



NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

**SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION**

BSc THESIS

CIR – A Citizen Incident Reporting tool

Athanasios A. Filippidis

Supervisors:

Alexios Delis, Professor NKUA

Maria Roussou, Assistant Professor NKUA

ATHENS

MARCH 2019



ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

**ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

CIR – A Citizen Incident Reporting tool

Αθανάσιος Α. Φιλιππίδης

**Επιβλέποντες: Αλέξιος Δελής, Καθηγητής ΕΚΠΑ
Μαρία Ρούσσου, Επίκουρος Καθηγήτρια ΕΚΠΑ**

ΑΘΗΝΑ

ΜΑΡΤΙΟΣ 2019

BSc THESIS

CIR – A Citizen Incident Reporting tool

Athanasios A. Filippidis

S.N.: 1115201400215

SUPERVISORS: **Alexios Delis**, Professor NKUA
Maria Roussou, Assistant Professor NKUA

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

CIR – A Citizen Incident Reporting tool

Αθανάσιος Α. Φιλιππίδης

A.M.: 1115201400215

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Αλέξιος Δελής**, Καθηγητής ΕΚΠΑ
Μαρία Ρούσσου, Επίκουρος Καθηγήτρια ΕΚΠΑ

ABSTRACT

To date, the main and possibly the only way to report a public incident (e.g., a fallen tree blocking a street) to the authorities is only by phone, email which is time-consuming and rather inefficient. Moreover, people of every age, day by day are using more and more their smartphones. A solution to the aforementioned problem could be found by taking advantage of that new habit people tend to have. The solution proposed in this thesis, is an Android application through which everyone will be able to capture an incident using their phone by getting a picture of the incident, write a short description and send it as simply and quickly as possible. The application attaches the exact coordinates of the user when the picture was taken, gives the user the option of choosing the level of the emergency and adds those to the information available regarding the problem. On the other end there is a server application that successfully receives the image and the information and presents those on a user friendly interface that can be used by the authorities in order to short, group and resolve the reported incidents.

Having the aforementioned in mind, the following thesis has been written in order to analyze and describe the procedure that was followed during the implementation of that Android application. The implemented application's name is CIR which is the acronym for Citizen Incident Reporting. The application can be found in my GitHub repository [1]. The language used for the development of the application was Java with the addition of some open source libraries like Picasso and Retrofit in an Android Studio environment. In order to present a fully functional system a server to receive those incident reports was developed too. For the server Python was used, and more specifically the Django REST Framework.

As this application is intended for use by a broad public, a user-centered design (UCD) approach was followed to ensure the usability and optimal user experience of the interface and user interaction mechanisms. This UCD approach included identifying the target users, designing user personas, conducting a hierarchical task analysis, devising a use case scenario, and finally creating low fidelity prototypes and a usability (field) test in which users were observed using the mobile application.

SUBJECT AREA: Mobile Application development, REST Web application development, Human Computer Interaction

KEYWORDS: django rest framework, retrofit, gson, android, picasso, low fidelity wireframes, personas, user-centered design

ΠΕΡΙΛΗΨΗ

Αυτή τη στιγμή, στην Ελλάδα, ο κύριος και πιθανότατα μόνος τρόπος να αναφερθεί ένα δημόσιο συμβάν (π.χ. ένα πεσμένο δέντρο που εμποδίζει σε μία οδό) στις αρχές είναι μόνο μέσω τηλεφωνικής κλήσης, email κάτι το οποίο είναι χρονοβόρο και μη αποτελεσματικό. Επιπρόσθετα, άτομα κάθε ηλικίας, τείνουν να χρησιμοποιούν καθημερινά όλο και περισσότερο τα έξυπνα κινητά τους. Μια λύση στο προαναφερόμενο πρόβλημα θα μπορούσε να βρεθεί εκμεταλλευόμενοι αυτή τη νέα συνήθεια που τείνουν να έχουν οι άνθρωποι. Η λύση που προτείνεται μέσω αυτής της πτυχιακής είναι μια εφαρμογή Android μέσω της οποίας ο καθένας θα μπορεί να τραβήξει μια φωτογραφία του συμβάντος, να γράψει μια σύντομη περιγραφή αυτού και να το αναφέρει όσο πιο απλά και σύντομα γίνεται. Η εφαρμογή επισυνάπτει τις ακριβείς συντεταγμένες του χρήστη κατά την στιγμή της αναφοράς, δίνει την επιλογή του επιπέδου κρισιμότητας του συμβάντος στον χρήστη και προσθέτει αυτά στις πληροφορίες που είναι διαθέσιμες για το συμβάν. Στην άλλη μεριά, βρίσκεται μια εφαρμογή εξυπηρετητή (server) η οποία πετυχημένα λαμβάνει την εικόνα και τις πληροφορίες του συμβάντος τα οποία και παρουσιάζονται σε ένα φιλικό προς τον χρήστη περιβάλλον το οποίο μπορεί να χρησιμοποιηθεί από τις αρχές με σκοπό να ταξινομήσουν, ομαδοποιήσουν και επιλύσουν τα αναφερθέντα συμβάντα.

Έχοντας τα παραπάνω κατά νου, η πτυχιακή αυτή γράφτηκε με σκοπό να αναλυθούν και να περιγραφούν οι διαδικασίες που ακολουθήθηκαν κατά την διάρκεια της υλοποίησης αυτής της Android εφαρμογής. Το όνομα της υλοποιημένης εφαρμογής είναι CIR το οποίο είναι το ακρωνύμιο του Citizen Incident Reporting tool. Την εφαρμογή μπορεί να την βρει ο αναγνώστης, εάν επιθυμεί, στον προσωπικό μου λογαριασμό στο GitHub [1]. Η γλώσσα που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής ήταν η Java με μερικές προσθήκες από βιβλιοθήκες ανοιχτού κώδικα όπως η Picasso και η Retrofit. Το περιβάλλον στο οποίο αναπτύχθηκε ήταν το Android Studio. Προκειμένου να παρουσιάσουμε ένα πλήρως λειτουργικό σύστημα δημιουργήσαμε και έναν εξυπηρετητή (server) ο οποίος παραλαμβάνει τις αναφορές των συμβάντων. Για τον εξυπηρετητή χρησιμοποιήθηκε Python και πιο συγκεκριμένα το Django REST Framework.

Εφόσον αυτή η εφαρμογή έχει ως σκοπό να χρησιμοποιηθεί από ένα ευρύ κοινό, η σχεδιαστική προσέγγιση η οποία ακολουθήθηκε ήταν αυτή του σχεδιασμού με επίκεντρο τον χρήστη προκειμένου να εξασφαλιστεί η χρηστικότητα και η βέλτιστη εμπειρία του χρήστη σχετικά με την διεπαφή και τους μηχανισμούς αλληλεπίδρασης. Ο σχεδιασμός με επίκεντρο τον χρήστη συμπεριλαμβάνει μελέτη των ομάδων χρηστών που θα αλληλεπιδρούν με την εφαρμογή και την υλοποίηση προσωπικών χρηστών, ανάλυσης εργασιών, σεναρίων χρήσης, πρωτοτύπων χαμηλής πιστότητας και τέλος ένα τεστ αξιολόγησης κατά το οποίο παρακολουθήθηκαν χρήστες την ώρα που χρησιμοποιούσαν την εφαρμογή.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ανάπτυξη εφαρμογής για κινητό τηλέφωνο, Ανάπτυξη εφαρμογής διαδικτύου τύπου REST, Επικοινωνία ανθρώπου-μηχανής

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: django rest framework, retrofit, gson, android, picasso, πρωτότυπα χαμηλής πιστότητας, προσωπεία, σχεδιασμός με επίκεντρο τον χρήστη

*Η παρούσα πτυχιακή είναι αφιερωμένη στην οικογένεια μου,
στους γονείς μου Τάσο και Έφη, τα αδέρφια μου Άννα και Χρήστο,
και στον πυλώνα που ήταν εκεί κάθε φορά που ήμουν έτοιμος να τα παρατήσω,
την Ελένη.*

ACKNOWLEDGMENTS

I would like to thank my supervisors, Mr. Alexis Delis and Ms. Maria Roussou for their guidance, assistance and especially patience in the preparation of this thesis.

CONTENTS

1. INTRODUCTION	13
2. BACKGROUND WORK	14
2.1 Systems for Civil Engagement	14
2.2 Previous and Similar applications	14
3. APPLICATION DESIGN	15
3.1 User-Centered Design (UCD).....	15
3.2 User requirements analysis.....	15
3.2.1 User groups.....	15
3.2.2 Task analysis	16
3.2.3 Usage scenario	17
3.3 Prototyping.....	17
3.3.1 Wireframes introduction	18
3.3.2 Low fidelity wireframes.....	18
3.4 Usability Testing	18
3.4.1 Introduction.....	18
3.4.2 Testing conditions	18
3.4.3 Testing results.....	18
3.4.4 Changes made based on the results	19
3.5 Summary.....	19
4. IMPLEMENTATION	20
4.1 Introduction	20
4.2 Server-side	21
4.2.1 REST	21
4.2.2 Django – Django REST Framework.....	22
4.3 Android application	24
4.3.1 Android operating system	24
4.3.2 Android development	25
4.3.3 Picasso.....	26
4.3.4 Retrofit.....	26
4.3.5 Google Mobile Services (GMS)	26
4.3.6 Camera 2 API.....	26
4.4 Presentation of the application	27
4.4.1 Android client-side.....	27
4.4.2 Django server-side	38
4.5 Summary.....	42
5. CONCLUSION	43

5.1 Summary..... 43

5.2 Future Development 43

6. ANNEX.....44

TABLE OF TERMINOLOGY58

ABBREVIATIONS - ACRONYMS59

REFERENCES60

LIST OF IMAGES

Illustration 1: John - first persona.....	15
Illustration 16: The intro view of the application; a loading screen to provide some time for the resources needed to load.	27
Illustration 17: Log in view; the user can type his or her credentials in order to log in to his or her account, to sign up or to retrieve the account's password. If the user has already logged in before he or she will be redirected to the home view without having to log in again.	28
Illustration 18: Sign up view; the user has to type a valid email, a password (while the strength bar below will show its strength dynamically) and a valid phone number for verification reasons.....	29
Illustration 19: Home - Capture incident view; User can turn on and off the flash, can tap on the menu button to navigate to the application's menu or can tap the camera button to capture an incident.	30
Illustration 20: Review image view; in this view the user can review the image captured and can decide between canceling this capture by tapping the "X" button, saving the image for personal use by tapping the save button (in which case the pop up that can be seen is displayed to inform him or her about the success of his or her action), or tap the check button to proceed with the creation of the report.	31
Illustration 21: Create incident report view; In this view the user can tap on the text area to type a short description of the incident, can move the slide bar to determine the emergency level of the incident reported, can tap the back button to return to the review image view or can finally tap the send button to send the review to the server.	32
Illustration 22: Successful incident report view; this is the view that the user is redirected after sending the report, a message appears also to inform him or her about the success of the report.	33
Illustration 23: Menu view; here the user can either tap the back button to go back to the home view or tap one of the four choices in order to be redirected in their respective views.	34
Illustration 24: My account view; in this view the user can see his or her credentials and change them by tapping the "save changes" button or go back to the menu by tapping the "back" button.....	35
Illustration 25: My reports view; in this view the user can see his or her previous reports as well as to go back b tapping the "back" button.....	36
Illustration 26: Alternative my reports view; this is how the "my reports" view looks like when there have been no reports by this user.....	37
Illustration 27: My settings view; in this view the user can set his or her personal settings (if he or she wants to receive notifications from the application or to use a mobile data efficient mode of the app to avoid spending too much mobile data).	38
Illustration 28: Administrator Home Page. We have implemented the two main models required, user and incident model. We can always add or edit a new instance of those from the administrator page.....	39
Illustration 29: Administrator Log In page	39

