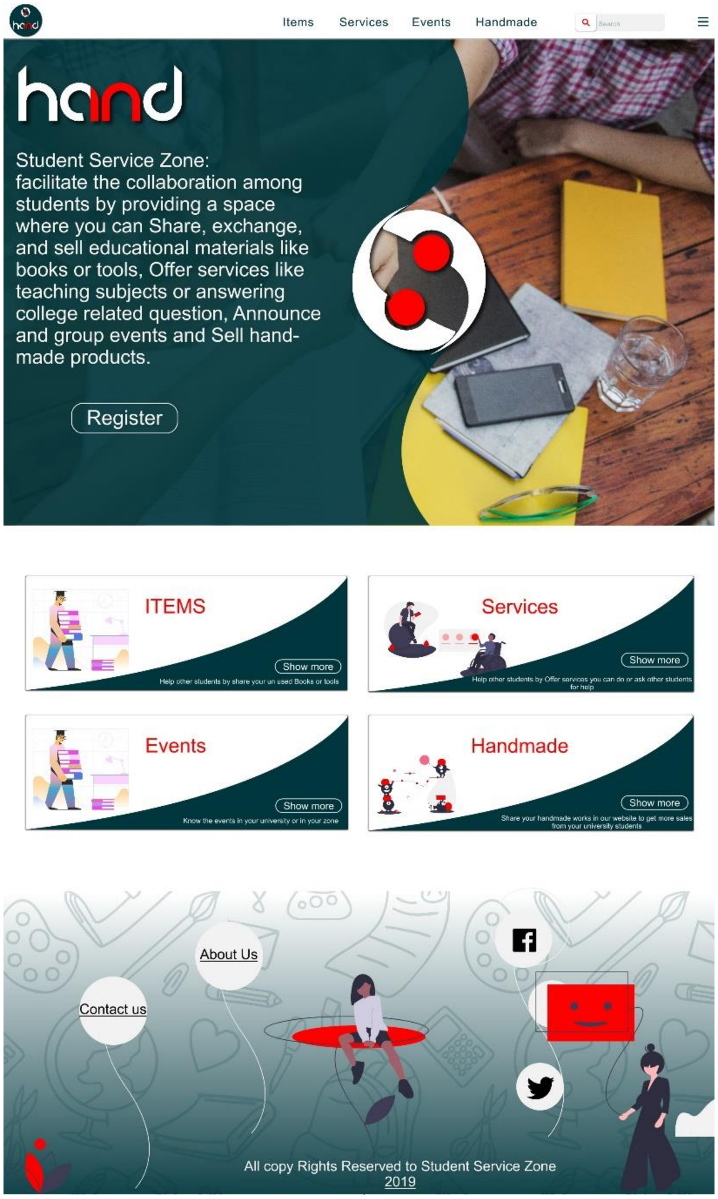
Student Service Zone:
facilitate the collaboration among students by providing a space where you can Share, exchange, and sell educational materials like books or tools, Offer services like teaching subjects or answering college related question, Announce and group events and Sell hand-made products.

Figure 4: first part of the landing page

We choose an image which will give the meaning of helping and donation so when the student opens the web site, he will see it as the first page. That **will leave a good feeling** in it to continue Login or register and help other students.

Also, in the landing page we used the space efficiently to show the main components of the project, and at the top of the website the **navigation panel** of the website which the user can navigate easily from one page to another. Also, we show one of the important functions of our website which is search. **Search function** will let the user search through the whole items in our website using keyword so it's a good point to show this important function in the navigation panel to make it easily to access and use with the minimum effort.

As it shown in **figure 5** the navigation panel also contains a menu that will let the user navigate easily to his/her profile and logout if he/she is logged in or login or register if he/she is not logged in. So, it's awesome that we show most of the project function in the landing page it gives the user quick overview about us and what the service we are introduce.



Student Service Zone:
facilitate the collaboration among students by providing a space where you can Share, exchange, and sell educational materials like books or tools, Offer services like teaching subjects or answering college related question, Announce and group events and Sell hand-made products.

Help other students by share your un used Books or tools

Help other students by Offer services you can do or ask other students for help

Know the events in your university or in your zone

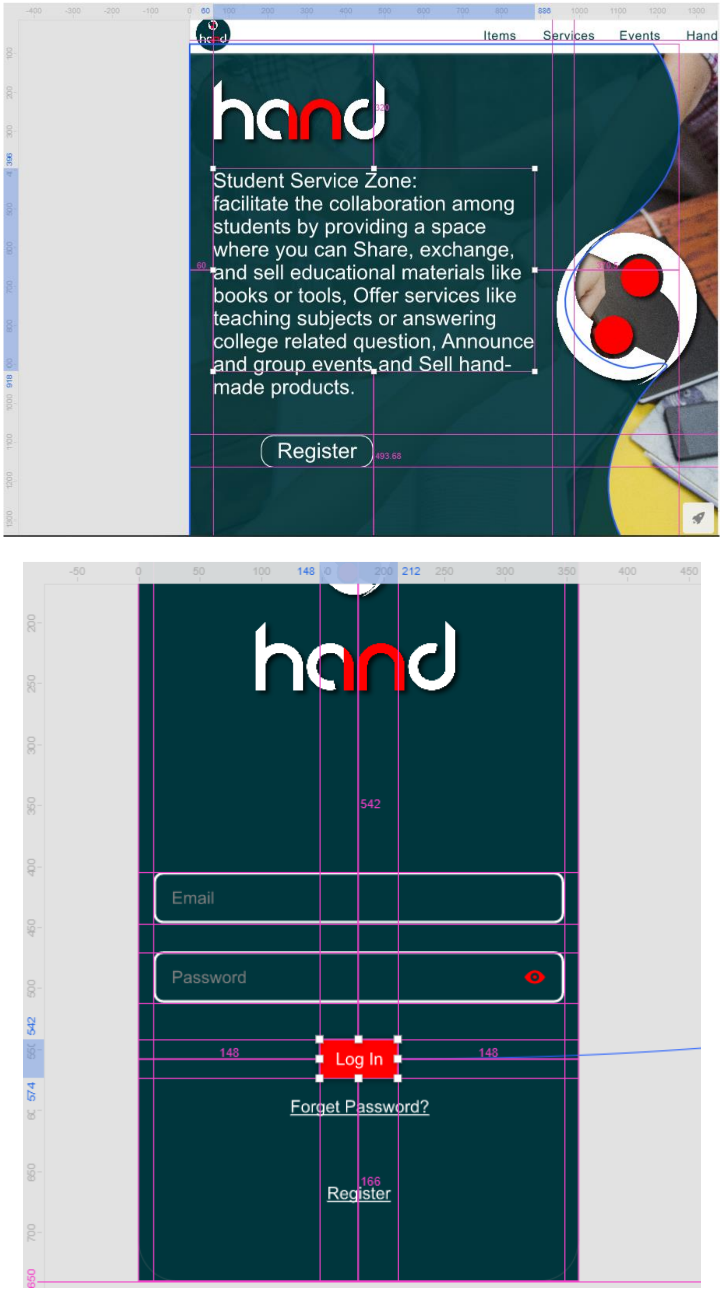
Share your handmade works in our website to get more sales from your university students

All copy Rights Reserved to Student Service Zone 2019

Figure 5: the whole landing page

We aren't just spread the items or the images in the page we used InVision studio to design the UI and we used the margins and paddings in our mind to make it easy to the developers team to just take the dimensions from the prototype. We followed the material design guidelines of Google to design the prototype of the website and the application

As it shown in **figure 6** it appears that each item takes a specified place carefully and we also give each item an enough space to let the user feel satisfied with the usage of the white space:



Student Service Zone:
facilitate the collaboration among students by providing a space where you can Share, exchange, and sell educational materials like books or tools, Offer services like teaching subjects or answering college related question, Announce and group events and Sell hand-made products.

Figure 6: margins and padding of each item

In the android app there is no landing page but there is a splash screen and it’s designed as the best practice of the android documentation. It does not let the user wait for a time, but it appears once the application icon is clicked as it shown in figure 7 below:

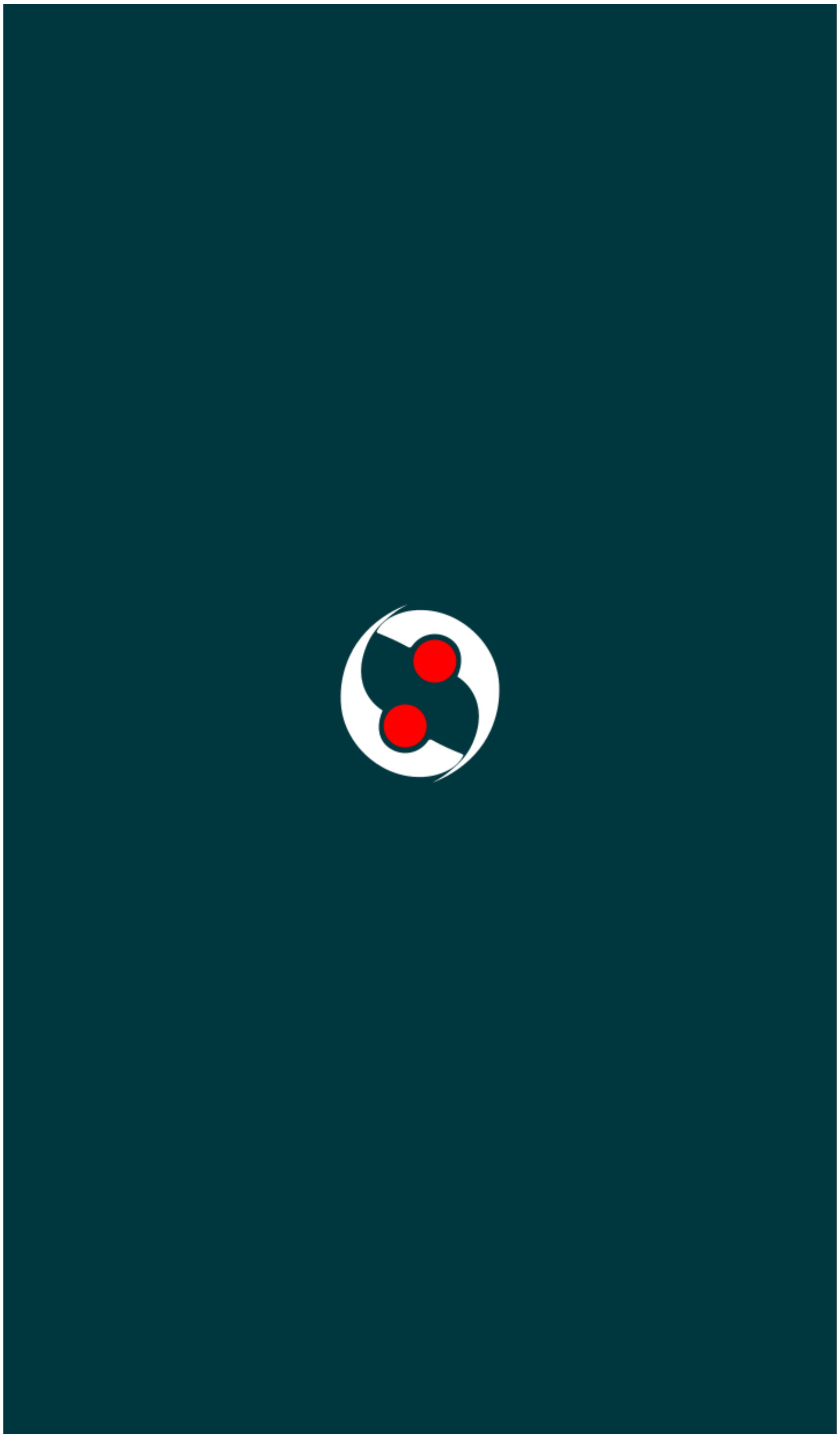
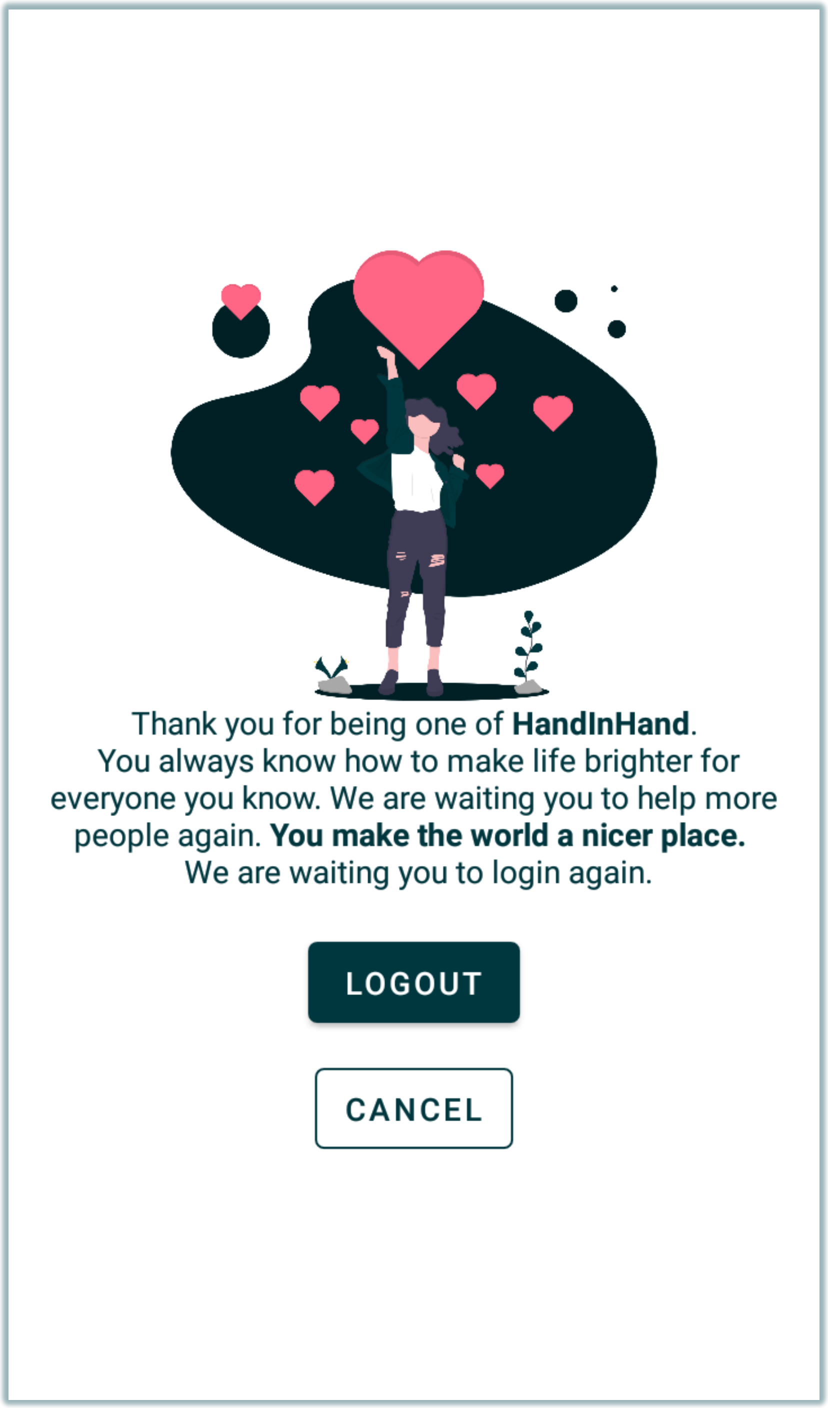


Figure 7: splash screen of the android app

As we mention at the beginning of the UI section, we need to encourage students donate to each other without any emphases. So, in our design we let the user know that he/she does a good thing and **we really appreciate her/his support** and sharing with us. So, when the user want to logout the screen in **figure 8** will be shown, in this logout page we show a message that will let a good feeling in the user that he really did a great job by helping his friend and donate them:



Thank you for being one of **HandInHand**.
You always know how to make life brighter for everyone you know. We are waiting you to help more people again. **You make the world a nicer place.**
We are waiting you to login again.

