**Faculty of Computers and Artificial intelligence- Cairo University (credit hours system)**



Cairo University, Faculty of Computers and Artificial intelligence

Graduation Project 2022

Midyear Short Documentation

**Kher Wallet**

|  |  |
| --- | --- |
| **Implemented by** | |
| **ID** | **Name** |
| 20186031 | Nada Mohamed |
| 20186007 | Ayat Hany |
| 20186008 | Sarah Khaled |
| 20186043 | Mark Rofaeel |
| 20186049 | Yasmine Shehab |

**Supervised by:**

Dr. Khaled Wassif

TA. Ahmed Ali

Table of Contents

[1. Abstract: 3](#_Toc97763406)

[2. Background: 3](#_Toc97763407)

[3. Problem definition: 3](#_Toc97763408)

[4. Project Specifications: 4](#_Toc97763409)

[4.1 System architecture: 4](#_Toc97763410)

[4.2 Stakeholders: 4](#_Toc97763411)

[4.3 Functional Requirements: 5](#_Toc97763412)

[4.4 Non-functional Requirements: 7](#_Toc97763413)

[4.5 Constraints: 7](#_Toc97763414)

[4.6 Use-case Diagram: 8](#_Toc97763415)

[4.7 Use-case Tables: 9](#_Toc97763416)

[4.8 Class Diagram: 11](#_Toc97763417)

[4.9 Sequence Diagram: 12](#_Toc97763418)

[4.10 Entity Relationship Diagram (ERD): 15](#_Toc97763419)

[5. Work Plan: 16](#_Toc97763420)

[5.1 Table: 16](#_Toc97763421)

[5.2 Gantt Chart: 16](#_Toc97763422)

# 1. Abstract:

Nowadays, there are a lot of organizations that accept donations from different people, either by donating money, clothes, devices, books or even by volunteering in many activities. It takes time and effort for people to reach out to such organizations, and people do not even know when they need such help. So, our motivation is to solve this problem by implementing an application which gather different kinds of organizations, that help people to donate in an easier way and also notify when there is an event around the corner or if there are specific cases that need help. This application will be implemented using Flutter to be used on Android and IOS platforms.

# 2. Background:

* **Our motivational beneficiary:** The main category people who will have the most benefit out of our application is the general public. The organizations who will add their places to our system will benefit from the different donations. As well as new people will reach these organizations.
* **Main Techniques:** Android and iOS platforms as the client-side platform using Flutter.

# 3. Problem definition:

Our problem is that people spend so much time and effort to reach different organizations as charities, orphanages or food banks that has led to the shortage of the overall amount of donations. Other problem is that those organizations do not have the capability to reach large amount of people to help in notifying them with events and specific needed cases. So, let’s imagine that there is an application that gather all of these aspects together and that is what our application is going to do.

# 4. Project Specifications:

## Graphical user interface, text, application Description automatically generated System architecture:

Figure

\*In *Figure 1*, it shows the different system components and the relations between them. A user, through UI layer, calls a certain function, that is then passed to the logic and domain layer. Data may be retrieved from database or not based on the called function.

## Stakeholders:

* + Internal Stakeholders:
    - Developers.
    - Designers.
    - Testers.
    - Project managers.
  + External Stakeholders:
    - Organizations (places’ owners).
    - Donators.

## Functional Requirements:

* Donator’s functionalities:
  + Sign up:
    - Donator should be able to **create an account** on our application with their full name, email, password, address, and phone number or by using Facebook account or Google account.
  + Sign in:
    - Donator should be able to **sign into** the application by their email and password.
  + Sign out:
    - Signed-in donator should be able to **sign out** of their current account at any given time.
  + Update donator’s profile:
    - Signed-in donator should be able to **update** any information in profile.
  + View notifications:
    - Donator should be able to **view notifications** whenever a new place is added, or a new donation category is available, or when there is an event where help is needed, or when specific needed cases are posted by the organizations.
  + View events:
    - Donator should be able to **view** the upcoming events where help is needed.
  + Expand event description:
    - Donator should be able to **view more** about a specific event with full description by expanding it.
  + Donate:
    - Donator should be able to **donate** by using credit card or by requesting a representative to donate money, clothes, devices, or books.
  + Add to cart:
    - Donator should be able to **add donations** to cart from same or different places.
  + Remove from cart:
    - Donator should be able to **remove donations** from cart.
  + View cart:
    - Donator should be able to **view** different donations in cart.
  + Search:
    - Donator should be able to **search** for different places and events or categories.
  + Write feedback:
    - Donator should be able to **write the feedback** of his own**.**
  + Request help:
    - Donator should be provided by a ‘**help**’ section that offers detailed guidelines for both users and place owners. It will also include the most frequently asked questions with provided answers.
  + Contact us:
    - Donator should be able to **contact us** to solve any problem.
  + Display places categories:
    - Donator should be able to **display** which place category they want to explore from places categories.
  + Select specific place:
    - Donator should be able to **select** specific place from places categories and can view detailed information about it.
  + Display donation categories:
    - Donator should be able to **display** which donation type they want to explore from donation categories.
  + Select specific donation:
    - Donator should be able to **select** specific donation type and then view different places accepting this kind of donation.
* Organization’s functionalities:
  + Sign in:
    - Organization should be able to **sign into** the application by their email and password (provided by administrators).
  + Sign out:
    - Signed-in organization should be able to **sign out** of their current account at any given time.
  + Update organization’s profile:
    - Signed-in organization should be able to **update** any information in profile.
  + View events:
    - Organization should be able to **view** the upcoming events where help is needed.
  + Expand event description:
    - Organization should be able to **view more** about a specific event with full description by expanding it.
  + Write event:
    - Organization should be able to **write** any upcoming event where help is needed which is then send as a notification to donators.
  + Update event:
    - Organization should be able to **update** existing event which is then send as a notification to donators.
  + Remove event:
    - Organization should be able to **remove** existing event.