Illustration 30: Incidents List Page. A search field is implemented, so the administrator can search for keywords in reports, the incidents can be sorted by each of their fields simply by clicking on the name of a field and there is a date filter implemented to help tracking incidents by the time they occurred.....	40
Illustration 31: Incident's details page. There the administrator can inspect all the details of an incident as well as click on the image of it to gain more information about it. Moreover the administrator has the option of editing some of the details in case there has been a mistake.	40
Illustration 32: Users List Page. There the administrator can view all the signed up users as well as their id's.	41
Illustration 33: User's details page. There the administrator can see all the details of a user as well make any necessary changes in their accounts.	41
Illustration 2: Intro loading page wireframe.....	44
Illustration 3: Log in wireframe.....	45
Illustration 4: Sign up wireframe	46
Illustration 5: Forgot your password? wireframe	47
Illustration 6: Home wireframe during first user entrance showing a short tutorial, first part	48
Illustration 7: Home wireframe during first user entrance showing a short tutorial, second part	49
Illustration 8: Home wireframe during first user entrance showing a short tutorial, third part	50
Illustration 9: Home wireframe every time after the first time the user enters the application	51
Illustration 10: Incident image preview wireframe	52
Illustration 11: Create incident report wireframe	53
Illustration 12: Menu wireframe; the user can tap either to each of the four available views in order to be redirected to those or back to go back to the home view.....	54
Illustration 13: My account wireframe	55
Illustration 14: My reports wireframe.....	56
Illustration 15: Settings wireframe.....	57

1. INTRODUCTION

Nowdays, we live in a modern society in which Internet and smartphones have become an integral part of our lives. People of different ages use them for different purposes. Some kinds of uses like music applications or social media have thrived but there are other aspects of everyday life that have not yet become that associated to those new technologies.

Among these aspects is the active participation of citizens in their local community, through acts such as voting, volunteering, helping the police, etc. An important aspect of civil society is the ability for citizens to communicate directly, quickly, and reliably with their local government or municipality. Bureaucracy has always been a burden in such kind of communication and now technology is offering a viable solution to problems of that kind. Automated systems can provide easy and simple ways to report and interact with the responsible authorities for each kind of issue that could occur in a neighborhood. People could avoid spending their time waiting in phone lines or googling for the right authority to call while their daily life is more stressful and time-pressing than ever. The easiest way to resolve this obstacle without adding any extra expenses to peoples' life (either in time or money) is the use of a device that almost everyone owns already. A smartphone.

A smartphone application that follows a minimal design and offers to the users a very simple way to report an incident is exactly what could solve that problem. An application of that kind is something that is really missing from the market. The only two similar applications that were found are the "Eye of Sharjah" [2] developed by Neologix and the "1st Incident Reporting" [3] developed by emAPPetizer. Both of those applications, though, have flaws and they are not sufficient to cover that need of the market, so our application aims at avoiding these flaws while in the same time providing to the users a user-friendly interface that has all the functionalities they need.

The two main smartphone operating systems are Android and iOS. The market share of each is approximately 88% for the Android and 11.5% for the iOS [13]. Based on that statistics we based the decision to implement an application for the most commonly used operating system, Android.

2. BACKGROUND WORK

2.1 Systems for Civil Engagement

The information and communication technologies that are available to almost everyone nowadays can and should be used in order to create new ways of interaction between the citizens and the government. Those ways include many different types of interaction. Probably the most famous of them is e-voting, a modern way of voting for a wide variety of elections, from local to nationwide, which incorporates Internet technologies to resolve mobility issues for the voters while in the same time decreasing the probability of corruption. A citizen reporting tool which is taking advantage of cutting edge technologies could be easily considered to be associated with that ideology too. Information systems can be used to support various forms of civic engagement, such as community policing, and urban infrastructure maintenance. In these systems, citizens can be utilized to act as data sensors to help populate databases regarding local issues. Then, the collected data can be visualized and shared within local communities and the responsible authorities. In this way the local communities and the responsible authorities can have a simple but also trusted way to categorize those issues and finally to resolve them.

2.2 Previous and Similar applications

After conducting some research it was found that no other applications exist, web or mobile, that could serve this use in the Greek market. Worldwide there have been some other mobile applications, the most relevant one being an Incident Management Mobile App developed by Neologix for the city of Sharjah in the United Arab Emirates, named “Eye of Sharjah” [2] which, though, does not seem to be available on Play Store [11]. One more application worth mentioning is the “1st Incident Reporting” [3] developed by emAPPetizer which is not using images as part of the incident report and it is designed to be used by companies, not the government. According to the Google Play store [11], the most common public reporting application, “Report of Inconvenience,” is rated at 1 star, with 1,316 reviews out of a total of 3,577 reviews [14]. At this point we would like to mention that the lack of supply of this kind of products in combination with the levels of demand that they would have if the average citizen realized how much easier they would make his or her everyday life could be quite significant. As it can be noticed those applications are far from what the average citizen would need. They are either not designed for the specific purpose that we mentioned or they have a non-user-friendly design which leads to the low ratings that they have received.

3. APPLICATION DESIGN

3.1 User-Centered Design (UCD)

User-Centered Design is a design process focused on the needs of the user and used in human-computer interaction analyses [10]. The human-computer interaction is a field of science that incorporates knowledge from various other fields such as computer science, behavioral sciences and design sciences. Its main focus is on the optimal design of interfaces between humans and applications (computer, mobile and others). User-Centered Design incorporates different kind of methods such as interviews, surveys, as well as brainstorming in order to fully understand the needs of the user regarding one application. A successful UCD has as a result a design plan for one application which, when finally implemented, leads to a user-friendly application. That means an application that will avoid creating any confusion to the user. For example, the decision to use a button in the shape of a camera to capture an image is a user-friendly design. The use of one other random shape, a rectangle for example, would not make clear to the user the purpose of this button and would lead to confusion. UCD has as a target to avoid confusions like this. UCD, especially on applications that are used under time-pressure, can prove crucial. For example, a mobile camera application in which the “capture” button is not obvious can make the user miss the shot of a bird, which is a total failure of the application. Even in applications that will not be used under time-pressure UCD can have a significant impact. Nowadays the users are accustomed to fast applications, with almost zero waiting for the execution of procedures. A user who is not able to find quickly the function he or she is looking for at each phase of an application will, extremely easily, leave the application and uninstall it or decide to not use it again. In order to be clear, because “quickly” can be a vague notion, this can happen within seconds and lead a perfectly developed and fully functional application to become deprecated.

3.2 User requirements analysis

3.2.1 User groups

An application can have as a target either a wide group of users or a quite specific one. One very important part of the design process is to determine the user groups of the application and their needs regarding the interface of it. A way of determining those user groups is the creation of personas. A persona is a description of a typical user for whom the application is designed, it is the target of a design.



Illustration 1: John - first persona

First persona: John; John is twenty three years old and he is using his smartphone everyday more and more to make every aspect of his life simpler. He lives in a region where due to strong winds trees fall on the streets quite often. He would love to be able to report such an incident with an easy and quick way instead having to call to the municipality every time, figuring out to whom he should report an urban incident like this and having to describe the problem each time. Moreover, he is usually having a hard time explaining where exactly each incident has happened since some of the roads are on the way to the forest and are not numbered.

3.2.2 Task analysis

Hierarchical task analysis or task analysis is the analysis of the actions that one person will execute, in step by step detail, when using the final application. Its objective is to help the design process by elaborating how the application will be used.

1. Entering the application (If user has already logged in once)

1.1. Incident Report

1.1.1. Capturing the incident

1.1.1.1. Moving the camera towards the incident

1.1.1.2. Tapping the “camera” button that takes the photo

1.1.2. Reviewing the photo taken

1.1.2.1. Tapping the “check” button to move to the next screen

1.1.3. Adding details of the incident

1.1.3.1. Tapping on the description field

1.1.3.2. Typing in the keyboard that appears a short description of the

incident

1.1.3.3. Scrolling downwards the notifications bar

1.1.3.4. Tapping the GPS icon to turn it on

1.1.3.5. Scrolling upwards the notifications bar

1.1.3.6. Tapping the “Emergency level” button

1.1.3.7. Choosing one of the three available emergency levels

1.1.3.8. Tapping the “send” button to submit the incident report

1.1.4. Exiting the application (Not necessary)

1.1.4.1. Pressing the middle (home) button

1.2. View my account

1.2.1. Tapping the “menu” button

1.2.2. Tapping the “My Account” button

1.3. View my reports

1.3.1. Tapping the “menu” button

1.3.2. Tapping the “My Reports” button

1.3.3. Scrolling through my reports

1.4. View my settings

1.4.1. Tapping the “menu” button

1.4.2 Tapping the “My Settings” button

1.5. Log out

1.5.1. Tapping the “menu” button

1.5.2. Tapping the “Log Out” button

Alternative Jobs Flow

1.1.a. (User is not logged in but has account) Typing the user’s email

- 1.2.a. Typing user's password
- 1.3.a. Tapping the "Log in" button
- 1.4.a.a. Redirected to normal flow 1.
- 1.4.a.b Error message appears due to wrong password

- 1.1.b. (User is not logged and doesn't have account) Tapping the "Sign Up" button
- 1.2.b. Typing email
- 1.3.b. Typing password
- 1.4.b. Retyping password to confirm
- 1.5.b. Typing mobile phone number
- 1.6.b. Tapping "Sign Up" button
- 1.7.b. Redirected to normal flow 1.

- 1.1.2.1.a Tapping the "X" button to cancel that shot and take again the picture (go to 1.1.1.)
- 1.1.2.1.b Tapping the "save" button to save the photo to user's device

3.2.3 Usage scenario

A persona is the main character of a scenario. This scenario describes the interface and the actions that have to be taken through the eyes of that character in order to achieve the intended goal of that application (a successful incident report in our case). The scenario defines where, when and how the story of the persona takes place. It is the narrative that describes how the persona behaves as a sequence of events.