Figure 8: Logout screen

In **figure 9** there is a navigation graph of the android app that show all main action that will let the user navigate through the pages:

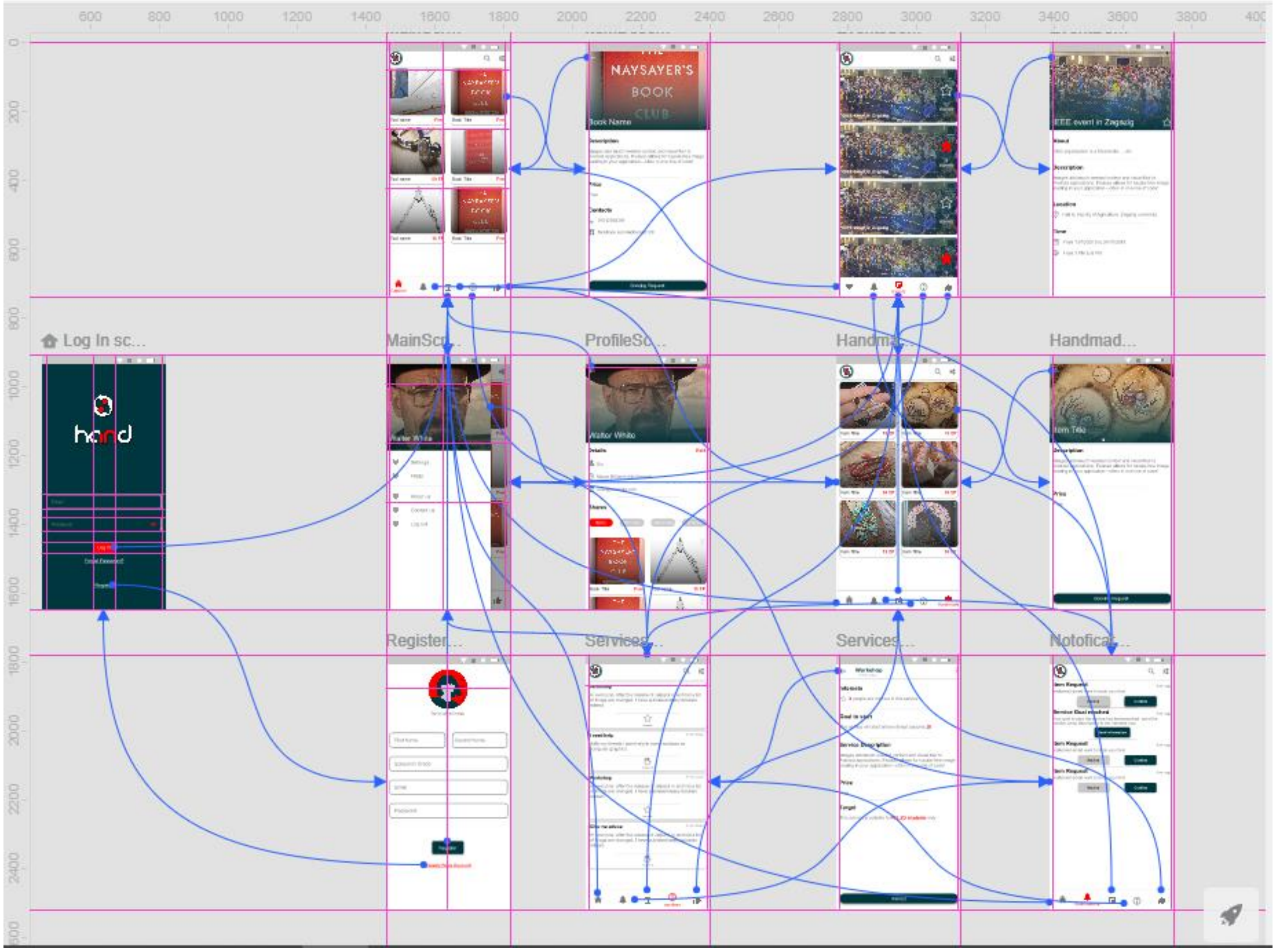
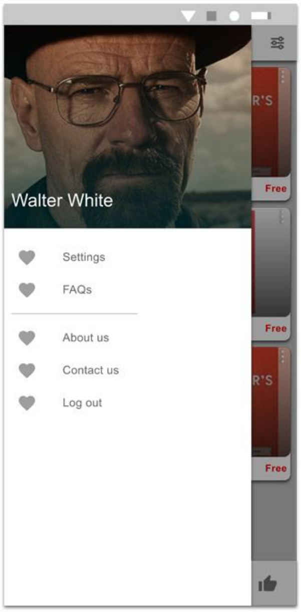
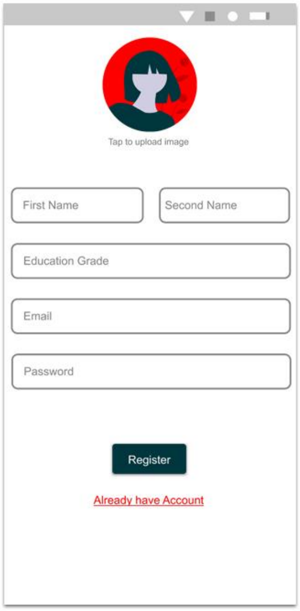


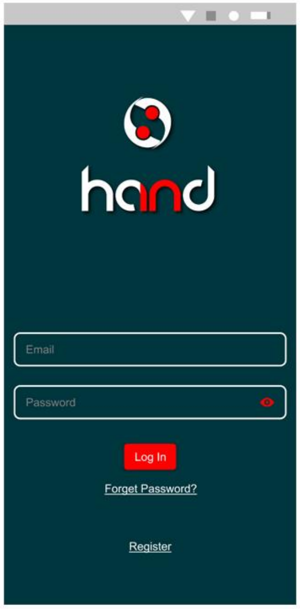
Figure 9: the navigation graph of the android

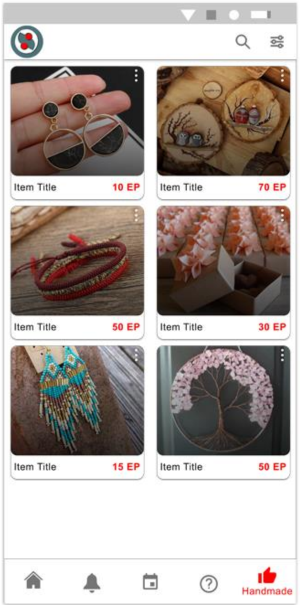
After we list our style in designing the UI, we will list the whole UI of the main parts of the android application:

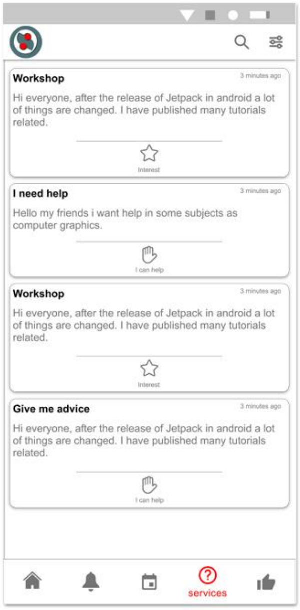
UI OF ANDROID










Workshop 3 minutes ago

Hi everyone, after the release of Jetpack in android a lot of things are changed. I have published many tutorials related.

Interest

I need help 3 minutes ago

Hello my friends i want help in some subjects as computer graphics.

I can help

Workshop 3 minutes ago

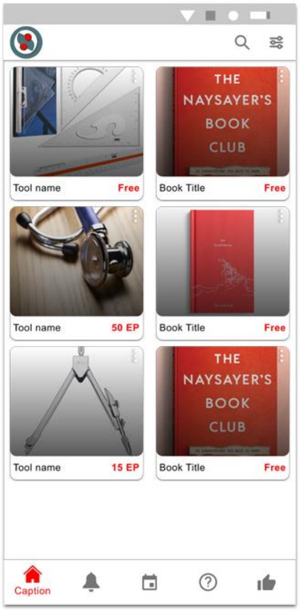
Hi everyone, after the release of Jetpack in android a lot of things are changed. I have published many tutorials related.

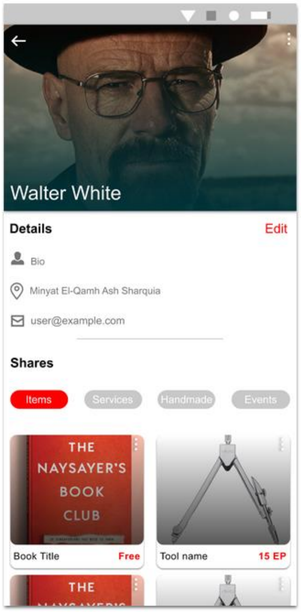
Interest

Give me advice 3 minutes ago

Hi everyone, after the release of Jetpack in android a lot of things are changed. I have published many tutorials related.

I can help





Walter White

Details Edit

Bio

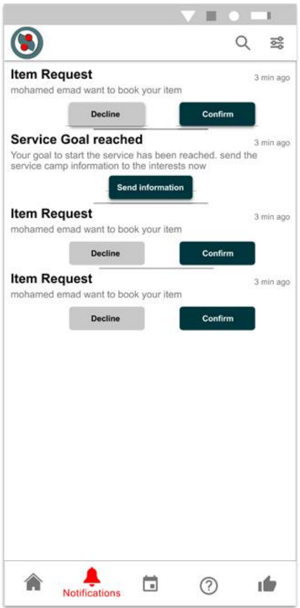
Minyat El-Qamh Ash Sharquia

user@example.com

Shares

Book Title Free

Tool name 15 EP



Item Request 3 min ago

mohamed emad want to book your item

Service Goal reached 3 min ago

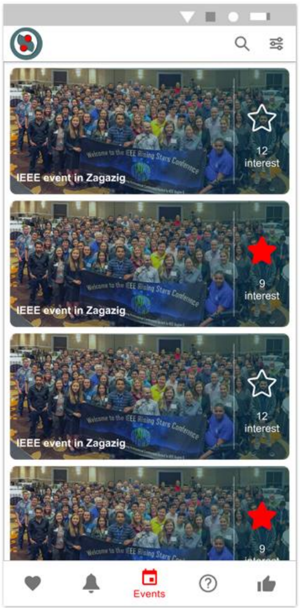
Your goal to start the service has been reached. send the service camp information to the interests now

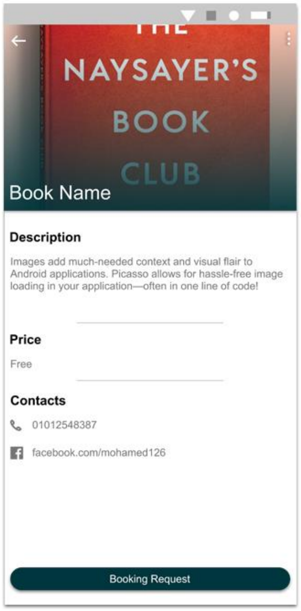
Item Request 3 min ago

mohamed emad want to book your item

Item Request 3 min ago

mohamed emad want to book your item





Book Name

Description

Images add much-needed context and visual flair to Android applications. Picasso allows for hassle-free image loading in your application—often in one line of code!

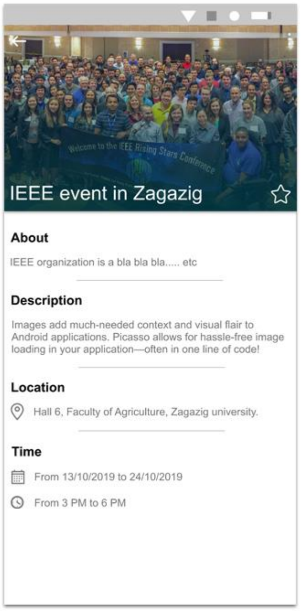
Price

Free

Contacts

01012548387

facebook.com/mohamed126



IEEE event in Zagazig

About

IEEE organization is a bla bla bla..... etc

Description

Images add much-needed context and visual flair to Android applications. Picasso allows for hassle-free image loading in your application—often in one line of code!

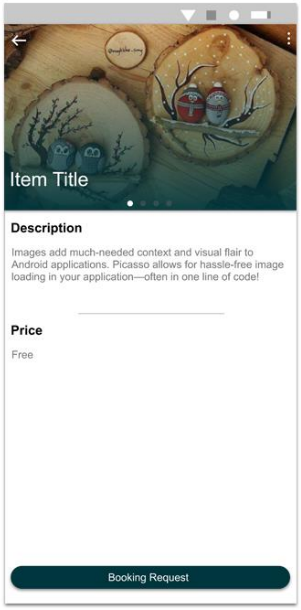
Location

Hall 6, Faculty of Agriculture, Zagazig university.

Time

From 13/10/2019 to 24/10/2019

From 3 PM to 6 PM



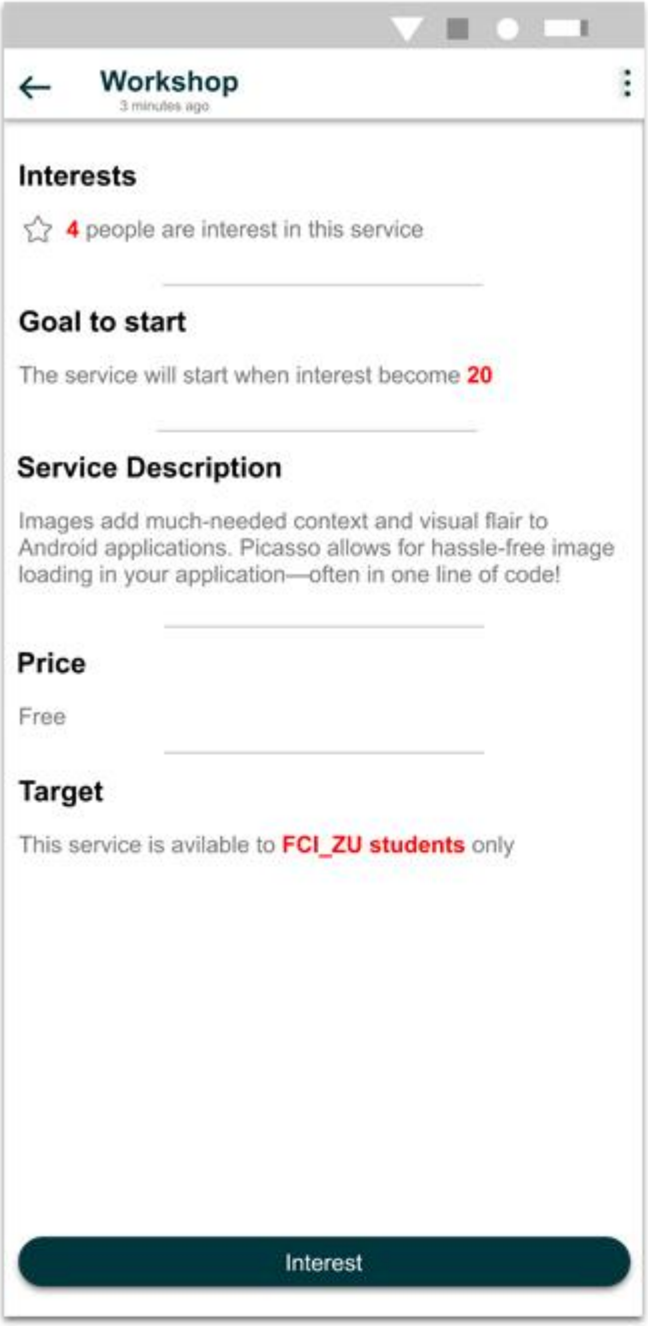
Item Title

Description

Images add much-needed context and visual flair to Android applications. Picasso allows for hassle-free image loading in your application—often in one line of code!

Price

Free



Interests

☆ 4 people are interest in this service

Goal to start

The service will start when interest become 20

Service Description

Images add much-needed context and visual flair to Android applications. Picasso allows for hassle-free image loading in your application—often in one line of code!