## Non-functional Requirements:

* + **Portability:**
    - The software will be available for two platforms as it will run on both Android and iOS devices.
  + **Usability:**
    - Users can easily determine what a feature is and what it can do, also, users can easily navigate the application’s interface.
  + **Availability:**
    - The application will be available to use 24/7.
  + **Response time:**
    - The application should load at not more than 5 seconds.

## Constraints:

* + - The password should contain at least 8 characters including at least one uppercase letter, at least one lowercase letter and a digit.
    - The email must be unique.
    - The phone number of the user should not be less than 11 digit and must be unique.

## Use-case Diagram:

Diagram

Description automatically generated

Figure

\*In Figure 2, it shows the different functionalities that a donator or an organization can execute. Some functionalities are extended or included under other functionalities.

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Donate | |
| Actors: | Donator | |
| Pre-conditions: | The donator needs to have a successful account entry on our application. | |
| Post-conditions: | The donator can now donate by money, clothes, devices or books to any organization and then the donation is added to the cart. | |
| Flow of events: | **User Action** | **Application Action** |
| 1- Donator opens the application and click on places category then charities. |  |
|  | 2- Application shows different charities. |
| 3- Donator may select Resala Charity Organization. |  |
|  | 4- Application lets the user choose what kind of donation he needs. |
| 5- Donator may select Money Donation. |  |
|  | 6- Application shows different types of donations either credit/debit card or request representation. |
| 7- Donator may select Credit Card. |  |
|  | 8- Application adds the type of donation to the database and it can be viewed from the cart section. |
| Exceptions: | **User Action** | **Application Action** |
| 1- Donator may have entered an amount of money more than the actual card has. |  |
|  | 2- Application shows a failed donation transaction. |
| Extend: | Donate by money, donate by clothes and donate by devices. | |

## Use-case Tables:

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Write event | |
| Actors: | Organization | |
| Pre-conditions: | The organization needs to have a successful account entry on our application. | |
| Post-conditions: | The organization can write event easily. | |
| Flow of events: | **User Action** | **Application Action** |
| 1- Organization opens the application and signs in. |  |
|  | 2- Application verify the data from the database then show the home screen. |
| 3- Organization write event in the section provided. |  |
|  | 4- Application store this event to the database, that then is shown to the donator’s home screen. |
| Exceptions: | **User Action** | **Application Action** |
| 1- User may have lost internet connection while write an event. |  |
|  | 2- Application will not be able to store such event in the database. |

## Class Diagram:

Diagram, box and whisker chart

Description automatically generated

Figure

\*In Figure 3, it shows the different classes and the relations between them. Here are the main classes (donator and organization) and the classes connected to them.

## Sequence Diagram:

* + Sign up scenario:
    - Donator tries to sign up into the application. There are three wrong scenarios while registering: if the donator used a non-unique email, password less than 8 characters or phone number less than 11 characters, other than that the donator will be registered successfully.

Diagram, box and whisker chart

Description automatically generated

Figure

* + Sign in, view notifications, and view events scenarios:
    - Donator signs into the application, then receives a notification. Donator will view notifications that are retrieved from the database. Donator will navigate to home where events can be viewed and expanded.

Diagram

Description automatically generated

Figure

* + Sign in, list category and donation (clothes) scenarios:
    - Donator signs into the application, then navigate to category list where donation categories are chosen. Then a list of different donation types is displayed to the donator. Clothes are, then, chosen by the donator and a list of places that accept this kind of donations are viewed. Donator chooses a place, and a message is sent saying “A representative will contact you” and it is then added to the database that a specific donator has requested a representative to donate clothes.

Diagram

Description automatically generated with low confidence

Figure

## Entity Relationship Diagram (ERD):

Diagram

Description automatically generated

Figure

\*In Figure 7, it shows the different tables in the database the relations between them. There are Many-To-Many, Many-To-One, One-To-Many and One-To-One relationships.

# 5. Work Plan:

## Table:

|  |  |  |
| --- | --- | --- |
| Task | Task Title | Task status |
| 1 | Idea brainstorming. | Stared 22/10/2021 |
| 2 | Started the documentation. | Started 19/12/2021 |
| 3 | 1st requirement trial. | Finished 23/12/2021 |
| 4 | Changes in the documentation. | Started 11/2/2022 |
| 5 | Finished the diagrams and started doing the logo. | Started 14/2/2022 |
| 6 | UI prototyping screens. | Started 15/2/2022 |
| 7 | UI prototyping screens + logo. | Started 20/2/2022 |
| 8 | Presentation making. | Started 22/2/2022 Finished 24/2/2022 |
| 9 | Flutter implementation. | Started 8/3/2022 |
| 10 | First release. | Will be 8/4/2022 |
| 11 | Maintenance. | Will be 20/4/2022 |
| 12 | Second release. | Will be 15/6/2022 |
| 13 | Final touches in the project. | Will be 30/6/2022 |

## Gantt Chart:

Figure