Usage scenario: John was getting into his car to drive to his job when he noticed that once again, due to last night's winds, a huge branch of a tree had broken and fallen in the middle of the street. He stops before entering the car and he takes out his smartphone to report the problem. After quickly unlocking it, he opens the CIR and turns his phone in a way that his camera is capturing the broken branch. He takes a photo which is a little bit shaken, he cancels it and he retakes the same one, this time happy with the result. He taps the "next" button and he writes a short description of the incident that he is reporting. Shortly after, he enables his phone's GPS in order to have the incident's coordinates attached to the report and taps the send button to submit his report. Next to that, he gets into his car relieved that he avoided being late for work while in the same time he feels a responsible citizen and an active member of the local community.

3.3 Prototyping

Prototyping is the process of developing prototypes, an integral part of the user-centered design. The main purpose of prototyping is to involve the users in testing design ideas and get their feedback in the early stage of development, thus to reduce the time and cost. It provides an efficient and effective way to refine and optimize interfaces through discussion, exploration, testing and iterative revision.

3.3.1 Wireframes introduction

Wireframing is a phase of prototyping. A wireframe could be described as a sketch, a basic visualization of the final product that represents the layout of that product [4]. Usually it lacks complex graphics and it can be used to measure the practicality and efficacy of the product. There are two main categories of wireframes. The low fidelity and the high fidelity wireframes. The chosen style of wireframes for this application is called “low fidelity” because it focuses only on the very basic points and layout structure without providing any redundant information like color and images. The “high fidelity” wireframes are mostly about computer(device)-based simulation. The determining factor in prototype fidelity is the degree to which the prototype accurately represents the appearance and interaction of the product, not the degree to which the code and other attributes invisible to the user are accurate.

3.3.2 Low fidelity wireframes

The low fidelity wireframes that were created for our application are presented in the Annex [6]. The arrows linking every button with an id number point to the wireframe with that id when tapped.

3.4 Usability Testing

3.4.1 Introduction

Usability testing is a technique used as one of the final steps of a HCI analysis. It is used in user-centered analyses and its main focus is to measure the product’s capacity to meet the intended purpose. In a usability test some users are asked to complete specific tasks while being observed in order to see where they encountered problems or experienced confusion. Based on the observations made, if a problem comes up in multiple different users, a recommendation has to be suggested in order to change something in the implementation and avoid future confusion. The specific method that was used for that testing is called Hallway Testing. That name was given to this method because it utilizes randomly selected users, for example the first ten people passing from a hallway. The only limitation to the users who will test the product excludes people working on that same project since they can be considered experts and in that case we would have the “Expert Review” kind of usability testing.

3.4.2 Testing conditions

The users that helped that testing to take place were the first ten people who entered my house. That included a range age of 14-59 years old people, male and female. The users were video recorded (under their knowledge) while testing the application and notes have also been taken during the process. They were asked the first time of usage to act as they were in hurry and the second time as they had plenty of time to explore the application. The decision of those two scenarios was taken because of the nature of the application which might be used also under time pressure.

3.4.3 Testing results

After studying thoroughly the notes and the videos the biggest flaw that was noted was that five out of ten users got confused in the final view (the one that the user is typing the description of the incident) of the application, after typing the description of the incident. They were not sure how to hide the keyboard in order to tap the “send” button. It is worth mentioning that one of them even had to ask for help to overcome that problem.

3.4.4 Changes made based on the results

Following up the notes made on the usability testing results, a new patch was implemented for the application. That patch included a fix for the “exit the keyboard” aforementioned problem (now the user can tap wherever outside of the keyboard and it will disappear) and some other fixes for some buttons (their borders were increased so the user will have a successful result even if he or she taps a little bit outside of the content of the button).

3.5 Summary

In this chapter the research and analysis that was conducted before the implementation of the application regarding the human-computer interaction was reported. Moreover there has been some explaining on the reasons that such a research is necessary for every application that will reach the production level. Finally we presented an essential part in the development of any digital application, that of usability evaluation with users. In the next chapter, more technical details about the technologies that were used will be presented in order to start having a more concrete image of the final application.

4. IMPLEMENTATION

4.1 Introduction

In this chapter we will discuss the programming languages, the technologies, the frameworks and the environments that were used for the implementation of this application as well as the reasons behind the choices that were made about them. The implementation of this application can be divided in two parts since it utilizes the client-server model. The first one is the server side of the application. A server is a computer program or a device that provides functionality for other programs or devices called “clients”. A single server can serve multiple clients and a single client can use many servers. Usually the functionalities that a server provides are called “services” and they are mainly providing or receiving data from the clients through the Internet. In our case, where we are dealing with a mobile application, we need an application server. An example of an application server could be a server that provides a weather application with data about the weather at the location of the user. The server of this application provides the functionality of receiving incident reports from Android smartphones, accepting or rejecting user credentials during the log in of a client as well as sending to a client his or her previous incident reports. The second part of this application is the client side. A client is a computer program or a device that accesses a service made available by a server. A client does not share resources but requests data or functionality results from the server. Examples of other kind of applications utilizing the client-server model are Email and World Wide Web. We have an Android application running in Android smartphones operating as a client. The user is interacting with this application while the authorities will use the interface of the server in order to receive the incidents.

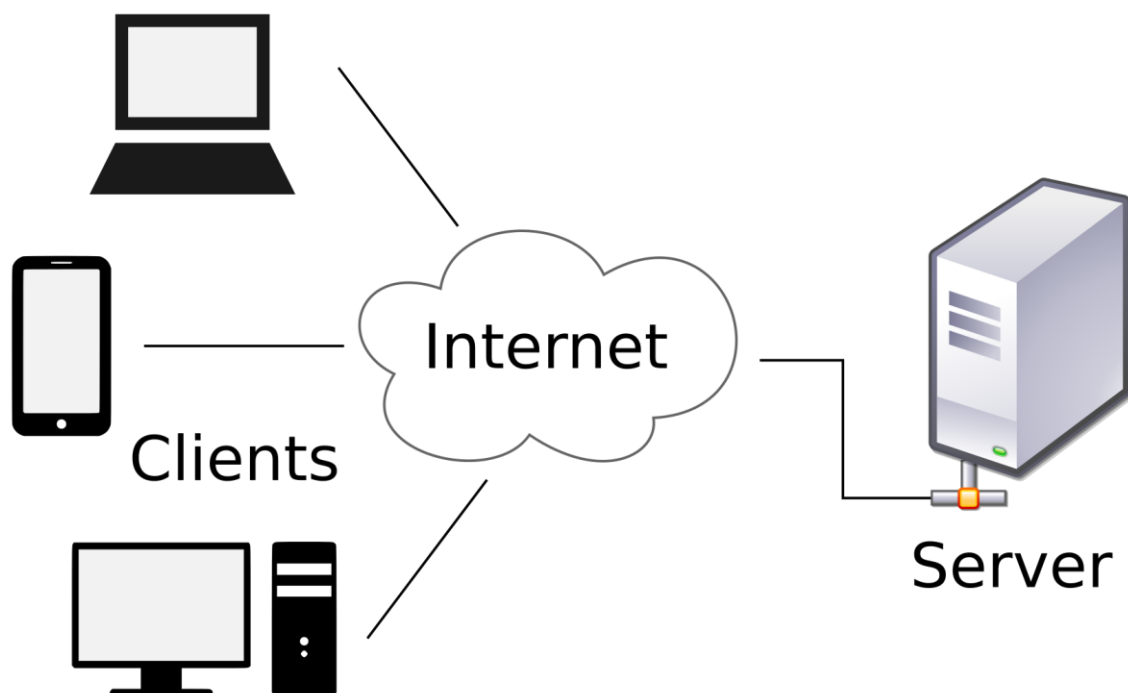


Illustration 34: Example of the client-server model. A laptop a smartphone and a desktop are all three clients accessing the resources of the server.

4.2 Server-side

4.2.1 REST

The architectural style that was decided to be used is REST. REST or Representational State Transfer is a software architectural style whose main focus is to provide standards of communications between computer systems usually using the Internet. Those systems are called RESTful systems. The basic concepts followed by all REST frameworks could be summarized in four ideas.

- The first is the idea of statelessness; the server does not need any information about the state of the client in order to have a successful communication. The same stands for the client in order to communicate with the server. Every communication is based on standard operations and resources.
- The second idea is the idea of the request; clients send requests to retrieve or modify data from the server and the server sends responses to these requests. A request is a plain-text message in ASCII [12] encoding. A request message consists of a request line which may request a resource from the server, request header fields with one of those possibly being the language, an empty line and an optimal message body. The basic types of requests used are the Hypertext Transfer Protocol (HTTP) requests GET, POST, PUT, DELETE. HTTP is an application protocol in which is based the communication of the World Wide Web (WWW). It functions as a request-response protocol and it is used in the client-server model. A communication based on the HTTP protocol could be thought as a real life dialog between two people. The first one would be the client and the other one would be the server. They are both talking in turns, the first one asking questions and the second one responding to those questions. It is used by REST in order to provide the standards of communication that are used.

Request-line	Get /products/dvd.htm HTTP/1.1
General Header	Host:www.videoequip.com Cache-Control:no-cache Connection:Keep-Alive
Request Header	Content-Length:133 Accept-Language:en-us . . .
Entity Header	Content-Length:133 Content-Language:en . . .
Body	

Illustration 35: Example of the contents of a HTTP request.

- The third idea behind REST is the idea of the paths; every request is bound to a path which each request should have as destination. A path can be called also a Uniform Resource Identifier (URI). A path has the following syntax: `https://{servername}/{resource path}`. The server name field is self explanatory. The resource path is a path that defines a singleton resource. In order to achieve that every resource path includes a unique id. These paths layouts usually are designed to be user-friendly and easy to understand, for example in a path like this `myServer/incidents/15` the request will have as target in myServer from all the

incidents the one with id 15. In contrast to these simple paths layouts the reader may have seen paths of this kind “https://mySite/i=&dniOIGG#ASbbm&uPOu” that are not really enhancing the user experience.

- The last idea is the idea of the response; the server answers to the requests by sending responses back while utilizing two main concepts. The concept of the content-type, which is a field in the header of the response and declares to the client the type of data that is sent back, and the concept of response code which informs the client regarding the success if its request and in the case of failure a reason for the failure.