Price

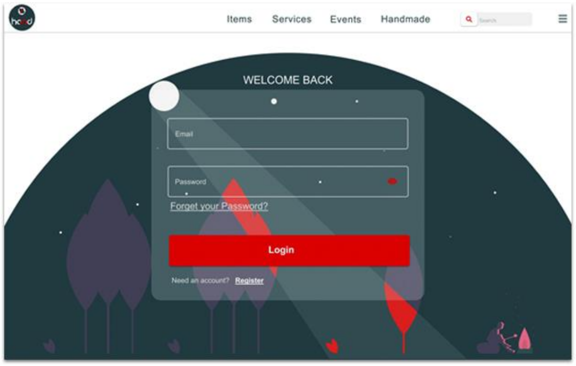
Free

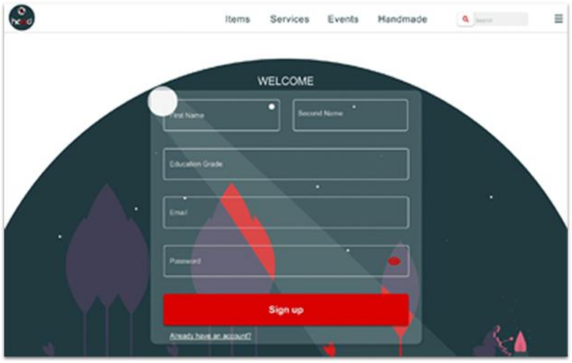
Target

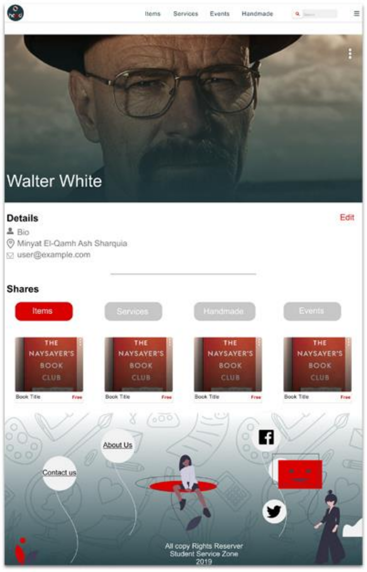
This service is avilable to **FCI_ZU students** only

After we list our style in designing the UI in android, we will list the UI of the website that shows the main parts of the **website**:

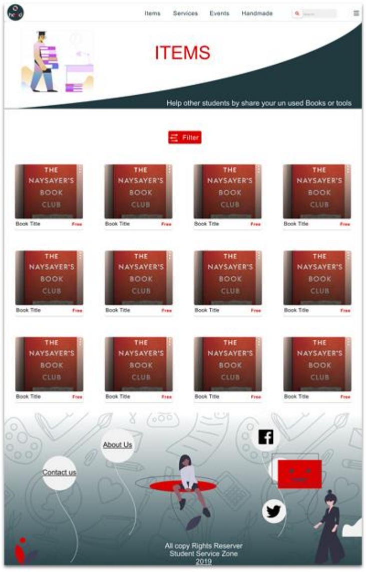
UI OF Website





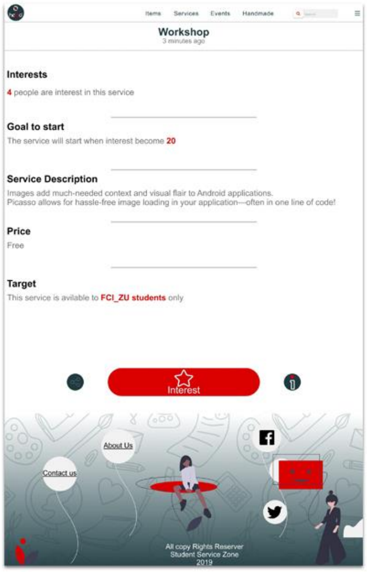


All copy Rights Reserver
Student Service Zone
2019



Help other students by share your un used Books or tools

All copy Rights Reserver
Student Service Zone
2019



Workshop
3 minutes ago

Interests
4 people are interest in this service

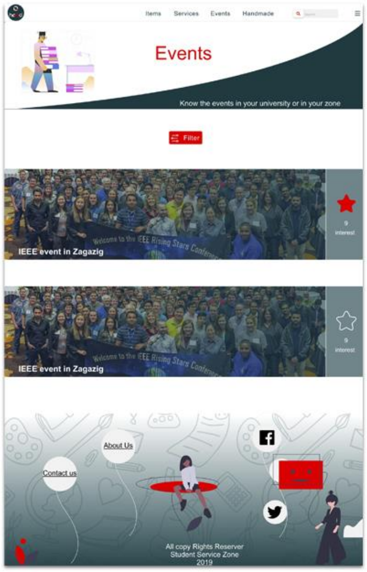
Goal to start
The service will start when interest become 20

Service Description
Images add much-needed context and visual flair to Android applications.
Picasso allows for hassle-free image loading in your application—often in one line of code!

Price
Free

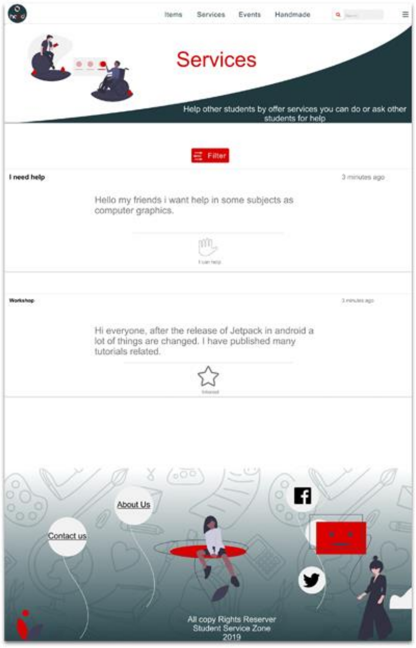
Target
This service is avilable to FCI_ZU students only

All copy Rights Reserver
Student Service Zone
2019



Know the events in your university or in your zone

All copy Rights Reserver
Student Service Zone
2019



Help other students by offer services you can do or ask other students for help

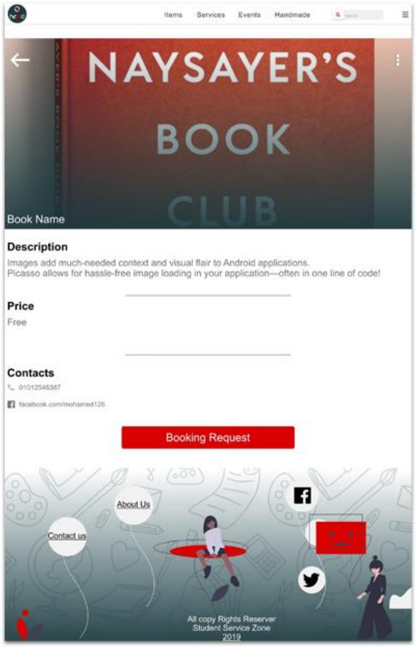
I need help
3 minutes ago

Hello my friends i want help in some subjects as computer graphics.

Workshop
3 minutes ago

Hi everyone, after the release of Jetpack in android a lot of things are changed. I have published many tutorials related.

All copy Rights Reserver
Student Service Zone
2019

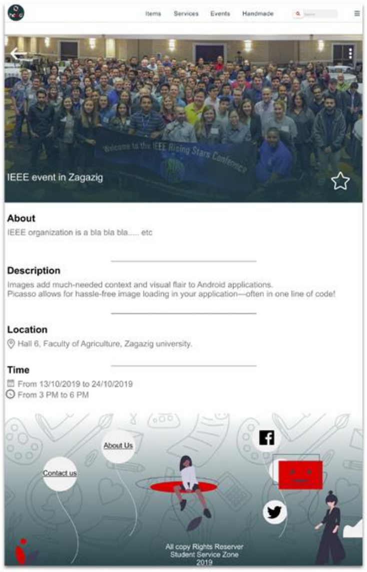


Description
Images add much-needed context and visual flair to Android applications.
Picasso allows for hassle-free image loading in your application—often in one line of code!

Price
Free

Contacts
☎ 01012548387
facebook.com/mohamed126

All copy Rights Reserver
Student Service Zone
2019



IEEE event in Zagazig

About

IEEE organization is a bla bla bla..... etc

Description

Images add much-needed context and visual flair to Android applications.
Picasso allows for hassle-free image loading in your application—often in one line of code!

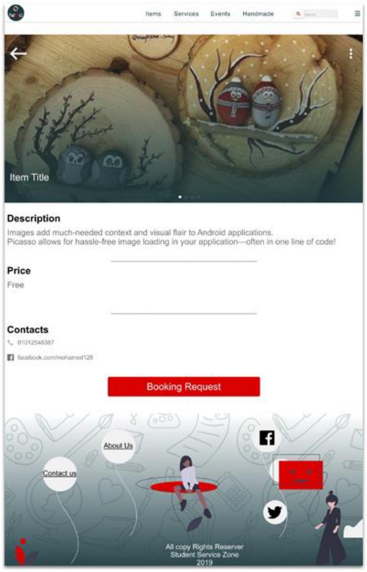
Location

Hall 6, Faculty of Agriculture, Zagazig university.

Time

From 13/10/2019 to 24/10/2019
From 3 PM to 6 PM

All copy Rights Reserver
Student Service Zone
2019



Item Title

Description

Images add much-needed context and visual flair to Android applications.
Picasso allows for hassle-free image loading in your application—often in one line of code!

Price

Free

Contacts

☎ 01012548387
facebook.com/mohamed126

Booking Request

All copy Rights Reserver
Student Service Zone
2019

Data Flow

Up until now we have been talking about the idea, how we came up with it, our plan to turn it to a real-life project and how the user can see and interact with it.

Now let us talk about the idea from development perspective. As mentioned before **the project consists of a web application and android application.** Although both offer the same functionality.

there are two differences:

1. When a user visits the website the first thing that he/she will see is a landing page as shown in Figure 5. This landing page demonstrates what a user can do or expect from the website. In other words, it acts as guide for new users.
   But on the android, there is no landing page the first page or in android terms **Activity** the user will see is the log in page in case the s user is not logged in.

2. On the website there are two types of users guest user and authenticated user with the later having more Features and privileges.
   For example, a guest user has the ability to search different resources like services and items and the ability to view these resources descriptions. However, guest users cannot interact with these resources.
   Meaning they cannot create, update, delete, report or show interest in any resource. All these actions require authentication.
   Guest users can authenticate simply by registering for a new account or log in if they already registered. On the other hand, android has only on types of users which is authenticated user. In order to use the android application, you must be logged in.

Other than these two differences both the website and android application works exactly the same.

Now that we cleared the differences between them let us look at the wide picture of the system then walk through the details.

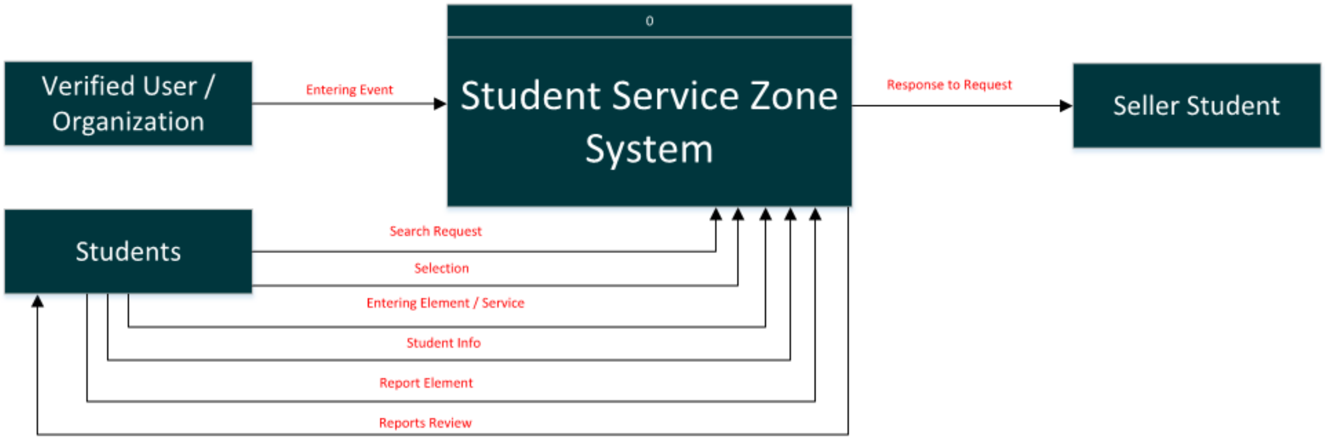


Figure 10: hand in hand Context diagram

Figure 10 represents the context diagram of the system. It demonstrates the major actions that can happen inside the system. These actions are: -

- Create element
- Search for an element
- View an element
- View student info
- Report element
- Requesting item/handmade product from student
- Review report (customer support action)

Note: we use the word element to represent the four types of resources (items, services, events, handmade products) just for the sake of simplicity.

You may have noticed two things in the context diagram: -

1- There are two types of students (Student and seller student) but in fact they are just one type we used the terms student and seller student to differentiate between the student who put an item for sale (seller) and the student who wants to buy the item (buyer).

2- Although the action Add element is mentioned and an element can represent an event there is an explicit action for adding an event with the user doing this called organization or verified user.
That is because we believe that events especially among students and teenagers shouldn't be made by anyone it should be made by selected people that students can trust like college professors, student council and well known organizations like IEEE.
Users who want to create events must provide and evidence that they are part of an organization, professor, etc. after that their accounts will be verified but our support team and they will be able to create events as they want.
Before we dive into the road map of the system here are the three types of users we talked about quickly: -

   - Guest users (web only): they can only browse the web site and view resources, but they cannot interact with them.

   - Authenticated users: these are the users who are logged in. they can view and interact with all the resources except for events. they can only interest in events

   - Verified users/organizations: they have access to all the resources of the project including creating events.

The **level 0 (context diagram)** showed the big picture of the project now **let us dive into the level 1 data flow diagram.** It acts as the road map of the system. It shows all the actions the users can take with more details than the context diagram. See **Figure 11** below to have a better understanding.

Let us describe each process and action the user can take. As mentioned before a lot of actions especially interacting with resources require authenticated users and on android the user must be authenticated to use the app so let us start with the authentication

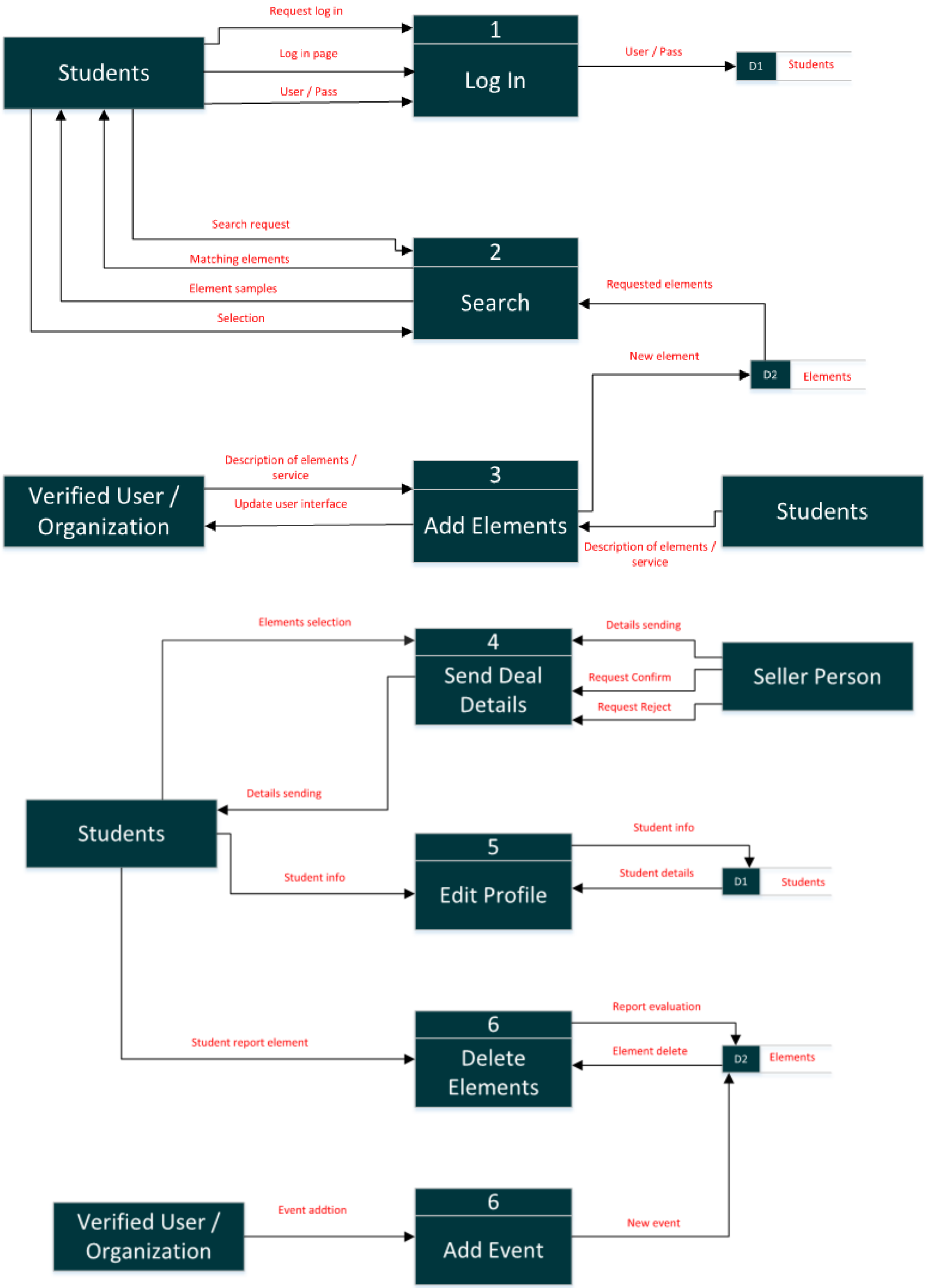


Figure 11: level 1 data flow diagram.

Log in

On the website if a guest user tried to take an action that requires authentication the user will be redirected to the login page. Guest users also can access the login page through the menu located at the rightmost of the navigation bar.

The login page asks the user to enter the email and password. Based on this information the user will be logged in or showed an error message that they entered wrong credentials. If you do not have an account, the login page has a link to the register page where you create an account. You can also access the register page as guest user through the same menu in the navigation bar or a button located in the landing page.

Once you are logged in you will be asked to verify your account. Keep in mind this is not the verification required by users to prove they are trusted organization.it is just an email that will be sent to you to make sure you did not register with a fake email.

For the android it is even easier. You will see the login page as the first page, and it will also have a link to register in case you do not have an account.

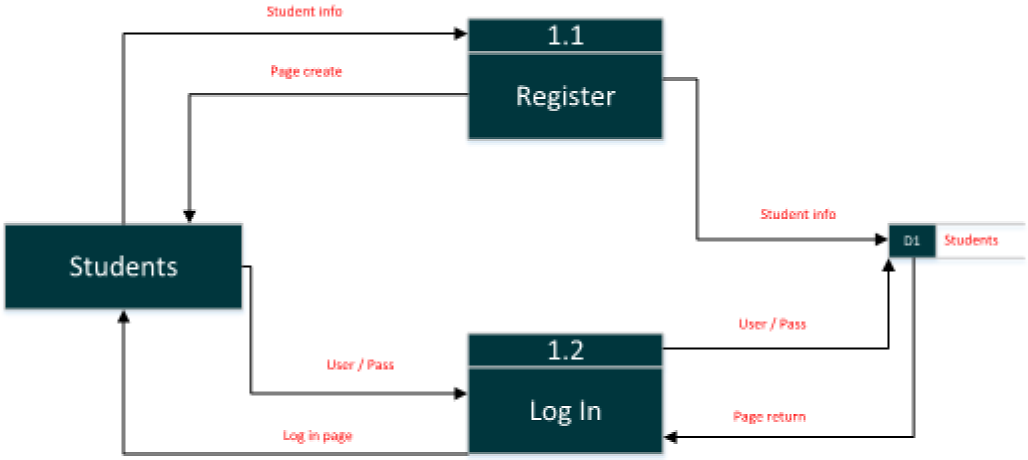


Figure 12: Login process

Search

All users can search for an element. There is a general search box which searches for types of resources. If you typed “something” in this search box it will try to find an (event, service, item or product) which is named “something”.

There are also dedicated search for each resource if the user wants to search only it. With each search request you must enter the element name that you want to search for, but you can also provide more parameters to filter the results of more than one element with the provided name exists. These parameters include price and date both in ascending and descending order.

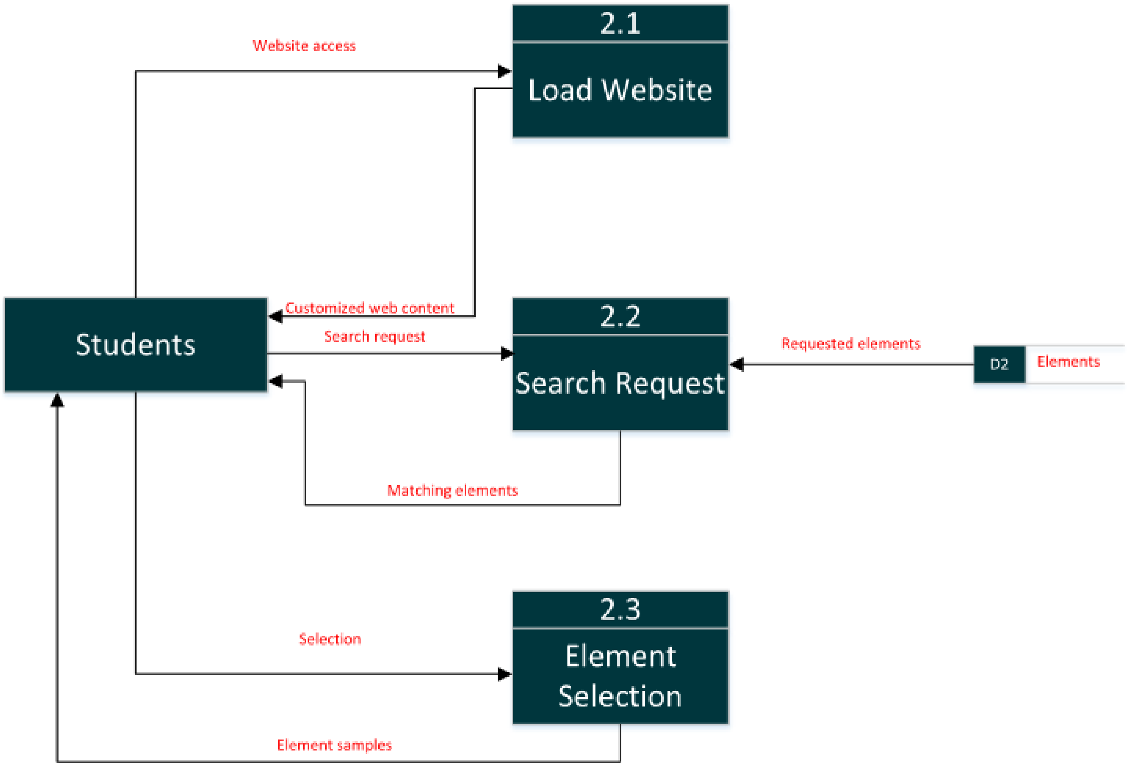


Figure 13: search process

Add element

As an authenticated user you can add three types of resources which are items, services and products with each type having different information to be filled before creating it. Title, price and service content are examples for this information.

After filling in all need information and clicking the adding button the server will validate your request and all other users will see your created resource after validation succeed otherwise you will see or more error messages.

The created resource will also be added to your profile so you can access and modify it with ease.

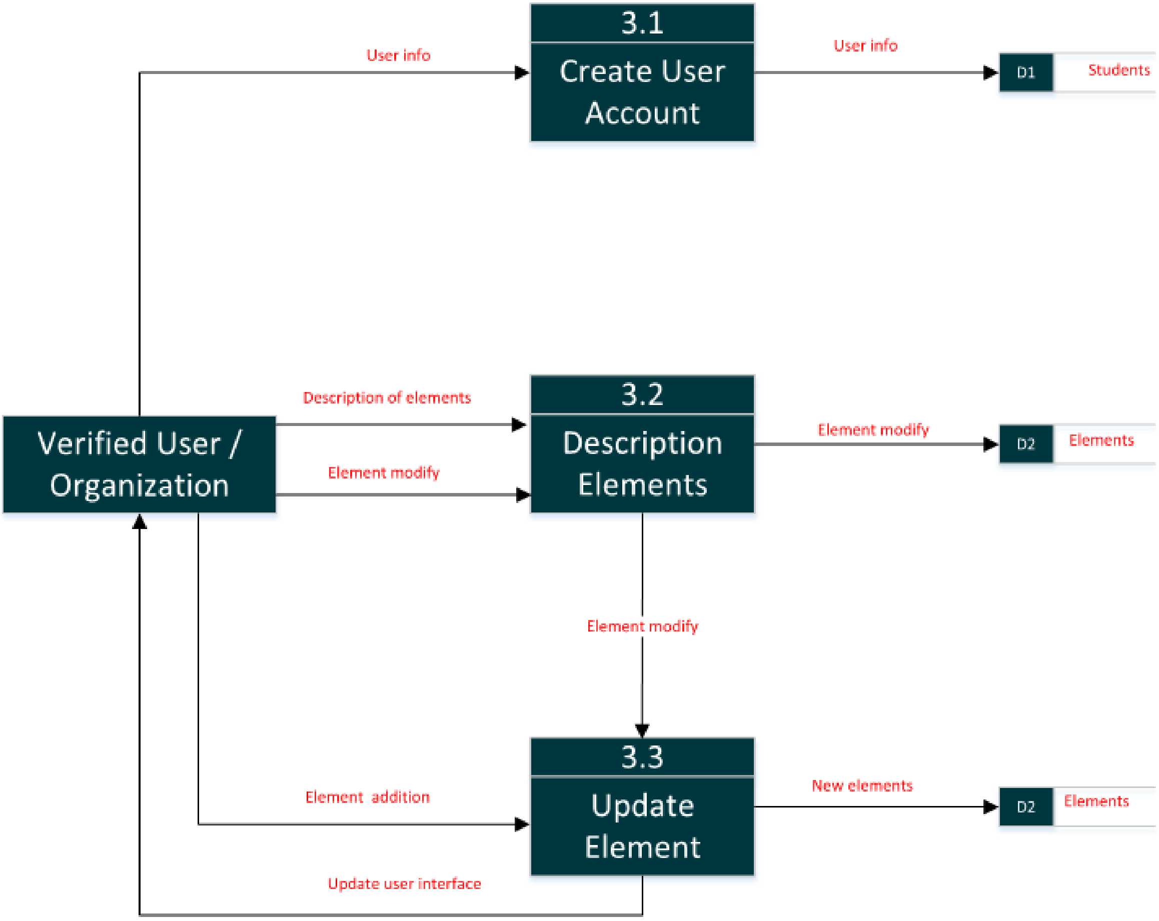


Figure 14: add element process

Send deal details

Once you are interested in an item or a product by clicking the book button the owner will be notified immediately that you want to take this resource.

Resource owners can accept or decline your request. As an owner if you accept a request the user who requested this resource will be notified with some information that you will provide for contacts, meeting location, etc.

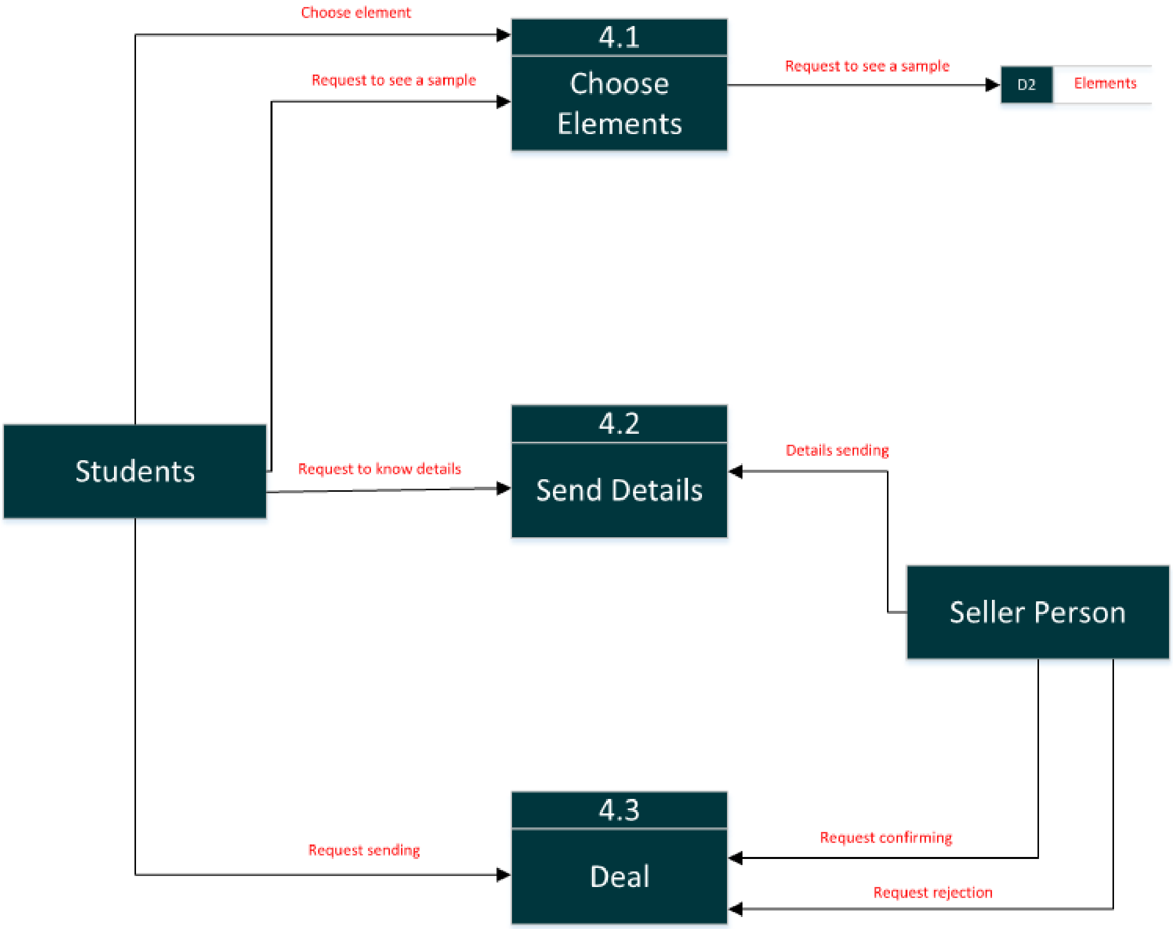


figure 15: send deal details process

Edit profile

You may want to change some information after registering like Name, email or password. All you have to do is to go to your profile page. From there you can access the editing page. You can change all the information you want about yourself but be careful when trying to change your email.

Emails must be unique meaning two users cannot have the same email so you can't update your email with an existing email.

Since all the resources you create will be added to your profile, you can also update any resource you have already created.

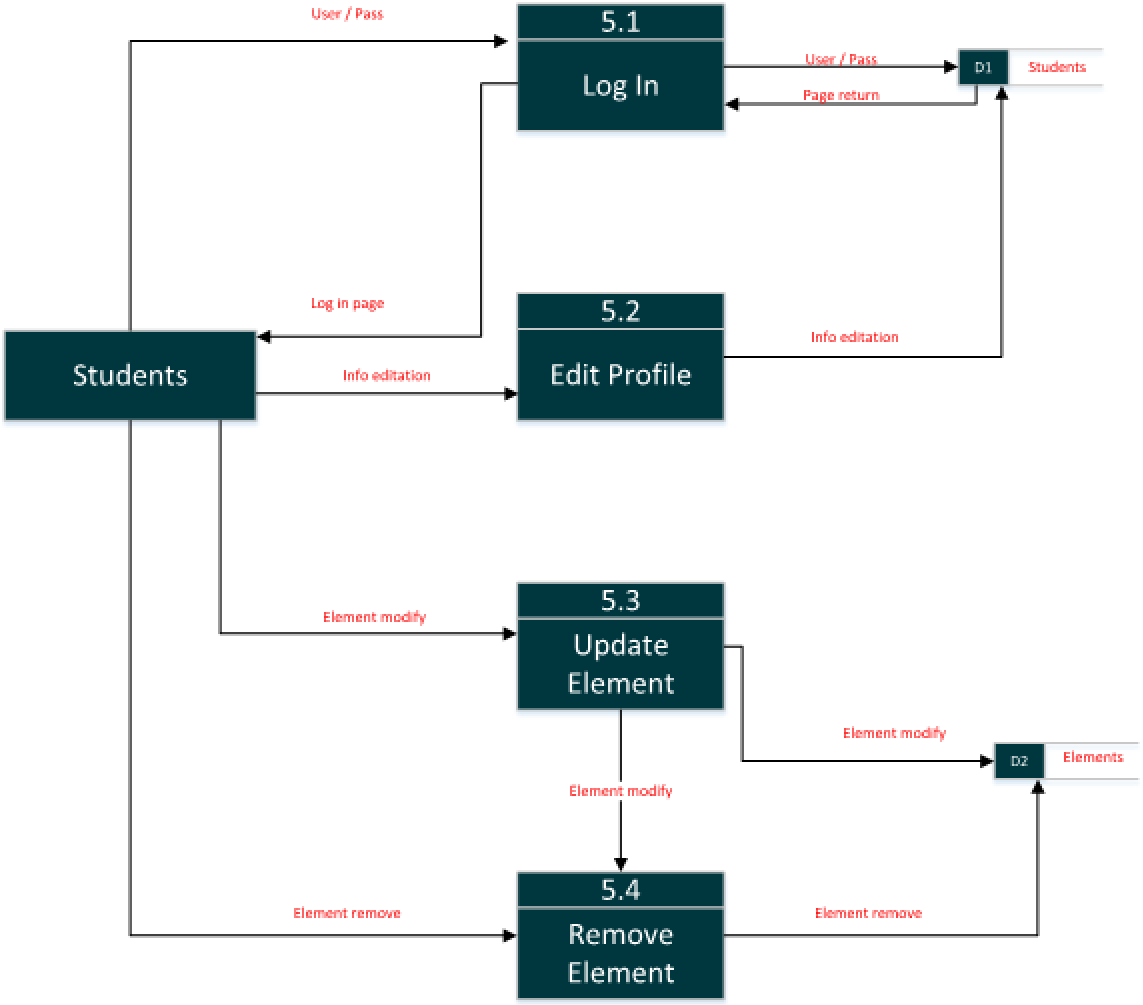


figure 16: edit profile process

Delete element

There are two cases where an element can be deleted: -

1. As the owner of the element you are done with it and want to delete it. In that case you can go to your profile and remove it from there.