4.2.2 Django – Django REST Framework

The framework used on the server side is Django which is a high-level Python Web framework provided as free open-source. It uses the Model-View-Template (MVT) architectural pattern. In this kind of architectural pattern the model is responsible for managing the data of the application. It responds to the request from the “view” and it also responds to instructions from the template to update itself. The “view” means presentation of data in a particular format, triggered by a template's decision to present the data. The template is a HTML file mixed with Django Template Language (DTL). Django’s main benefits that attract developers are the reusability and “plugability” of contents which can lead to fast paced and very productive development while in the same time it prevents developers from code repetition. Django REST Framework is a toolkit mainly used for web Application Programming Interfaces (API). It is actually an extension for the Django framework. It provides to the developer some very useful tools that are implemented in almost every web application. Some of these are the authentication policies including packages for OAuth1a and OAuth2 and the Serializers. Serializers are methods that allow complex data types and data models that are written in Python to be easily rendered into JavaScript Object Notation (JSON) or Extensible Markup Language (XML). JSON is an open-standard file format that is mainly used for asynchronous communication between a server and a client. It consists by a set of key-value pairs that are written in human-readable text form. Serializers can prove very useful when building an application that will use multiple programming languages. In our application they are used to transmit data from the server to the client and they are extremely useful because they are enabling us to send a Java object to a Python server which otherwise would be a very difficult task. Their main functionality is “translating” a Python object to JSON format that can be read by the client as well as “translating” a JSON formatted object that is being received by the server to a Python object. Following there are some code examples of Serializers, JSON and XML from our application.

```

1  from rest_framework import serializers
2  from myServer.models import IncidentReport, User
3  import time
4
5  class UserSerializer(serializers.ModelSerializer):
6      class Meta:
7          model = User
8          fields = '__all__'
9          # fields = ('id', 'email', 'password', 'phone',)
10
11 class IncidentReportSerializer(serializers.ModelSerializer):
12     image = Base64ImageField(
13         max_length=None, use_url=True,
14     )
15     user = UserSerializer(read_only = True)
16     user_id = serializers.PrimaryKeyRelatedField(
17         queryset=User.objects.all(), source='user', write_only=True)
18
19     class Meta:
20         model = IncidentReport
21         fields = '__all__'
22

```

Illustration 36: Example of the Django REST Serializer that was used.

```

1  {
2      "description": "The garbages are left out of the bins polluting the environment.",
3      "image": "UklGRlowAABXRUJQVLA4WAoAAAAoAAAAAQAA3QAA",
4      "emergencyLevel": "50",
5      "user_id": "2",
6      "latitude": "38.025805",
7      "longitude": "23.848238"
8  }

```

Illustration 37: Example of incident submission request in JSON format.

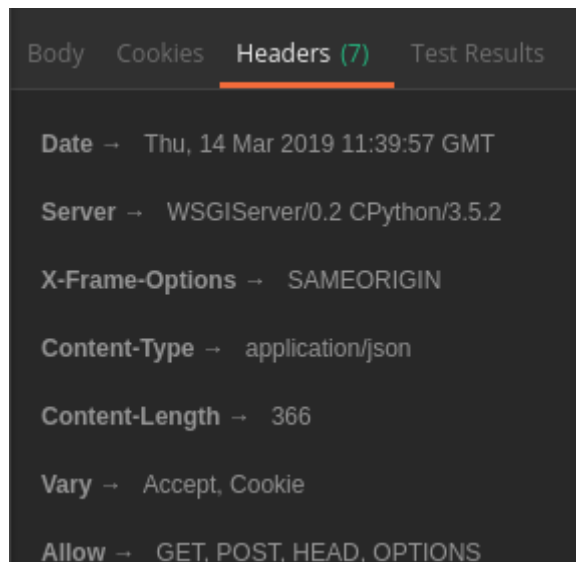
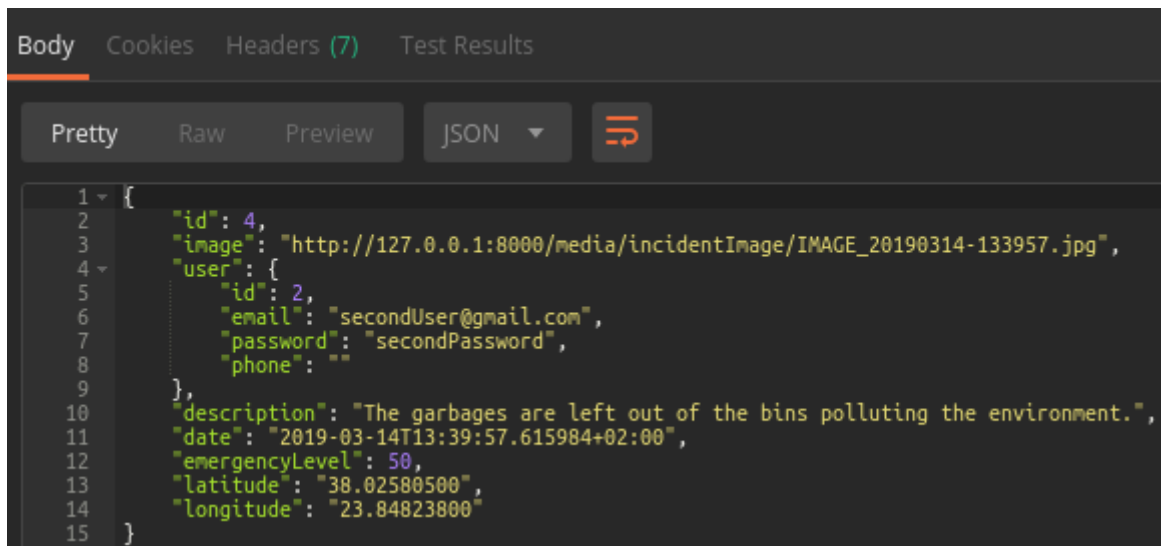


Illustration 38: Example of the header of the response to the previous request.

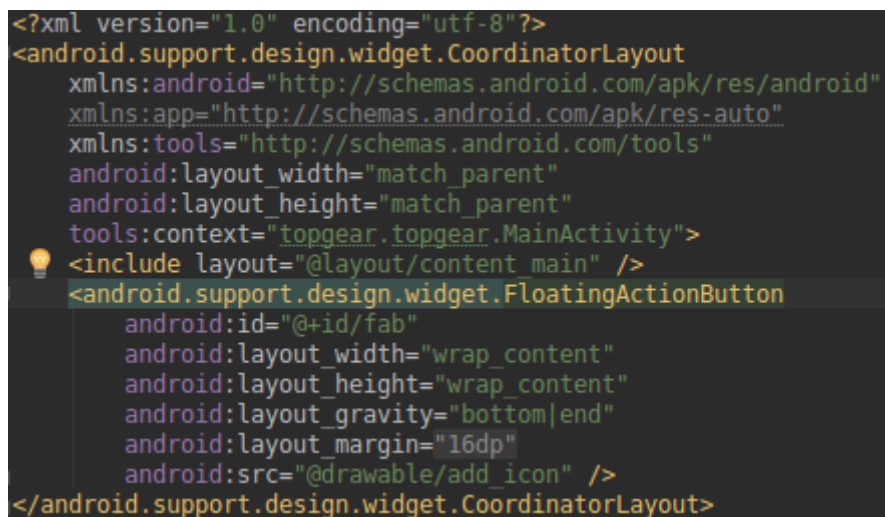


```

Body  Cookies  Headers (7)  Test Results
Pretty  Raw  Preview  JSON
1 {
2   "id": 4,
3   "image": "http://127.0.0.1:8000/media/incidentImage/IMAGE_20190314-133957.jpg",
4   "user": {
5     "id": 2,
6     "email": "secondUser@gmail.com",
7     "password": "secondPassword",
8     "phone": ""
9   },
10  "description": "The garbages are left out of the bins polluting the environment.",
11  "date": "2019-03-14T13:39:57.615984+02:00",
12  "emergencyLevel": 50,
13  "latitude": "38.02580500",
14  "longitude": "23.84823800"
15 }

```

Illustration 39: Example of the body of the response to the previous request in JSON format.



```

<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
  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:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context="topgear.topgear.MainActivity">
  <include layout="@layout/content_main" />
  <android.support.design.widget.FloatingActionButton
    android:id="@+id/fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|end"
    android:layout_margin="16dp"
    android:src="@drawable/add_icon" />
</android.support.design.widget.CoordinatorLayout>

```

Illustration 40: Example of XML code used in our application.

Django and Django REST are recognized and used by international companies like Mozilla, Facebook and Instagram [5]. One of the reasons that Django was chosen as the server-side framework was to explore and learn how to resolve any kind of complications that could occur during the development of an application that is using in the same time Java (for the client/Android front-end) and Python (for the server). One more reason was the customizable administrator page that Django provides which is more suitable for this application keeping in mind that the incidents are only going to be reviewed by “administrators” such as government authorities and the clients will have access to the application only through their Android smartphones. Moreover, Django is scalable and reliable which gives us the perspective of an easier expansion of the application in the future.

4.3 Android application

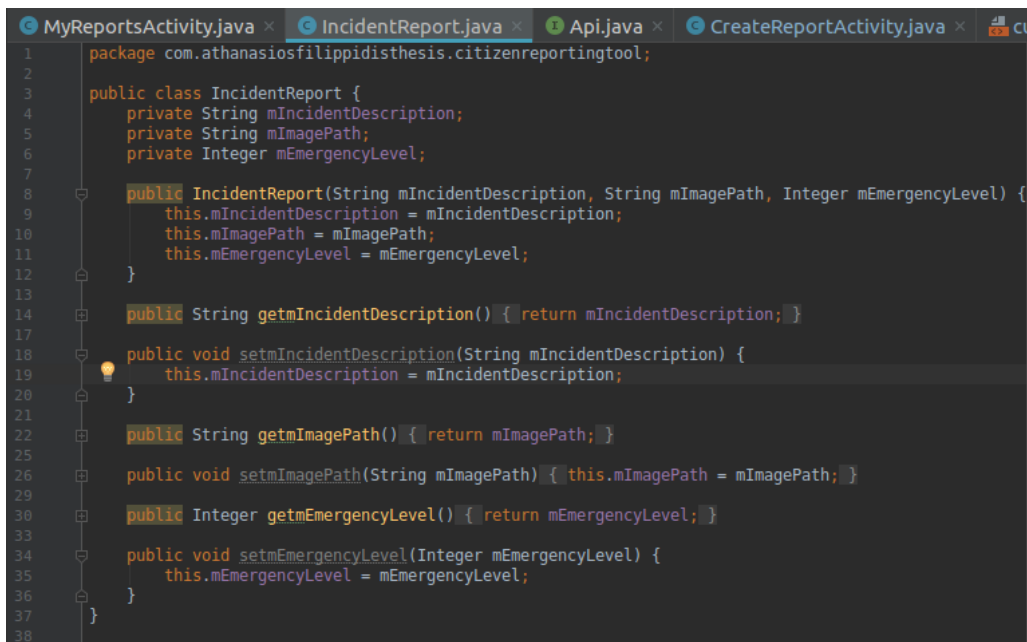
4.3.1 Android operating system

Android is a mobile-operating system developed by Google. Its core is written in C and its User Interface (UI) is written in Java. It is a Unix-based operating system that was created to be used for touch screen mobile devices such as smartphones and tablets although now days it has expanded to all kind of smart devices – from watches to televisions. Google officially supports the development of Android applications written in Java, Kotlin

and C++ but the use of other languages is also possible by specific techniques. The UI of an Android application is written in Extensible Markup Language (XML) while in the same time it can also be controlled and affected by the application's development language (i.e. Java). The environment usually used to develop an Android application is the Android Studio, the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. The operating systems on which Android Studio is available are all the main three – Windows, macOS and Linux.