2. Other users reported the element. In that case the system will look into the reports if it is proven that the element is a spam or inappropriate for example this element will be deleted.

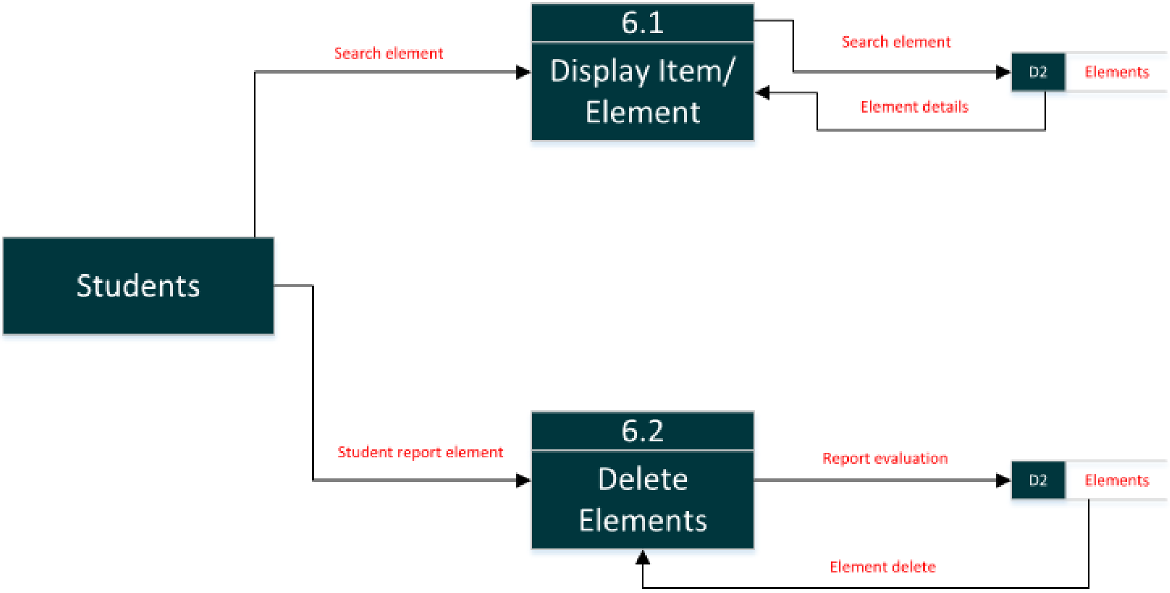


figure 17: delete element process

Add event

Event is one type of resources so adding it is the same as adding an element except for one thing that we talked about before.

You must be a verified user to create an event once you are verified you can create as many events as you like. Just fill in some core information for every event like the location and date and the event is created.

ENVIRONMENT AND TOOLS

These are the tools and the environment that we used during the life cycle of the project. Our choice for the development environments and tools was based on the budget, ease of use, experience and keeping up to date with what is currently used for the market.

Operating system:

- **Windows**: for the availability of all the tools specially designing and prototyping tools

Front end

- Html
- CSS
- Bootstrap
- JavaScript
- Visual studio code

Back end

- PHP (Laravel 6.0)
- Pusher
- MySQL
- MySQL workbench
- Visual studio code

Android

- Java, Java8
- RXJava
- XML
- Android studio

Design and Prototyping

- Invision studio
- Photoshop
- Illustrator

IMPLEMENTATION

At first, we would love to mention that we started developing and learning **from scratch,** most of the developers team started learning the basics such as programming language then the technology itself until we have reached such as this level of coding. It was a long and interesting journey.

It was as a challenge to us to develop such as this system with these great functionality from scratch, we accepted the challenge and started to learn and search from many resources might be we did a little mistakes in the functionality at the beginning but now we gain a great knowledge to develop the whole system in a faster way and avoid the mistakes we did at the beginning.

So, what you will see in the coming implementation parts was made in less than 9 months from scratch until the full software system.

Now we will take about the important part about our project. **Our teams really did a great job in the implementation part.** We did not only implement any thing or just let the user do it, in most cases of the project we did the best practice of the implementation that the documentation of this framework or programming language till us how to do it.

We organize our team in a good way to make it easy to contact with us in the work or changing anything in the project as example of the organization that really helped us in the communication between the backend and the android teams is **Trello website**, we used it as our documentation to work in a parallel process, the backend work and edit the requests and the responses in their code then add these edits in the responses in trello to make it easy to the android team to see the last updates in a real time.

Now let us talk about each part of the implementation project in deep. We will take in our mind if the reader does not know the framework or the programming language and we will try to choose a simple word to make this part of the Documentation be understandable and anyone can read it.

Android

Nowadays mobile phones are a common thing and it becomes a part of our life it makes our life easy and help us with apps and even with the OS itself. So, we think that making an android app for our project will be great and will help us to go on and will help us in the spread of our project and idea.

It is easy to develop and android app and just do some functionality with it. But the most important part it to do these functionalities in an easy way and follow the best practices of google and show a great UI to please the user.

In android development we depend on the Backend in the whole processing and whole operations and functions. In sample words we just communicate with the backend and tell it to do say any function, then the backend tells us “Yeah I did it successfully tell the user”.

It is not simple as I said just to let you know how we communicate each other. The backend team used **PHP** and **Laravel** to create the database and the **API** and we use Trello site to make it a documentation of the Backed so the android team and the backend could see the requests and responses and make it easy to communicate or work together.

We communicate with the backend with **JSON** requests and responses such as we greet each other but in programming. We trying to keep the important and sensitive data of the user in the server side or as we will call it later in the database.

Now let us look at the flow of the user in the android app:

- At first, the user chose to login if he/she has an existing account or just register with new account. Or if the user has an account but the password of his account was forgotten we let the user to re-set the password and we send a reset email link to his/her mailbox.

- Second, and after login or register the user navigate to main content of the app and we give him/her the ability to edit profile and post an item or product or anything else that we mentioned previously in the data flow sections or in the use case section.

- Finally, if the user wants to logout, we show an appreciation message to leave a good feeling in the user and let him/her knows that the world will become a better place with their helping and donations.

Now days there’s many libraries to help us in our way to create an android app these libraries helps us to write a small piece of codes and the library itself will did the work for us the development now focusing on the idea it self so the coding is not a problem now.

In our way to build a great app and take care of the performance and the UI we choose many Open source libraries to help us do a great job with small amount of code too. We chosen these libraries carefully to help us and many of these libraries developed by Google and Square they really did a great job to help android OS to work perfectly now we will list the most important libraries that we used and we will show why we used it and how these libraries helped us:

Retrofit

Retrofit is type-safe REST client for Android and Java which aims to make it easier to consume RESTful web services. It helps us to make a network connections without worrying about the Threads and the body of the request and also the security of the requests itself for example if we send a sensitive information as the user email or password or even the token. And, the library take care of converting from **JSON format to an ordinary java object POJO.**

Retrofit use annotation processor to deal with CRUD operation when we want to communicate with the server these annotations will auto generate the desired code of requests and the responses when we send any request to the server, the annotations that we used is:

```
@GET
@POST
@UPDATE
@DELETE
```

And sometimes we change the methods of the requests in the code.

MVVM

As we said before we are working with the new technologies and choose the most important one know say welcome to MVVM Design pattern in our project. This design pattern is recommended by google it really helps us a lot and save a lot of codes that we could be write.

MVVM is a design pattern that split the view and the models and the data source each of them in a part that work with itself but any changes in any part of these parts will automatically affect the other parts it is really great.

If we did not use it, we would take care of routine things as the lifecycle of the data or anything that could be affected with the activity life cycle. This pattern consist of two main parts is a ViewModel class and the LiveData type.

In the past it was really a nightmare to any android developer to keep the data without changes on the screen and save the state of the data without retrieving it again but with ViewModel it becomes easy so that we choose this pattern.

The ViewModel class does not affected with the activity life cycle and it a great place to keep data and save the states of the UI also. And the life data is the type of the Data that we are keep it is a very clear name. it keeps the data as live until the activity destruction or until this ViewModel cleared.

Navigation Component

The navigation component is a part of Android jet back library and Android team did a really great job to help us with this library. In past we must take care of the backs tack if we used many fragments in one activity and should clear the back stack or replace one fragment with other but with this library it is become a past.

Now the navigation component take care of these things and it just let us where we need to navigate the other things it does for us, and this library also **recommended by google.** The library also gives us a great graph to just drag and drop fragments and tell the library when to navigate just in simple.

MDC

Material Design Component is a great library that Google developed, and it helps us to Draw or design a great UI and make a great animation with it, the MDC library contains many helpful and awesome layouts that helps us to develop the UI design easily.

Absolutely we used more than three libraries or techniques but the **MDC** and **Retrofit** and **MVVM pattern** is the most deservable libraries and the most effective libraries in our app. After you got a snap of how the android works and communicate with backend let us dive into **the most interesting part of the project (Coding)**, and show how we code it and take in our mind the Security of user information and follow the **best practices** also.

Before we dive into the actual logic of the application **let us discuss about main parts of the application code** that we will use in many processes in the app in many different places in the code:

First, **the network connection** is important in our app the whole application depends on network connections and if we try to send a response with the device is offline it will take a thread to do the request that we already know that it will fail, so in many parts of the code **we check the network connection before performing the requests**, the code below check the network connection:

```
public static int TYPE_WIFI = 1;
public static int TYPE_MOBILE = 2;
public static int TYPE_NOT_CONNECTED = 0;


    public static int getConnectivityStatus(Context context) {
        ConnectivityManager cm = (ConnectivityManager) context
                .getSystemService(Context.CONNECTIVITY_SERVICE);

        NetworkInfo activeNetwork = null;


        if (cm != null) {
            activeNetwork = cm.getActiveNetworkInfo();
        }


        if (activeNetwork != null) {


            if(activeNetwork.getType() ==
ConnectivityManager.TYPE_WIFI)
                return TYPE_WIFI;

            if(activeNetwork.getType() ==
ConnectivityManager.TYPE_MOBILE)
                return TYPE_MOBILE;
        }


        return TYPE_NOT_CONNECTED;
    }
```

Second, **we take care of the security of requests too in the java code we do not send an invalid email** we check if the email is follow the pattern of the email or not so if the user entered “email.com” as his email the app will say error the email is invalid we check the validation of the email with the method below:

```
/**
     * Function that check the validation of the Email
     * @param email text the user type in emailEditText
     * @return true if {@param email} match the pattern of Email
address.
     */
    public static boolean isEmailValid(CharSequence email){
        return (!TextUtils.isEmpty(email) &&
Patterns.EMAIL_ADDRESS.matcher(email).matches());
    }
```

Also, we check the empty strings and empty cells before sending any process or request to the server to save the battery and threads and to improve the performance of the app.

Now, let us discuss how we use the power of View model in our application and in our code to improve the performance. As we mentioned before view model does not affect by activity life cycle and it will end only when the activity destroyed. So we use this future will to improve the performance of network requests and responses we did these network operations in the view model and we observe the changes in the data then send these new data to the activity or the fragment. This is our main strategy to deal with MVVM pattern is to split the views and the business logic and the data logic each of them in an isolated package.

Let us now show how we use the view model in the android at first, we define a class that extends from the view model main class ViewModel:

```
public class ProfileViewModel extends ViewModel {
…. . .
}
```

now the class ProfileViewModel will **survive during the whole activity lifecycle** or until the activity will be destroyed so it seems that this class is a perfect place to do the network operations.

In each ViewModel class we use some LiveData variables to observe the changes in the responses status or in the data of the responses to change the views in the fragments. Let us say an example, if the user navigates to the profile fragment to see his/her information. Now we send a request to server but if the request take too long to performed or there is an error in the server what we will show to the user now? Because of that we use a **MutableLiveData** objects to see if the request still loading or there is an error happened, and with these objects we can observe their values and show a progress bar for example or an error message. For your information, each fragment has a specific ViewModel to it so there is many ViewModels in the project as many as the fragments we have, now let us see how we define these objects in the ViewModel class:

```
     private MutableLiveData<Boolean> isLoading = new
MutableLiveData<>();
    private MutableLiveData<Boolean> isError = new
MutableLiveData<>();
    private MutableLiveData<Profile> mProfile;

    public LiveData<Boolean> getIsLoading() {
        return isLoading;
    }

    public LiveData<Boolean> getIsError() {
        return isError;
    }

    public void setIsError(Boolean isError) {
        this.isError.postValue(isError);
    }
public LiveData<Profile> getProfile(String token){
        if(mProfile == null){
            mProfile = new MutableLiveData<>();
            loadProfile(token);
            return mProfile;
        }
        isLoading.postValue(false);
        return mProfile;
    }
```

the above code will be repeated in any ViewModel class we will create the only part that will be changed is the Profile POJO model and absolutely it is name mProfile and the getter of this object the method we do the requests with which is called above `loadProfile(token)`.

The ViewModel now is ready to work as we expected. The methods in the ViewModel where we do the network operations such as the above method `loadProfile(token)` is our target now to show how we deal with retrofit in the view model.

The code below shows one of the methods that we deal with it in the ViewModel and what will be in it to send the requests to the server side and also you cans see how and when we change the values of the MutableLiveData objects that we have to take care of:

```
private void loadProfile(String token) {
        ProfileClient client =
RetrofitApi.getInstance().getProfileClient();
        Call<Profile> call = client.getProfile(token);
        isLoading.postValue(true);
        call.enqueue(new Callback<Profile>() {
            @Override
            public void onResponse(Call<Profile> call,
Response<Profile> response) {
                if(response.isSuccessful()){
                    Profile profile = response.body();

                    if(profile.getDetails().getUser() != null){
                        mProfile.postValue(profile);
                        isError.postValue(false);
                    }
                }
                else{
                    isError.postValue(true);
                    isLoading.postValue(false);
                }
            }

            @Override
            public void onFailure(Call<Profile> call, Throwable
t) {
                isError.postValue(true);
                isLoading.postValue(false);
            }
        });

    }
```

in these methods we define our retrofit object to do the network operations with it we define our clients of the retrofit in the `RetrofitApi` class, the RetrofitApi class is a singleton class that contains all our clients that have all the routs we will use below we will show the code of the `RetrofitApi` class:

```
public class RetrofitApi {
    private static final String BASE_URL =
"http://59cbcc73.ngrok.io";
    private static RetrofitApi ourInstance = new RetrofitApi();
    private static Retrofit retrofit = null;
    private static MainActivityClient client;
    private static ProfileClient profileClient;
    private static ItemsClient itemsClient;
    private static EventsClient eventsClient;

    public static RetrofitApi getInstance() {
        if(ourInstance == null){
            ourInstance = new RetrofitApi();
        }
        return ourInstance;
    }

    private RetrofitApi() {

        HttpLoggingInterceptor logging = new
HttpLoggingInterceptor();
        logging.setLevel(HttpLoggingInterceptor.Level.BODY);
        OkHttpClient clientLog = new OkHttpClient.Builder()
          .addInterceptor(logging)
          .build();
        retrofit = new Retrofit.Builder()
                            .baseUrl(BASE_URL)

.addConverterFactory(GsonConverterFactory.create())
                            .client(clientLog)
                            .build();
        client = retrofit.create(MainActivityClient.class);
        profileClient = retrofit.create(ProfileClient.class);
        itemsClient = retrofit.create(ItemsClient.class);
        eventsClient = retrofit.create(EventsClient.class);
    }
```

and we access the clients in the same class with the getter methods because we define their objects as private:

```
public EventsClient getEventsClient() {
        return eventsClient;
    }

    public ItemsClient getItemsClient() {
        return itemsClient;
    }

    public MainActivityClient getMainActivityClient() {
        return client;
    }
    public ProfileClient getProfileClient() {
        return profileClient;
    }
}
```

if you noticed that we receive all objects in a `Response<Object> response` where the object is the POJO model of the JSON Response or request, we do that to use the features of retrofit to tell us if the response was completed successfully or not and also we call the retrofit object using

`Call<POJOObject>`

then

`call.enqueue`

in android it is not allowed to do network operations in the Main thread so we define a call to use a retrofit then we enqueue it that means we tell retrofit to do it in a background thread.

In some fragments we will need to upload file or image to the server here is how we upload the image to the server, we use the @MultiPart annotation from retrofit to upload images or files, it might be little tricky, but it is work fine and upload the image with its full resolution:

```
public static MultipartBody.Part prepareFilePart(Context
context, String partName, Uri fileUri) {
        File file = FileUtils.getFile(context, fileUri);

        RequestBody requestFile =
                RequestBody.create(
                        MediaType.parse(Objects

.requireNonNull(context.getContentResolver().getType(fileUri))),
                        file
                );
        return MultipartBody.Part.createFormData(partName,
file.getName(), requestFile);
    }
```

We used navigation component to help us in navigation from fragment to another with just some lines of code and also the navigation layout has an editor that are supported in android studio so it is a good feature of the app to use it, we define it in the XML tag as following:

```
<navigation
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/nav_graph"
    app:startDestination="@id/loginFragment">
….
</navigation>
```

The start destination is the start point of the app and the fragment that will be the first to be hosted in the activity. We define fragments inside the navigation XML tag as following:

```
<fragment
        android:id="@+id/loginFragment"

android:name="com.example.handinhand.UI.Fragments.MainActivityFragment
s.LoginFragment"
        android:label="fragment_login"
        tools:layout="@layout/fragment_login" >
        <action
            android:id="@+id/action_loginFragment_to_registerFragment"
            app:destination="@id/registerFragment"

            app:enterAnim="@anim/anim_slide_in_right"
            app:exitAnim="@anim/anim_slide_out_left"
            app:popEnterAnim="@anim/anim_slide_in_left"
            app:popExitAnim="@anim/anim_slide_out_right"
            />
    </fragment>
```

And we add the action to each fragment to which fragment it will be navigate and we decide when the action will be performed in the java code.

Now let us discuss the main operations and how we deal do it in the code.

We the application code depends on the previous pieces of codes, so now we will show a little process how it works and how we use these components together to achieve the desired functionality, and the whole other functions or process will be the same with some little changes in the code we use to achieve its functionality lets discuss the main parts of the login process deeply then take a look on the add element process then the whole other functionality will be the same with the codes we showed previously.

Let us have a look on how the **login process** works in the code and how we deal with the security of information of the user that we send from the mobile app to the server and vice versa.

At first, if the user has an account, he just login to the application once. As we mentioned previous, **we tried to code as the best practice** of the android guideline so nowadays it is a best practice to let the user login just once and use the main content of the app forever with this login until he/she logout. if the user has not an account he must register with email and password.

As we mentioned that we used trello as a documentation of the requests and responses between the backend and the android team. The server expects a JSON request in this format:

```
 {
   “email”: “example@gmail.com”,
   “password”: “itisatrap”
 }
```

To tell us if the operation completed successfully or there is a problem like this response:

```
{
    "login": {
        "token": "sometoken"
    },
    "status": true,
    "error": null
}
```

As we mentioned that we used retrofit library to send this request and the library will convert the model of the login process to JSON format while sending it we send the requests with retrofit like this:

```
@POST("api/login")
    Call<LoginResponse> login (
            @Body LoginInfo emailAndPassword
    );
```

The library will deal with the responses and will give it to us as java object to parse it and extract the desired information from it and to know if the operation completed successfully or not.

The LoginResponse above is a **POJO** model we create it and Retrofit convert it to a **JSON** response the model was created like below:

```
public class LoginResponse {
    private String error;
    private boolean status;
    private Login login;
    public String getError() {
        return error;
    }
    public void setError(String error) {
        this.error = error;
    }
    public boolean getStatus() {
        return status;
    }
    public void setStatus(boolean status) {
        this.status = status;
    }
    public Login getLogin() {
        return login;
    }
    public void setLogin(Login login) {
        this.login = login;
    }
}
```

If the login process complete successfully the most important part of the code come in to **secure the token of the user and save it in a SharedPreferences** with the package name of the project and with a unique key:

```
public static void saveToken (Context ctx, String token) {
        SharedPreferences sharedPref = ctx.getSharedPreferences(
                ctx.getString(R.string.token_shared_preference),

                              Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPref.edit();
        editor.putString(ctx.getString(R.string.token), token);
        editor.apply();
    }
```

We till the android OS to save this key as unique to our app and save it with this key and we give the OS a unique key that we will not mention it as a security of the app.

If the login operation failed, we till the user to check if the email and password are correct or not. We do not let the user confused much time if there is a problem with the internet as there is no internet or the internet time out we till the user that you cannot do this process due to no internet or something went wrong if the request time out or there is a fixes in the server side.

We can say that the Register process and forget password and reset password all of them are part of the login process. So, let us discuss the register process too.

Now you know the format of the JSON response come to us from the server all of them like the login except the login object it will be register or forget password or anything but there is a static part is the status and the error.

The register process contains uploading image a file. The file is the image of the user if the user chooses an image from the phone, and this the request of the register request:

```
{
“first_name”: ”example”,
“last_name”: ”test”,
“email”: “email@gmail.com”,
“password”: “itisatrap”,
“gender”: ”male”,
“grade”: ”FCI ZU”,
“avatar”: image.png
}
```

And here is how we define the request in our client or interface that retrofit will use:

```
    @Multipart
    @POST("api/register")
    Call<RegisterResponse> register(
            @PartMap Map<String, RequestBody> user,
            @Part MultipartBody.Part image
            );
```

As we mentioned before that the responses of the server are very good and let us know if the process completed successfully or not, the response of the register process will be like that:

```
{
     "Register": {
        "token": "some token"
      },
      "status": true,
      "error": null
}
```

The forget password fragment the same concept of the previous models register and login nothing tricky or new.

The most interesting part of the login process at all is the resetting password, we used a deep link to develop it you cannot open this fragment without using the reset ling that will be sent to your mailbox the schema of the link you will see is:

http://ourproject.com/password/reset/{user_token}?email={user_email}

and when you will click on it in your mailbox the android OS will noticed that our app might be a target with this link because we put it as deep link in our app, we define it in the navigation **XML** of the reset fragment as following:

```
<fragment
        android:id="@+id/resetFragment"

android:name="UI.Fragments.MainActivityFragments.ResetFragment"
        android:label="fragment_reset"
        tools:layout="@layout/fragment_reset" >


        <deepLink
            android:id="@+id/deepLink"
app:uri="http://ourproject.com/password/reset/{user_token}?email
={user_email}" />


        <argument
            android:name="user_email"
            app:argType="string" />
        <argument
            android:name="user_token"
            app:argType="string" />


        <action
         android:id="@+id/action_resetFragment_to_loginFragment"
            app:destination="@id/loginFragment"
            app:popUpTo="@+id/loginFragment" />
```

Now let us look on how **add element** works in our app nothing will be new here but to show that we discussed the most important part before, and you know the main concept of our the coding style and you can imagine the upcoming parts when you read its explanation in the dataflow or the use case.

We add element as in the registration part with multipart request with the image if the element is product or event for example. And here is how we define add item request in the items interface:

```
    @Multipart
    @POST("api/user/{user_id}/item")
    Call<AddItemResponse> addItem(
            @Header("Authorization") String token,
            @Path("user_id") String user_id,
            @PartMap Map<String, RequestBody> item,
            @Part MultipartBody.Part image
    );
```

As in the registration part we upload the image using @MultiPart annotation and also here **we add more security rule,** the user want to add item so he/she must be logged in here we send the user token in the header requests to show that this request is authorized and trusted then the backend knows that this request is not from any tracker or hacker or anything like this.

Another feature we did in the app that we do not consume much requests until the user request it, we mean pagination, in the app the user will see a list of elements if the user did not scroll to the end of these list we did not send another request to the server to show more element as we said that we care about data consumption and the app performance too.

For example, let us see how the pagination works in the items, this the request that we send to the server:

```
@GET("api/items")
    Call<ItemsPaginationObject> getItems(
            @Query("page") int page,
            @QueryMap Map<String, String> Queries
    );
```

The integer value page is the page that we want from the server.

We receive a response that contains a list of items and the current page and the second and the last and another information too as following:

```
{
    "items": {
        "current_page": 1,
        "data": [
                {
                "id": 104,
                "user_id": 51,
                "title": "title",
                "description": "simple",
                "price": "0.00",
                "phone": "01119067038",
                "facebook": "m.m.m",
                "image": "1583883044.jpg",
                "created_at": "2020-03-10 23:30:44",
                "updated_at": "2020-03-10 23:30:44",
                "spam": 0
            },
            .
            .
        ],
        "first_page_url": "http://127.0.0.1:8000/api/items?page=1",
        "from": 1,
        "last_page": 7,
        "last_page_url": "http://127.0.0.1:8000/api/items?page=7",
        "next_page_url": "http://127.0.0.1:8000/api/items?page=2",
        "path": "http://127.0.0.1:8000/api/items",
        "per_page": 16,
        "prev_page_url": null,
        "to": 16,
        "total": 105
    },
    "status": true,
    "error": null
}
```

The ItemsPaginationObject contains an item POJO that contains a list of Data which is the items that we want.

In the recycler view we check if the user reached to the end of the list or not if yes, we send request to the server to request the next page and add it to the list:

```
recyclerView.addOnScrollListener(new RecyclerView.OnScrollListener() {

            @Override
            public void onScrolled(@NonNull RecyclerView recyclerView,
int dx, int dy) {
                super.onScrolled(recyclerView, dx, dy);


                if(dy > 0){ // only when scrolling up

                    final int visibleThreshold = 2;

                    GridLayoutManager layoutManager =
(GridLayoutManager)recyclerView.getLayoutManager();
                    int lastItem  =
layoutManager.findLastCompletelyVisibleItemPosition();
                    int currentTotalCount =
layoutManager.getItemCount();

                    if(currentTotalCount <= lastItem +
visibleThreshold){
                        Toast.makeText(getActivity(), "End",
Toast.LENGTH_SHORT).show();
                        if(page != lastPage ||
(itemsViewModel.getIsLoading().getValue() != null &&

itemsViewModel.getIsLoading().getValue())){
                            itemsViewModel.loadNextPage(page + 1);
                        }
                    }
                }

            }
        });
```

All other requests or fragments will be like the previous parts, nothing new we did not mention, now you know our style in writing the application code also you know the main parts that helps us in writing the whole app you can with the previous parts just read the use case and you will know how we implement it in the code.

Front end

today web design is important to think of our life, that appealing webpage that engage users and allow them to easily navigate information. in order, you secure new clients and get new work from existing customers. that's why in this project we followed the best practices and used modern technologies to provide the users with elegant and smooth experience to encourage them to help each other and progress with ease.
It is easy for the designer to design UI / UX through some HTML, CSS through the codes. This site is designed for the user to interact with him and help him with no problems.

This site must be linked by the backend to use it without any obstacles by activating the buttons and linking the site to the database.

Now let us look at the flow of the user in the site:

- At first, the user chose to log in if he/she has an existing account or just register with a new account. Or if the user has an account but forgot the password, we let the user reset it and we send a reset password link to his/her mailbox.
- Second, and after login or register the user navigate to the main content of the site and we give him/her the ability to edit profile and post an item or product or anything else that we mentioned previously in the data flow sections or the use case section.
- Finally, if the user wants to logout, we show an appreciation message to leave a good feeling in the user and let him/her knows that the world will become a better place with their helping and donations.

To design this website, we use some languages to write codes and functions to make it easier for us and to appear beautifully and comfortably so that he can help the community with better ideals. And from these languages:

Html

HTML (Hypertext Markup Language) is a text-based approach to describing how content contained within an HTML file is structured. This markup tells a web browser how to display text, images, and other forms of multimedia on a webpage.
HTML is a formal recommendation by the World Wide Web Consortium (W3C) and is generally adhered to by all major web browsers, including both desktop and mobile web browsers. HTML5 is the latest version of the specification.

the annotations that we used is:

```
<! DOCTYPE html>
<html Lang="en">
<head>
   <title></title>
   <meta charset="utf-8">
   <link rel="stylesheet" type="text/css" href="">
</head>
<body>
         <img src="">
         <a herf=""></a>
         <h1></h1>
         <p></p>
         <form></form>
         <div></div>
</body>
</html>
```

CSS

Stands for "Cascading Style Sheet." Cascading style sheets are used to format the layout of Web pages. They can be used to define text styles, table sizes, and other aspects of Web pages that previously could only be defined in a page's HTML.

CSS helps Web developers create a uniform look across several pages of a Web site.

the annotations that we used is:

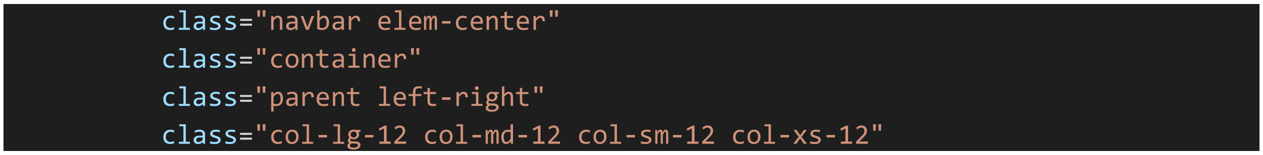
```
width: % ;   height: % ;  position: ;  margin: px;  border: ;

padding: px; background-color: ;   font-size: px ;

font-family: 'Arial', sans-serif;
```

Bootstrap

Bootstrap is a free and open-source front end development framework for the creation of websites and web apps. The Bootstrap framework is built on HTML, CSS, and JavaScript (JS) to facilitate the development of responsive, mobile-first sites, and apps. Responsive design makes it possible for a web page or app to detect the visitor’s screen size and orientation and automatically adapt the display accordingly; the mobile-first approach assumes that smartphones, tablets and task-specific Mobile apps are employees' primary tools for getting work done and addresses the requirements of those technologies in design.

Bootstrap includes user interface components, layouts, and JS tools along with the framework for implementation. The software is available precompiled or as source code.

the annotations that we used is:



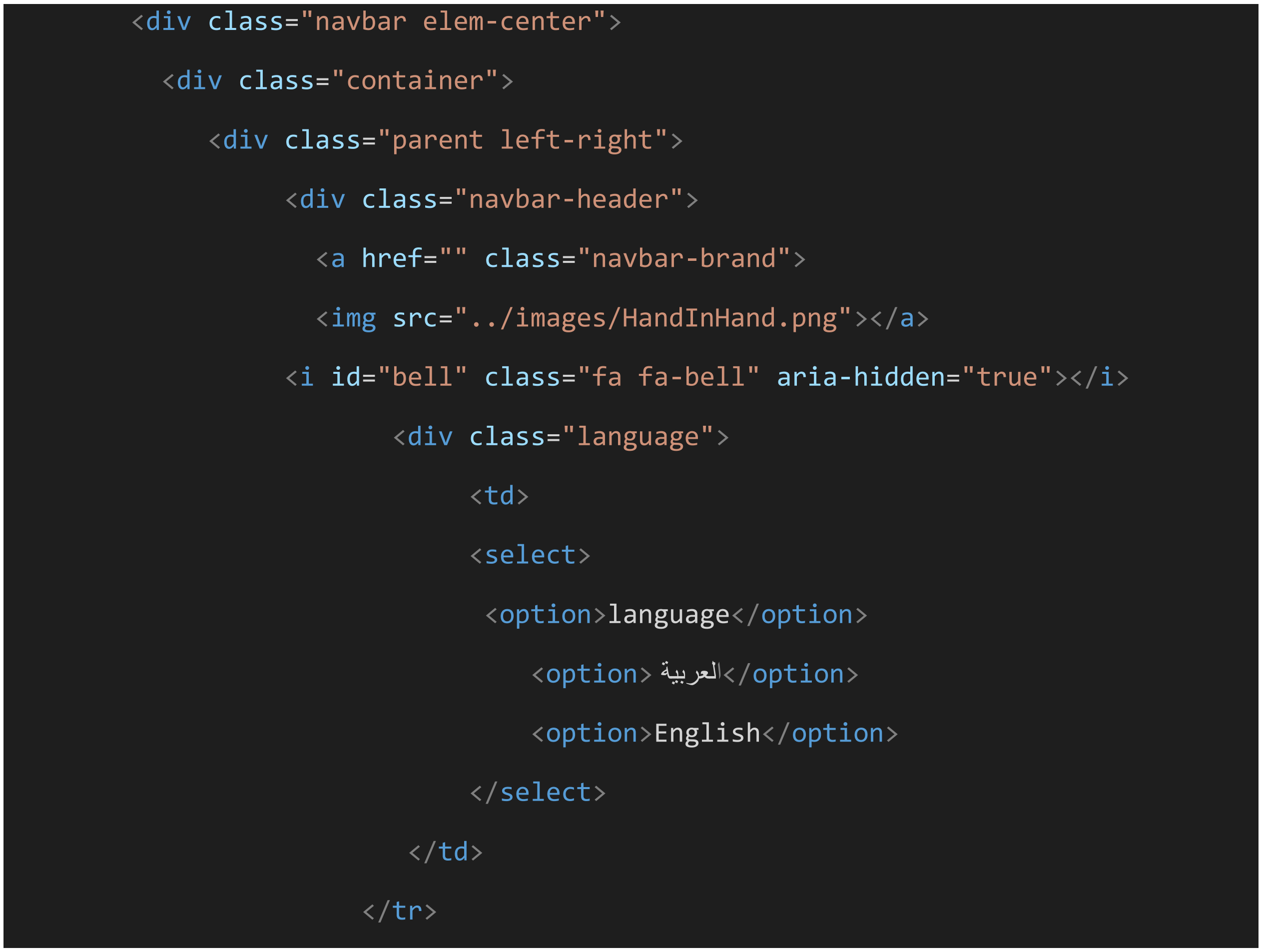
```
          class="navbar elem-center"
          class="container"
          class="parent left-right"
          class="col-lg-12 col-md-12 col-sm-12 col-xs-12"
```

JavaScript

JavaScript is a scripting or programming language that allows you to implement complex features on web pages — every time a web page does more than just sit there and display static information for you to look at — displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc. — you can bet that JavaScript is probably involved. It is the third layer of the layer cake of standard web technologies, two of which (HTML and CSS) we have covered previously.