4.3.2 Android development

An Android application is composed by one or more views often referred also as “activities”. Those views can be considered as separate pages of our application. Each view consists of at least one XML and one Java class file. The XML file is responsible for stating clearly the visual layout of that view by declaring constraints between the objects that will be shown in this view and giving unique id's to those objects. The Java class file is used to add the functionality in that view.



```

1 package com.athanasiosfilippidisthesis.citizenreportingtool;
2
3 public class IncidentReport {
4     private String mIncidentDescription;
5     private String mImagePath;
6     private Integer mEmergencyLevel;
7
8     public IncidentReport(String mIncidentDescription, String mImagePath, Integer mEmergencyLevel) {
9         this.mIncidentDescription = mIncidentDescription;
10        this.mImagePath = mImagePath;
11        this.mEmergencyLevel = mEmergencyLevel;
12    }
13
14    public String getmIncidentDescription() { return mIncidentDescription; }
15
16    public void setmIncidentDescription(String mIncidentDescription) {
17        this.mIncidentDescription = mIncidentDescription;
18    }
19
20    public String getmImagePath() { return mImagePath; }
21
22    public void setmImagePath(String mImagePath) { this.mImagePath = mImagePath; }
23
24    public Integer getmEmergencyLevel() { return mEmergencyLevel; }
25
26    public void setmEmergencyLevel(Integer mEmergencyLevel) {
27        this.mEmergencyLevel = mEmergencyLevel;
28    }
29
30 }
31
32 }
33
34
35
36
37
38

```

Illustration 41: An example of a Java Class: IncidentReport Class represents an Incident, it contains the image, the description of the incident and the emergency level of the incident

For example, if we have a view with a single button, the color and the position of the button would be specified in the XML file but the action that is taken at the tap of the button is specified in the Java file.



```

45 <ImageButton
46     android:id="@+id/menuBtn"
47     android:layout_width="40dp"
48     android:layout_height="40dp"
49     android:layout_marginTop="16dp"
50     android:layout_marginEnd="16dp"
51     app:layout_constraintEnd_toEndOf="parent"
52     app:layout_constraintTop_toTopOf="parent"
53     android:background="?android:selectableItemBackground"
54     app:srcCompat="@drawable/ic_round_menu_24px"
55     android:contentDescription="@string/menu_btn"
56     android:onClick="toMenu" />

```

Illustration 42: The button that takes the user to the main menu of the application

In order to have a functional view/activity the class of that activity has to extend one of the standard Android activity classes (for example “AppCompatActivity” class). Moreover, a basic step in creating a functional activity is to override some standard methods of it in

order to customize them to fit the special needs of the functionality being implemented. (For example, the “onCreate” method, responsible for the actions that have to be taken at the first time this view is created, can be overridden so it will load the custom resources of our view). In the following section, we are going to present some basic external libraries that were used during the development of the Android application.

4.3.3 Picasso

Picasso is a free open-source library developed by Square Open Source [6]. Its main focus is on downloading and caching images in Android. It achieves an optimized image approach by using adapter downloads which detect an already cached image and prevent the application from downloading it again, automated image transformation algorithms that optimize the image in order to fit in each view as well as reduce the memory size. In the same time, Picasso, provides generic methods that can be applied to multiple types of image objects. It is used in our application for all the types of image handling that are take place, from capturing and saving an image to projecting it in multiple different views of our application.

4.3.4 Retrofit

Retrofit is also a free open-source library developed by Square Open Source [7]. Its main focus is on providing a type safe HTTP client for Android and Java. That is happening by making available for the user five built-in annotations regarding the request method that can be used when the client is building a request as well as appropriate annotations for the modification of the path, the header and the content of a request. Furthermore it gives to the developer the option of making those request calls synchronously or asynchronously which can be an extremely useful feature in Android where a synchronous call could block the main thread and lead in delays on the render of the UI. In coordination with Retrofit we used also gson, a Java serialization/deserialization open-source library to convert Java Objects into JavaScript Object Notation (JSON) and back provided by Google.

4.3.5 Google Mobile Services (GMS)

GMS [8] is a collection of Google applications and APIs that help support functionality across devices. One of those applications is the location API. This API was used in order to ensure that the user’s coordinates at the time of the incident reporting are sent along the other information regarding the incident.

4.3.6 Camera 2 API

One of the main decisions that had to be made during the development of the application was the version of the Android camera API that was going to be used. The two choices were the Camera API and the Camera 2 API [9]. The first one is a very well documented API, with multiple examples of uses available and has been fully explored by developers. The only drawback in this one is that from Android 5.0 and later it was declared deprecated by Google and to be replaced by the Camera 2 API. Even though that was the case, by taking advantage of the backward compatibility property that the Android Operating System has, most of the developers have stuck with the older API and still use that instead of the newest one. That has led to a lack of thrive for the newest one which means lack of documentation and lack of examples of usage. Yet, it was decided to use the Camera 2 API having in mind that this application should have an as long as possible life-cycle, and using an officially deprecated API could be a burden on that. Moreover, the new API provides some features regarding the quality of the images as well as the opportunity of creating a custom camera view instead of using the standard one. We took

advantage of this opportunity and indeed created a custom camera functionality for our application.

4.4 Presentation of the application

4.4.1 Android client-side

This section will present the UI of our Android application explaining the purposes of each view.



Illustration 2: The intro view of the application; a loading screen to provide some time for the resources needed to load.

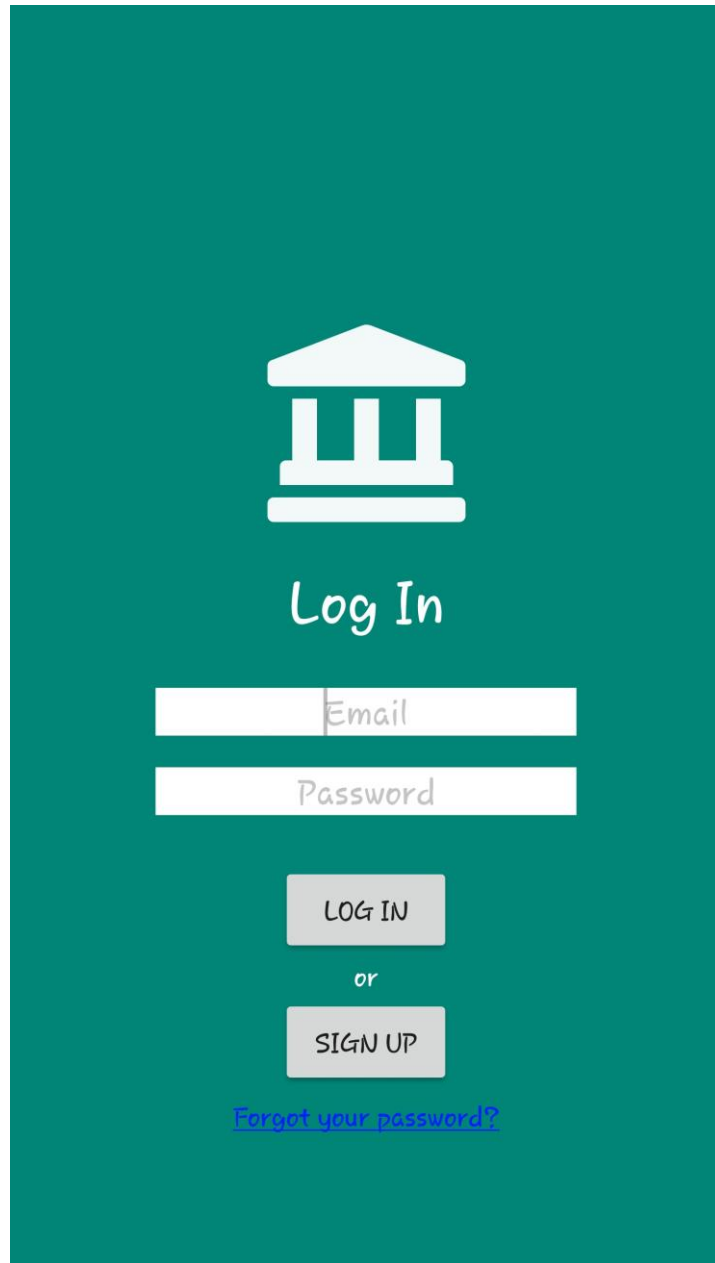


Illustration 3: Log in view; the user can type his or her credentials in order to log in to his or her account, to sign up or to retrieve the account's password. If the user has already logged in before he or she will be redirected to the home view without having to log in again.

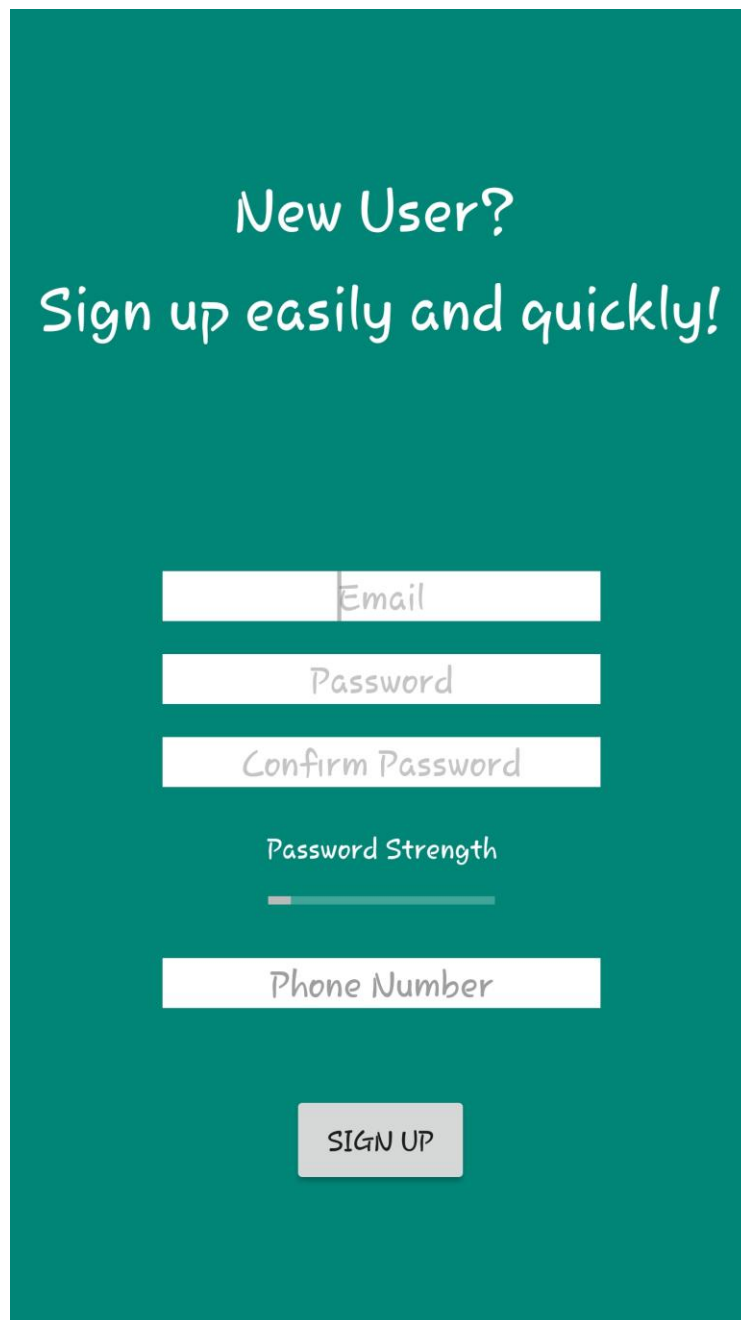


Illustration 4: Sign up view; the user has to type a valid email, a password (while the strength bar below will show its strength dynamically) and a valid phone number for verification reasons.