We also used some of the libraries available online, such as font awesome, so we can add icons on the site and use pictures and SVG.

Navigation bar is the most important part because it consists of logo, services, and notifications, so we will explain it first:



```
        <div class="navbar elem-center">

          <div class="container">

             <div class="parent left-right">

                  <div class="navbar-header">

                    <a href="" class="navbar-brand">

                    <img src="../images/HandInHand.png"></a>

                  <i id="bell" class="fa fa-bell" aria-hidden="true"></i>

                         <div class="language">

                              <td>

                              <select>

                               <option>language</option>

                                  <option> العربية</option>

                                  <option>English</option>

                              </select>

                          </td>

                      </tr>
```

```
                        </div>

                   </div>
```

In the navbar part we put it in the center by using **elem-center**, after that, we divided the navbar into two parts, the first part is the logo and notifications and its language,

```
<i id="bell" class="fa fa-bell" aria-hidden="true"></i>
```

you may call the notification icon from **font awesome**

```
           <option>عربي</option>
       <option>English</option>
```

to facilitate the site for everyone we made it in both Arabic and English languages by using a **select** tag and putting the two options so that you can choose the language that suits you.

```
  <ul class="nav navlist" id="links">

             <li class="active">

             <a href="items.html" data-value="about"
                             class="effect">Items</a></li>

             <li><a href="service.html" data-
                              value="port"class="effect">Services</a></li>

            <li><a href="events.html" data-
                              value="foll"class="effect">Events</a></li>
            <li><a href="handemade.html" data-
                               value="cont"class="effect">Handmade</a></li>

 </ul>
```

The second part, which consists of the services that we provide to help the community and search and the menu, the services are items, services, events and handmade products, and they are linked to the page so that when clicked on it is transferred to the page directly

```
       <form class="navbar-form navbar-right">

            <input type="text" placeholder="Search">
            <i class="fa fa-search"></i>

         </form>
```

In order to help the user with using the site, we used the search text input in the navbar:

```
<input type = "text" placeholder = "Search">
```

It is such as regular input field text type so that the user can type then click the search icon, we used font awesome as placeholder in the input text:

```
<div class="container">

         <div class="row" >

              <div class="col-lg-6 col-md-6 col-sm-12 col-xs-12" >

               <div class="items">

                  <div class="image">

                    <img src="../images/marginalia-education.png"class="ig1">

                    <h3>ITEMS</h3>

                  </div>
                    <img src="../images/path2.png"class="background1">

                    <div class="paragraph">

                           <a href="items.html" data-
                               value="foll"class="showmore">Show more</a>

                           <p>Help other students by share your un used Books or
                                                                        tools</p>
                    </div>

          </div>
</div>
```

As we see the use of

```
<div class="col-lg-6 col-md-6 col-sm-12 col-xs-12" >
```

For us to split the page, this means that when the screen is large or medium, two elements are displayed so they take a value of 6, and when the screen is small or very small, one item is displayed so take a value of 12, and for more information about our services, click Show more and have linked it to the page by

```
      <a href="items.html"
            data-value="foll"class="showmore">Show more</a>
```

A registration: from is a list of fields that a user will input data into and submit it to create an account as individual, for first name, last name. we use in the design Html, CSS and JAVASCRIPT.

In html code:

```
<input class="input1"
                      type="text"
                      name="First Name"
                      required="">
                      <label class="focus-input">
                       First Name </label>
                  </div>
  <input class="input2"

                      type="text"
                      name="Second Name"
                      required="">
                      <label class="focus-input"id="inp2">
                        Second Name</label>
```

In this field it is required for the user to enter his first name   and last name, two users may have the same name it is not a problem because it will not be used to log in:

```
<input class="input4"
                        type="email"
                        name=""
                        required="">
                        <label class="focus-input" >
                        Email</label>
```

The user entered “email.com” and edit an e-mail address. It must be unique, the input value is automatically validated to ensure that it's either empty or a properly-formatted e-mail address, we check the validation of the email with the Function of JavaScript.

```
  function validate (input) {
        if($(input).attr('type') == 'email' || $(input).attr('name') == 'email')
{
```

```
            if($(input).val().trim().match(/^([a-zA-Z0-9_\-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-]+\.)+))([a-zA-Z]{1,5}|[0-9]{1,3})(\]?)$/) == null) {
                return false;
            }
        }
        else {
            if($(input).val().trim() == ''){
                return false;
            }
        }
    }

    function showValidate(input) {
        var thisAlert = $(input). parent ();

        $(thisAlert). addClass('alert-validate');
    }

    function hideValidate(input) {


        var thisAlert = $(input). parent ();

        $(thisAlert). removeClass('alert-validate');

    }
```

We use:

```
(/^([a-zA-Z0-9_\-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-]+\.)+))([a-zA-Z]{1,5}|[0-9]{1,3})(\]?)$/)
```

The problem is that some characters that should not be escaped were escaped, like ' and ^ inside the character classes. Note that - inside a character class may be escaped but does not have to when it is at the start of the email.

We use eye icon ("toggle password “) to hide or show password when editing it, to let us check if we entered the desired password or not.

```
<! --icon eye-->
                           <span toggle="#password-field"
                              class="fa fa-fw fa-eye field-icon toggle-password"></span>
```

to show or hide password, we use JavaScript function and library.

JavaScript function

```
$(document). on ('click', ‘. toggle-password', function () {
        $(this). toggle Class ("fa-eye fa-eye-slash");
        var input = $(‘. input5');
        input. Attr('type') === 'password’? input. Attr ('type’, ‘text'): input.
Attr ('type’, ‘password')
      });
```

Libraries:

```
<script src="../vendor/jQuery/jquery-3.2.1.min.js"></script>
```

Also, we check the empty strings and empty cells before sending any process or request to the server.

We provided users with everything they need for interacting with resources, such as the addition, modification and deletion processes. They can also report resources for being inappropriate or spam. Most of these processes share one thing in common and that is they are using a dialog, let us explain a simple yet an important process, which is the addition process.

Users can add any resource they need by using the add button. By clicking that button, a form will appear in the shape of dialog allowing users to fill in the resource information. This information may include an image, description of the resource, an address, a phone number, a price, a Facebook account, a date, a Goal and an About (for an organization to provide information about itself).

After that the user can press the save button to store the final information in the database.

```
 <! -- Trigger/Open The Modal -->

             <button id="myBtn">Add</button>
                         <! -- The Modal -->
                            <div id="myModal" class="modal” >
                        <! -- Modal content -->
                          <div class="content">
                            <span class="close" >&times;</span>
                        <form class="edit-item">
                     <! -- select photo -->
                         <center class="photo">
                     <img id="blah" src="http://placehold.it/180" alt="your image
" />
                      <div class="update-photo">
                         <input type="file" name="file" id="file"
                         class="inputfile" onchange="readURL(this);" />
                      <label for="file">select photo</label>
              </div>
                        </center>
                         <div class="home">
                        <label for="w3review">Description</label>
                        <textarea id="w3review" name="w3review" rows="3" cols="55
">
                         </textarea>

                        <label for="w3review">Title</label>
                       <textarea id="w3review" name="w3review" rows="2" cols="55"
>
                       </textarea>

                      <label for="w3review">Phone</label>
                      <textarea id="w3review" name="w3review" rows="1.500" cols="
55">
                      </textarea>
                       <label for="w3review">Facebook</label>
                         <textarea id="w3review" name="w3review" rows="2" cols="5
5">
                          </textarea>
                             <label for="w3review">Price</label>
                          <input type="number" name="price" style="width: 430px;"
>
            <br>
                              <input class="submit" type="submit " value="save">
            </div>
    </form>
             </div>

             </div>
```

The button was placed like this and took an id = "myBtn" in order to reference the button in JavaScript

```
<button id="myBtn">Add</button>
```

Through this id we call a function when the button is pressed to show the dialog.

```
var modal = document.getElementById("myModal");
// Get the button that opens the modal//
var btn = document.getElementById("myBtn");
```

As you see in the JavaScript code, we linked the button to the dialog via the id that we gave to the button, which is id = "myBtn" and div, which is id = "myModal" which includes the content of the dialog.

When exiting this dialogue, we set close to help the user exit, so we used

```
<span class="close" >&times;</span>
```

And we activated it via JavaScript, where we used a function called "onclick". Get the <span>element that closes the model.

```
var span = document.getElementsByClassName("close")[0];
```

When the user clicks the button, open the dialog:

```
btn.onclick = function() {
  modal.style.display = "block";
}
```

When the user clicks on <span> (x), close the dialog:

```
span.onclick = function() {
  modal.style.display = "none";
}
```

When the user clicks anywhere outside of the dialog, close it

```

window.onclick = function(event) {
  if (event.target == modal) {
    modal.style.display = "none";
  }
}
```

As we see, we have defined the span responsible for closing by id = "close". Then, we used the onclick function of the button so that the dialog appears when the button is clicked. So, the line:

```
modal.style.display = "block";
```

Is responsible for showing the dialog. When exiting the dialog, span took NONE to close it:

```
modal.style.display = "none";
```

When the user clicks anywhere outside of the dialog, it also closes. So, span took NONE again.

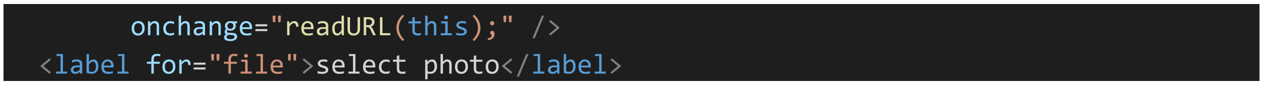
```
modal.style.display = "none";
```

We put an img element to act as a placeholder for the image that the user will choose. We gave it **alt = "your image"** and that shows an alternate sentence when a problem occurs in displaying this image and gave it the **id = "blah".**

```
 <img id="blah" src="" alt="" />
```

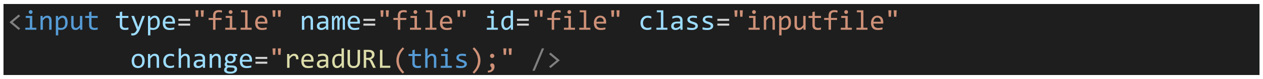
We used input type = "file" so that the user will be able to upload his photo, we also gave it a label “select photo” and linked them together through for = "file".

```
<input type="file" name="file" id="file" class="inputfile"
```



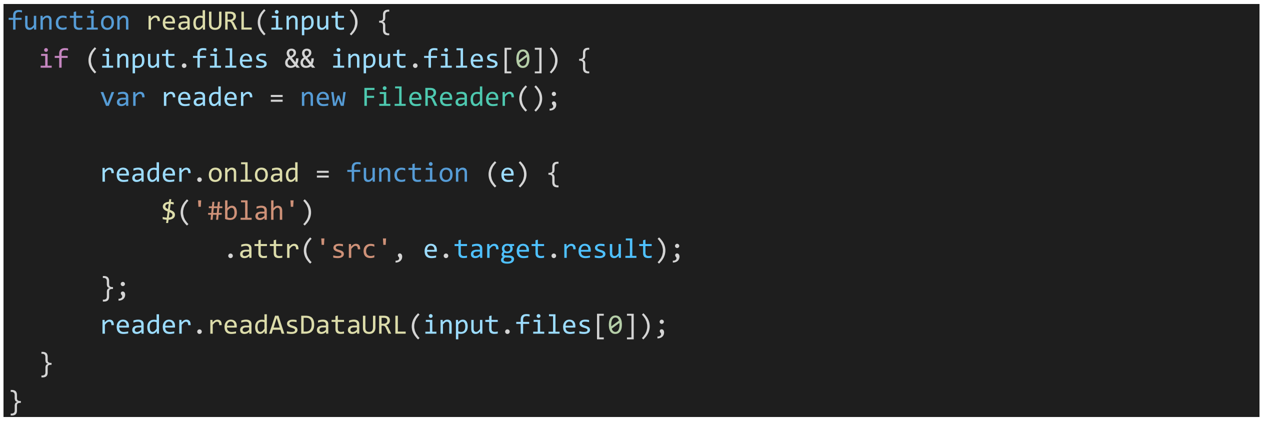
```
        onchange="readURL(this);" />
  <label for="file">select photo</label>
```

We used the function **“onchange =" readURL (this);"** to read the URL of the selected images:



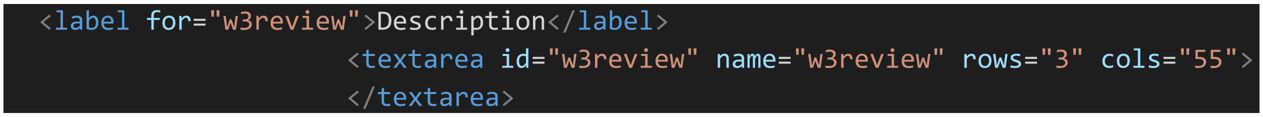
```
<input type="file" name="file" id="file" class="inputfile"
        onchange="readURL(this);" />
```

function onchange = "readURL (this);" will call the image from the browser, and when the user enters the browser and selects the image at this moment, the old picture will be deleted and the new image will be displayed.



```
function readURL(input) {
  if (input.files && input.files[0]) {
      var reader = new FileReader();

      reader.onload = function (e) {
          $('#blah')
              .attr('src', e.target.result);
      };
      reader.readAsDataURL(input.files[0]);
  }
}
```

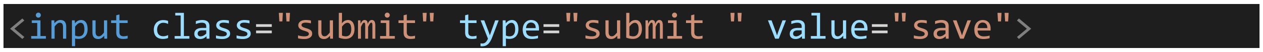
After that, we put a div that includes all the content, the first of them description and we put textarea to give the user a space to write in it



```
  <label for="w3review">Description</label>
                      <textarea id="w3review" name="w3review" rows="3" cols="55">
                      </textarea>
```

We decided this process will need title, phone, Facebook and About.

Finally, we put the save button at the end of the form to send the data to the database.



```
<input class="submit" type="submit " value="save">
```

Back end

For the back end of the project we decided to use Laravel for development, it is a well-known PHP framework. PHP is a very common server-side scripting language that made many famous products like Facebook and WordPress. It also has a big community that helps beginners. But development in pure PHP is not so easy so many frameworks were built on top of it. Laravel was our choice for a lot of reasons: -

1. It has elegant syntax; it is very popular because it allows web applications to be constructed quickly and make them easy to maintain afterwards.
2. The source code of **Laravel** is hosted on GitHub and is distributed under an open source license, so it is available for anyone to use.
3. It has a lot of features like Database migration, database Seeding, Scheduling, queuing and notifications etc.
4. It has command line interface called Artisan which is mapped to be sub-commands which simplifies the process of building and managing **Laravel-Based** apps.
5. It works on modular bases with lots of free build functions available.
6. This means that applications can be constructed quickly with no need for hours of work and lots of lines of codes.
7. Things like form validations and user authentications are built in **Laravel,** which means they do not need to be invented for each new task.
8. The code is designed to be simple readable and elegant make it easy for developer to understand what is happening if the code was created by someone else.
9. Testing is integrating into the **Laravel** framework too, so the resource intensive business of writing test procedures for each new task is reduced too.
10. Recent additions to the platform include **Laravel** Scout which allows full text searches to be performed.
11. The existence of Laravel Echo, which allows event broadcasting over the web.
12. **Laravel** Passport which is **Auth2** ready server which makes API authentication simpler.
13. **Laravel** has thriving and helpful developer community.
14. Laravel uses MVC architecture by default and that results in a cleaner code than pure PHP.