Illustration 5: Home - Capture incident view; User can turn on and off the flash, can tap on the menu button to navigate to the application's menu or can tap the camera button to capture an incident.



Illustration 6: Review image view; in this view the user can review the image captured and can decide between canceling this capture by tapping the "X" button, saving the image for personal use by tapping the save button (in which case the pop up that can be seen is displayed to inform him or her about the success of his or her action), or tap the check button to proceed with the creation of the report.

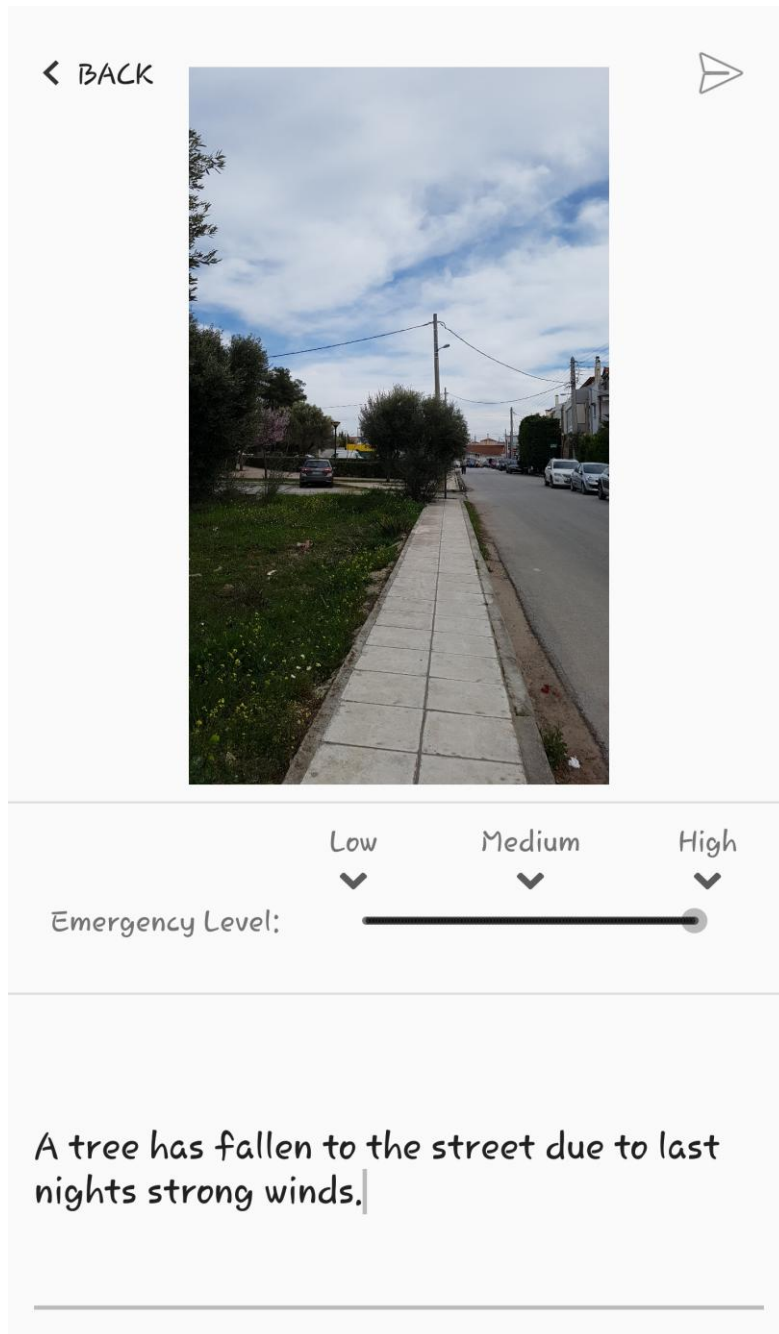


Illustration 7: Create incident report view; In this view the user can tap on the text area to type a short description of the incident, can move the slide bar to determine the emergency level of the incident reported, can tap the back button to return to the review image view or can finally tap the send button to send the review to the server.



Illustration 8: Successful incident report view; this is the view that the user is redirected after sending the report, a message appears also to inform him or her about the success of the report.

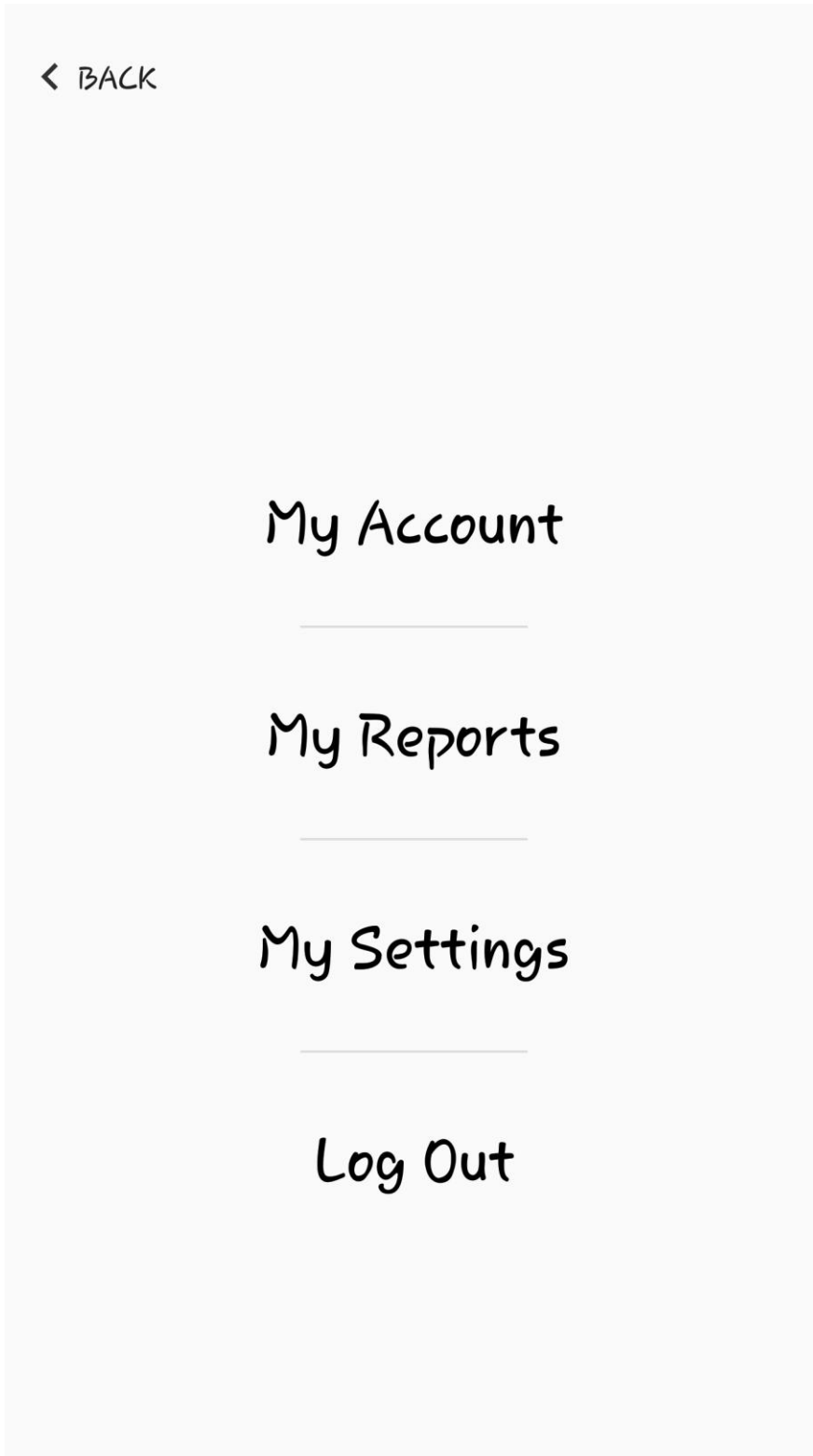


Illustration 9: Menu view; here the user can either tap the back button to go back to the home view or tap one of the four choices in order to be redirected in their respective views.