As mentioned before there is an API that serves the needed resources to the android and handling some processes for API can be different than handling it for the web browser so the coming sections will talk about both the web and API if there is a difference. There are many parts of the project that we can talk about so let us get started.

Authentication

Let us start with the authentication since it is the process that opens the door for more processes. When you sign up for the first time you will be required to fill in some information about yourself like first name and last name. The most important information is the email and password. The email must be unique. Two users cannot have the same email address if you entered an existing email address an error message will be shown to you that states you have to try a different email. The password does not have to be unique. There can be many users with the same password. However, they won't be stored as the same password. Passwords are not stored in their real state, they are hashed first then stored to the database. The same words can have different hashes according to PHP hashing algorithms. That is good because even if hackers access the database, they won't get the real password even if they tried to decrypt the hash.

```
$user = User::create([
    'email' => $request->email,
    'password' => bcrypt($request->password)
]);
```

After registration you will be signed in automatically, but you will be required to confirm your email address. The email confirmation process is done by sending an email to you after registration. This email is a temporary email that contains a link to activate your account. If you did not click the link soon the activation email will be useless, and you will need to request another one.

The log in is different from API than the web. Both authenticate the user but, they are not authenticating users in the same way.

- **Web**
  In the web after you log in your authentication state will be stored in a session. Each action that requires authentication after this will check to see if this session exists otherwise you will be redirected instantly to the login page again for a new session.

- **API**
  API log in on the other hand is not session driven. It uses OAuth2 standard protocol that is done through a Laravel package called **Passport.** Laravel Passport is native OAuth 2 server for Laravel apps. Laravel Passport package comes with a database migrations, routes, and middleware to ultimately create an authorization server that will return access tokens to allow access to server resources. It uses the League OAuth2 Server package as a dependency but provides a simple, easy-to-learn and easy-to-implement syntax. That means when

you log in through API you will receive a token. You will use this token in every action that requires users to be authenticated.

```
if (auth()->attempt($credentials)) {
    $token = auth()->user()->createToken('auth')->accessToken;
    return $this->apiResponse('login',['token' => $token]);
}
```

If you did not provide a token or provided a wrong token, you will not be authorized to do the action or access the resource.

Users who log in will be remembered until they log out or their tokens expired.

Security

Laravel framework comes with a lot of benefits. One of these benefits is security. Standard security risks that can face any backend developer can be handled with ease with the help of some great tools and api that the framework provides let us mention some of the security that we use in this project.

Middleware: -

Requests in Laravel are not handled directly by the application logic. A request has a life cycle. You can think of it as if a request goes through multiple layers before it reaches the right method in a controller. One of these layers is middleware. As stated in Laravel documentation

“Middleware provides a convenient mechanism for filtering HTTP requests entering your application. For example, Laravel includes a middleware that verifies the user of your application is authenticated. If the user is not authenticated, the middleware will redirect the user to the login screen. However, if the user is authenticated, the middleware will allow the request to proceed further into the application.

Additional middleware can be written to perform a variety of tasks besides authentication. A CORS middleware might be responsible for adding the proper headers to all responses leaving your application. A logging middleware might log all incoming requests to your application.

There is several middleware included in the Laravel framework, including middleware for authentication and CSRF protection.”

Here are some of the middleware we use for the security of our application: -

- **Auth**

Responsible for checking if users logged in before sending the request. If they are not logged in the middleware will redirect them to the log in page in case of web or show an unauthorized error in case of API.

- **Verify**
  Used to check if users have already verified their email addresses before sending the request. If not, the users will be redirected to the email verification page

- **Throttle**
  Used to rate limit access to routes. An example of this Is when you go to a website and enter a wrong password multiple times then you will be forced to wait for a short time before trying again. We use throttling when sending and processing emails as these actions can take long time.

- **Signed**
  Used to protect against URL manipulation by adding a hashed token to the URL. This middleware is used for actions like verifying email addresses.

- **CSRF**
  In a CSRF attack, an innocent end user is tricked by an attacker into submitting a web request that they did not intend. This may cause actions to be performed on the website that can include inadvertent client or server data leakage, change of session state, or manipulation of an end user's account. The CSRF protection is applied by default to all the web routes.

  Laravel Delegate syntax allows developers to use middleware with ease. An example of using middleware would be like this:

```
public function __construct()
{
    $this->middleware('auth:api')->only('resend');
    $this->middleware('signed')->only('verify');
    $this->middleware('throttle:6,1')->only('verify', 'resend');
}
```

XSS handling: -

Cross-site Scripting (XSS) is a client-side code injection attack. The attacker aims to execute malicious scripts in a web browser of the victim by including malicious code in a legitimate web page or web application. The actual attack occurs when the victim visits the web page or web application that executes the malicious code. The web page or web application becomes a vehicle to deliver the malicious script to the user’s browser.

Vulnerable vehicles that are commonly used for Cross-site Scripting attacks are forums, message boards, and web pages that allow comments.

By using blade template engine, we can prevent an XSS attack even if the user submitted a malicious java script code in a form and the code was stored in the database. By simply putting **{{ }}** around a variable we can escape it and the script will never be processed it will just appear as regular text.

```
<div>{{ $user->first_name }}</div>
```

SQL injection: -

Similar to XSS attacks SQL injection usually occurs when you ask a user for input, like their username/user id, and instead of a name/id, the user gives you an SQL statement that you will **unknowingly** run on your database.

The SQL statement can have very dangerous results varying from getting all the records of a table to deleting an entire table which is a disaster.

We used Laravel Query Builder to handle SQL injections with ease. That way
All the inputs to forms will be escaped before being stored in the database so again even if the user entered some malicious SQL statements none of them will be executed.

Authorization: -

Many of the routes and actions in the websites are protected by some kind of rules. These rules are necessary to protect the data of the users and deliver the desired experience. For example, any authenticated user can create an item, product or service and they must have the ability to edit or delete what they added but imagine if another user was able to delete or edit your own item. That would be a disaster that’s why user specific actions cannot be done by other users.
We check if the user for example requesting to edit an item is authorized to edit or not. That check can be done by checking if the user is the one who actually own the item or not by using the user id.

```
if(auth()->id()!=$item->user_id)
```

Another authorization rule is done when adding an event. Since only trusted users can add events, we added a rule to check if the user requesting to add an event is trusted or not.

```
if(!$request->user()->isTrusted())
```

This code is actually a part of a middleware that we already explained.

Encryption: -

For even more security important information like passwords, authentication tokens and even tokens used in URIs are all encrypted, and the hashed version is stored in the database. Even if someone could hack the database and get the hash he still can’t use it to log in because to log in the website take the password and hash it then compares the hashed password with the one stored in the database that will lead to different password.

The hashing algorithm used for that is the **CRYPT_BLOWFISH** algorithm which always results in 60-character string. The hashing is done using Laravel **bcrypt()** function.

Data base: -

For the database we chose MYSQL for the known performance and the ease of integration with PHP. During the work on project we created and changed many tables with different types of relation to maximize the performance as much as possible until we reached a number of 20 tables. We used a smart approach to design this number of tables. Instead of separating the design of the database from the PHP coding we used the code first approach. Thanks to Laravel migrations we can at any moment add a table, a column, an index or even a rule. And vice versa we can also drop them. We have complete control over the database while writing PHP code. There is no need to download a separate MYSQL GUI tool such as MYSQL Work Bench or phpMyAdmin to create or alter existing structure. Migrations act as version control at any moment any change, we do not want any more we can revert the database back to its previous state with a single command.

```
php artisan migrate:rollback
```

And similarly, for creating and running migrations each operation will require only ne command.

For creating a new migration file, we can simply write:

```
php artisan make:migration migration_name
```

And for running all migrations we use the command

```
php artisan migrate
```

by using this approach, we can focus on what data base schema we need and create it on the fly instead of wasting a lot of time thinking about a design that might be wrong in the future. The complete migration file can look something like this.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateInfoTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */     public function up()
    {
        Schema::create('info', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->unsignedBigInteger('user_id');
            $table->string('first_name');
            $table->string('last_name');
            $table->string('grade');
            $table->string('avatar');
            $table->enum('gender',['male','female']);
            $table->timestamps();
            $table->foreign('user_id')->references('id')->on('users');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */     public function down()
    {
        Schema::dropIfExists('info');
    }
}
```

ORM: -

Object-relational-mapping is the idea of being able to write SQL queries, as well as much more complicated ones, using the object-oriented paradigm of your preferred programming language. We used Eloquent ORM that Laravel supports by default to handle the database.

By using any ORM we abstract away the database system so that switching from MySQL to PostgreSQL, or whatever flavor we prefer, is easy. And thanks to the MVC architecture each entity in the database can be seen as a model. However Eloquent does not just make writing queries easy it also come with benefits that we used like accessors and mutators that allow us to intercept and change the data before being sent to the database or being used in a query. This can be used for example to cast a value or format a date before being stored in the database or do a specific operation when a field is null. Eloquent does that automatically for each query if you followed the rules of writing an accessor or mutator function in the model.

Another really great benefit of using eloquent models is the relationship methods using original SQL making a query using a relation can be complicated specially if you are joining multiple tables not just two but using eloquent it is so easy to make joins and get data from many different tables with a small amount of code as an example here is a one to many relation between a user entity and an item entity.

In User Model we write

```
public function items()
{
    return $this->hasMany(Item::class,'user_id');
}
```

Here we tell the user model that it can have many Item entities and the foreign key will be **user_id.**

On the other hand, in the Item Model we write this code

```
public function user()
{
    return $this->belongsTo(User::class, 'user_id');
}
```

Here we tell the Item model that it will always belong to a user entity and of course the foreign key will be **user_id** again.

Now that we defined the relation, we just need to use it for example to get all the items of a single user with the id of 1 we can just write

```
$user = User::find(1);
$items = user->items;
```

Seeding and faker library: -

For working with the database, we need some actual data. Instead of manually inserting data or looping for a specific amount of iterations we used Laravel seeding mechanism.

To seed a specific table, it is better to first create something called a **Factory.** As the name implies a factory is responsible for creating a single entity however, we need each entity to have some different or unique field values than other entities for better testing and debugging. That is why we used the **Faker library**. It is a great library for creating dummy data of different types ranging from names to sentences and date. A factory class using Faker library can look something like this.

```
<?php

/** @var \Illuminate\Database\Eloquent\Factory $factory */

use App\Item;
use Faker\Generator as Faker;

$factory->define(Item::class, function (Faker $faker) {
    return [
        'user_id’ =>$faker->numberBetween(1,50),
        'title' =>$faker->title(),
        'description' =>$faker->sentence(),
        'price' =>$faker->randomFloat(2,0,99999),
        'phone' =>$faker->randomNumber(),
        'facebook '=>$faker->url(),
        'image' =>$faker->imageUrl(),
        'created_at '=>$faker->dateTime,
        'updated_at '=>$faker->dateTime
    ];
});
```

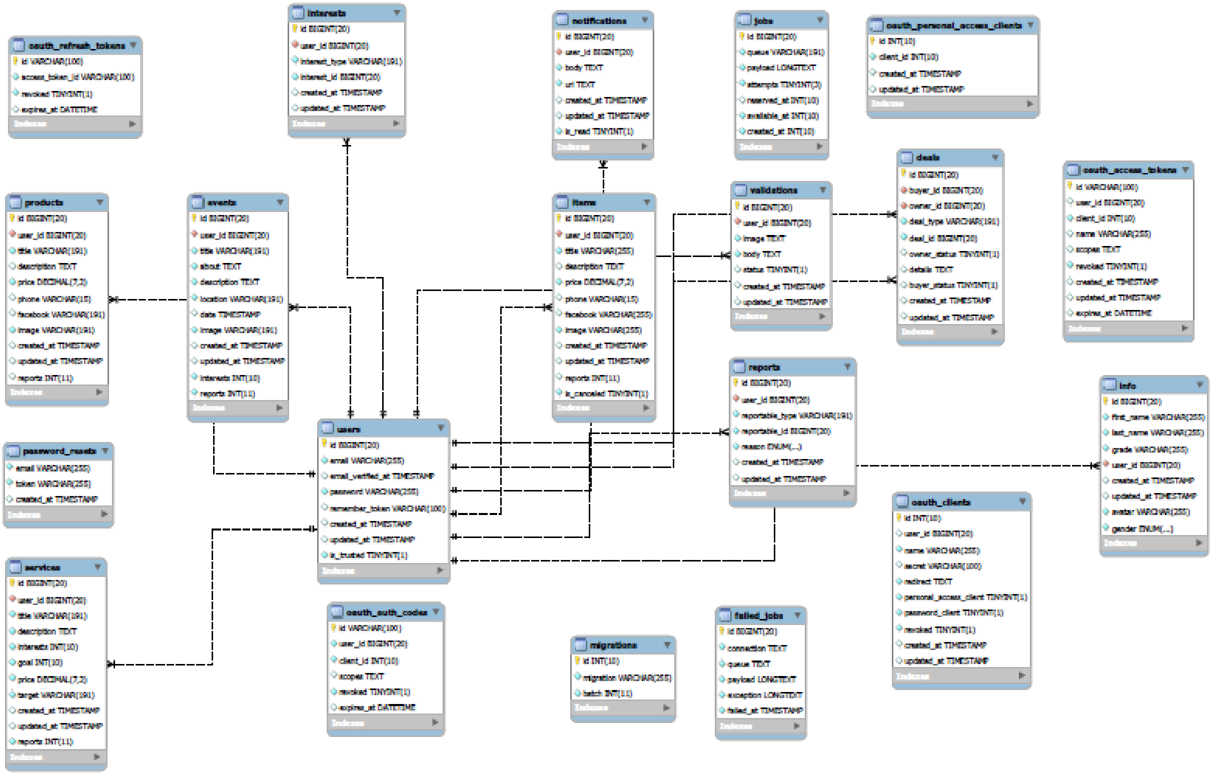
After that we create the seeder class to create any number of entities at once and we can even override some attributes if we for example want all the users to have the same password so we can test with many different users.

A seeding class can look like this.

```
<?php

use Illuminate\Database\Seeder;

class ItemsTableSeeder extends Seeder
{
    /**
```

```
     * Run the database seeds.
     *
     * @return void
     */
 public function run()
    {
        factory(\App\Item::class,100)->create([
            'image'=>'1582808660.jpg'
        ]);
    }
}
```

ERD



Results

At time of writing this documentation Hand in hand is graduation project which means it is still not tested in a real world so there is no real data or analysis made around the impact or the costs. The costs are just the costs for the server to host the application or maintenance for the developing tools and machines. However, we believe that if used right Hand in hand will positively impact the community in many ways. As it can impact not only the life of students but also the country for example: -

- **social impact: -**

  (Hand in Hand) encourages the culture of sharing, donation and caring. This is mainly resembled in allowing university students to get academic materials and university supporting stuff from each other without embarrassment.
  It works in the niche market for university students allowing talented students to get noticed easily. This can encourage students to make small student activities or clubs for crafting and selling products.

- **Financial impact: -**

  Hand in Hand enable Students to earn or save money depends on the way they use it. Students can earn money by selling simple products. Thanks to the dedication of our solution to the students it will be easier for talented students to gain more notice, reputation and of course even more money. Students can save up to 500LE or more as stated in the survey by using the donated materials. The financial impact can be (National) by saving part of the money needed for printing books or providing new materials each year as the need for new materials can drop down.

- **Environmental impact: -**

  There is a developed global interest in recycling and Egypt is following an ambitious plan towards the recycling. Our Hand in Hand provides a nearby recycling option for the highly waste providers via donating these stuffs to others who need them. If we cannot recycle these materials, then at least we can help find new use for them.

References

Find all our repositories and the references of our code and the libraries that we used on GitHub.

ANDROID:

https://github.com/Mohamed-Emad126/HandinHand

Back-End:

https://github.com/mohamed-fathy3010/hand-in-hand

Front-End:

https://github.com/aya-cooder/HandInHand-Website