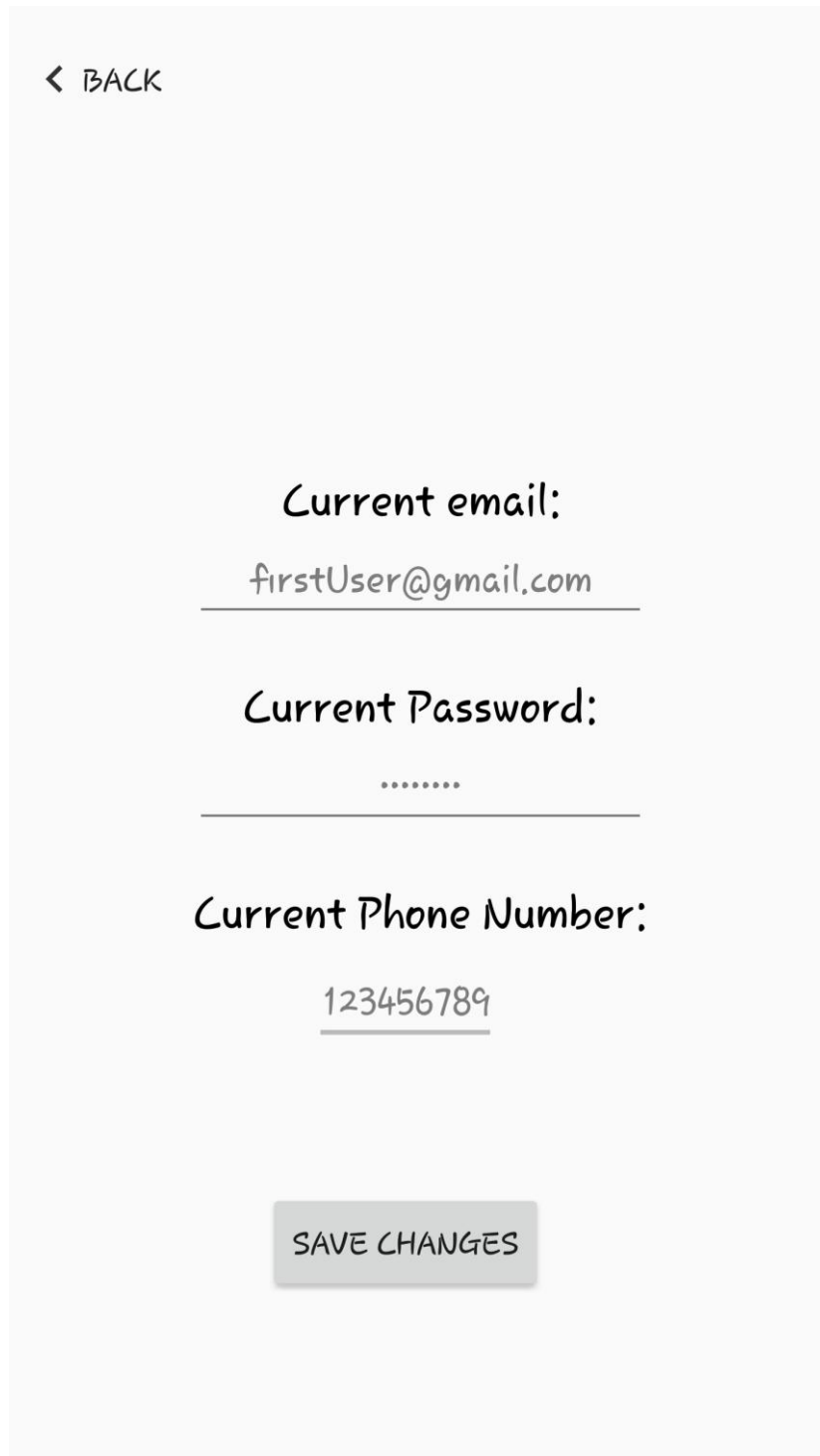


Illustration 10: My account view; in this view the user can see his or her credentials and change them by tapping the "save changes" button or go back to the menu by tapping the "back" button.

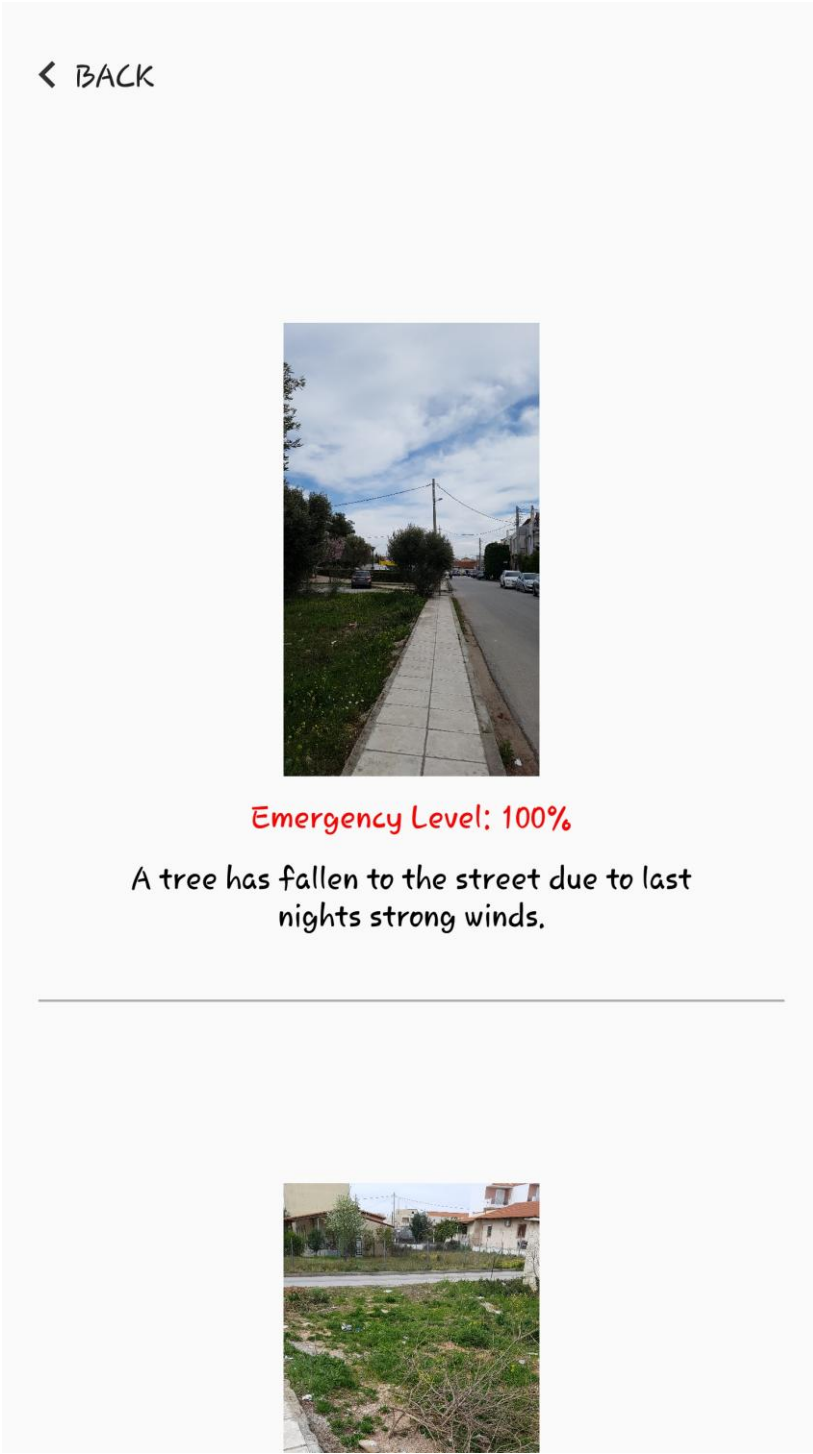


Illustration 11: My reports view; in this view the user can see his or her previous reports as well as to go back b tapping the "back" button.

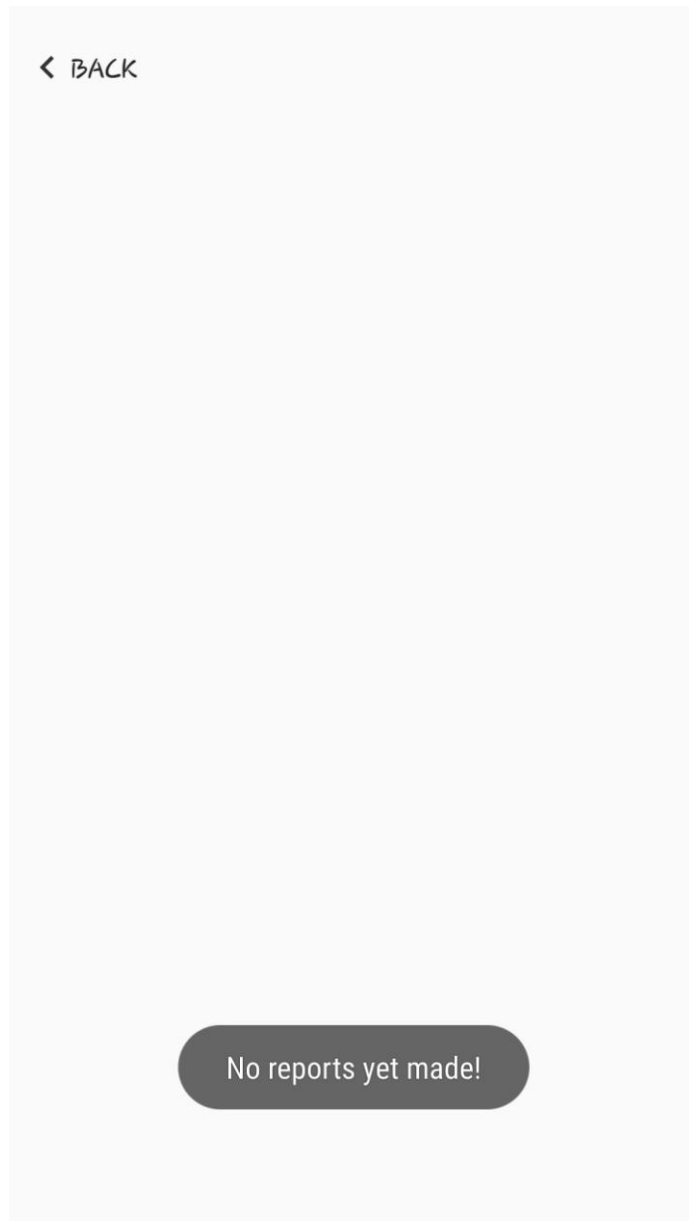


Illustration 12: Alternative my reports view; this is how the “my reports” view looks like when there have been no reports by this user.

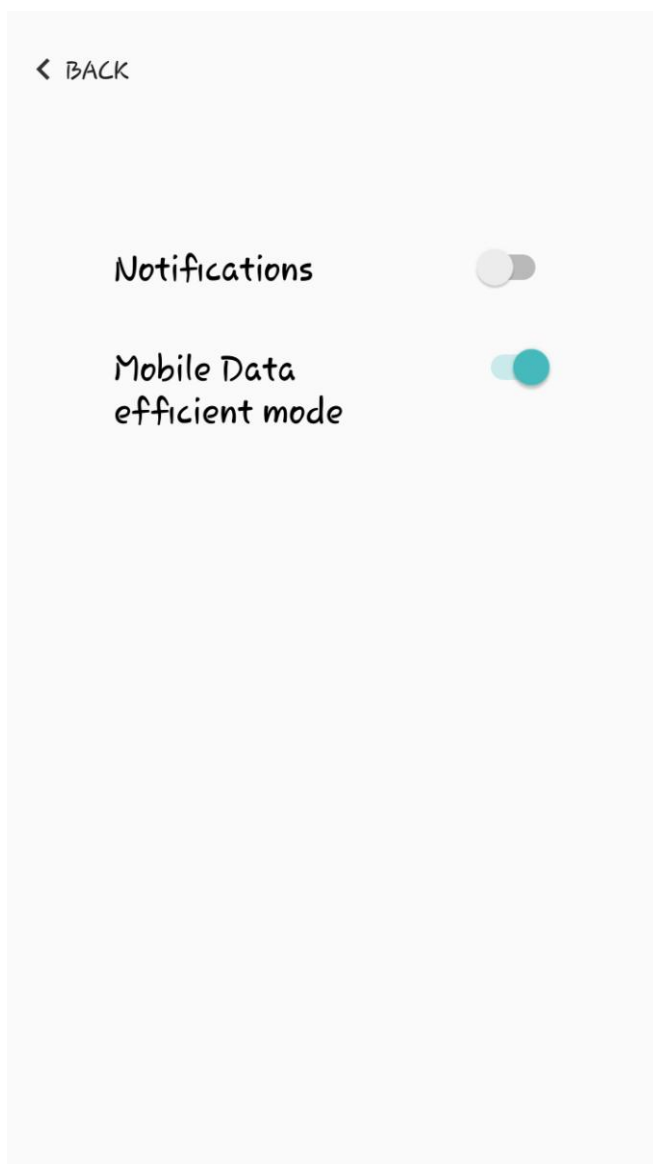


Illustration 13: My settings view; in this view the user can set his or her personal settings (if he or she wants to receive notifications from the application or to use a mobile data efficient mode of the app to avoid spending too much mobile data).

4.4.2 Django server-side

This section shows the UI of the server application which will receive the reports. Below, the administrator of CIR application is able to review the incidents reported by the citizens and assign them to the right department in order to get resolved.

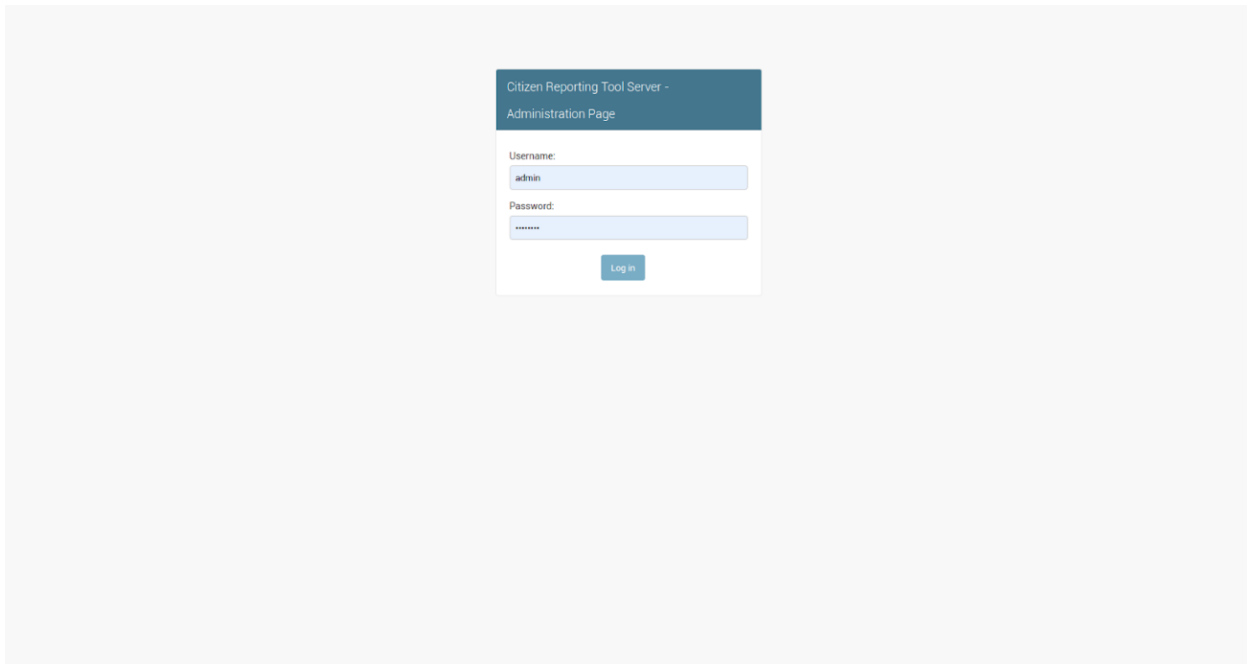


Illustration 15: Administrator Log In page



Illustration 14: Administrator Home Page. We have implemented the two main models required, user and incident model. We can always add or edit a new instance of those from the administrator page.

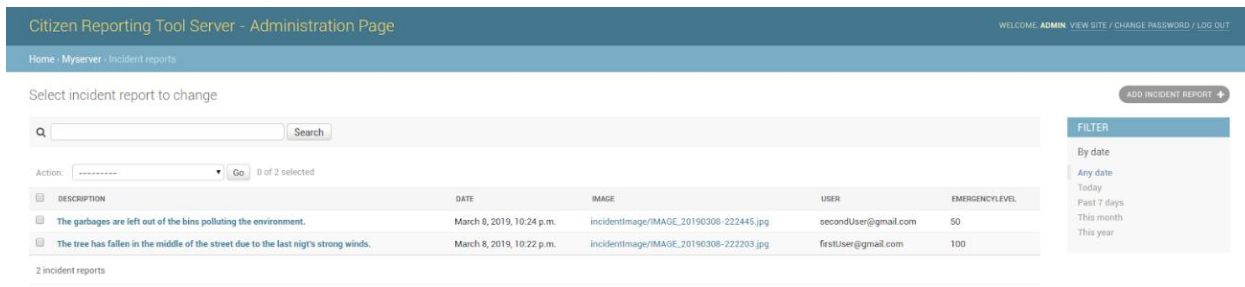


Illustration 16: Incidents List Page. A search field is implemented, so the administrator can search for keywords in reports, the incidents can be sorted by each of their fields simply by clicking on the name of a field and there is a date filter implemented to help tracking incidents by the time they occurred.

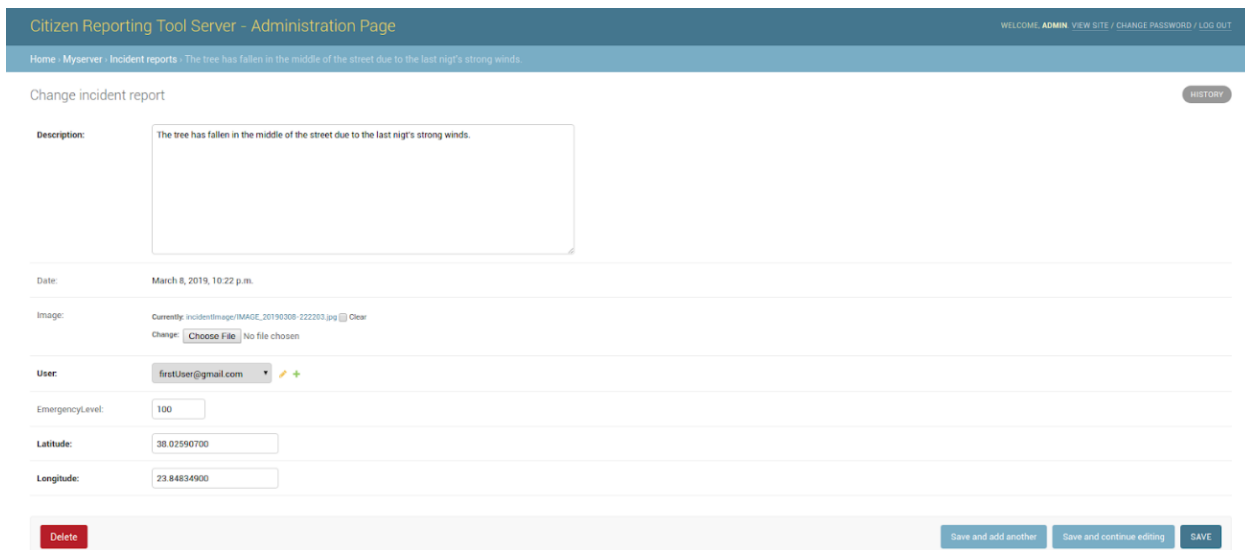


Illustration 17: Incident's details page. There the administrator can inspect all the details of an incident as well as click on the image of it to gain more information about it. Moreover the administrator has the option of editing some of the details in case there has been a mistake.

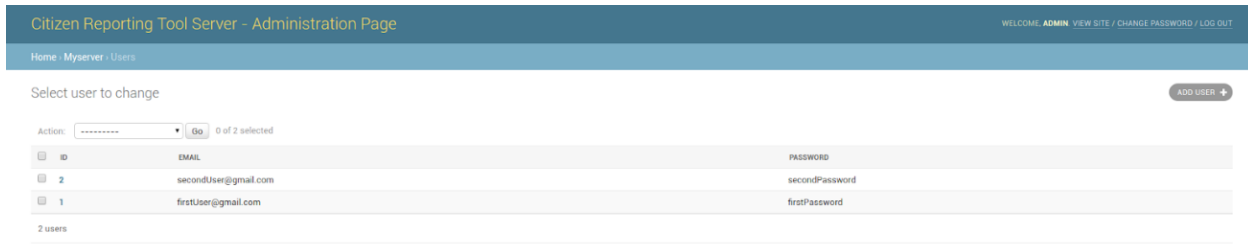


Illustration 18: Users List Page. There the administrator can view all the signed up users as well as their id's.

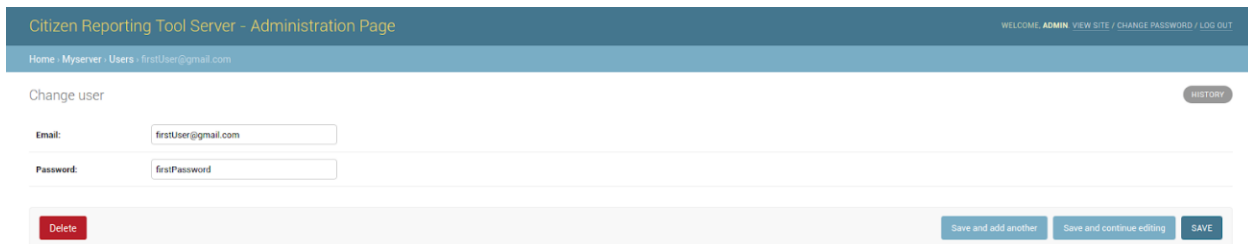


Illustration 19: User's details page. There the administrator can see all the details of a user as well make any necessary changes in their accounts.

4.5 Summary

In this chapter we introduced the technologies that were used for the implementation of the application while providing some basic knowledge for each one. Moreover we provided images of every view that was developed in both sides of our application; front and back end in order to familiarize a user with every aspect of our application. We tried to give a short description of each of these views and the implemented actions that a user can take on them. Having said that there is also no doubt that there are many ways in which this application could evolve in the future and it could be interesting to suggest some of these.

5. CONCLUSION

5.1 Summary

In this thesis we have presented the full cycle of the development of an Android application. We demonstrated the sequence of steps that constitute a User-Centered Design in order to produce a user-friendly final product. We have also demonstrated the model on which most of the applications using the Internet are based; the client-server model. Following on that we presented one of the most used software architectural styles, REST. Based on this information we explained the implementation choices that were made regarding both the server, on which we used the Python-based Django and the client on which we used mainly Java. The sequence of steps that we took gave us the opportunity to acquire a close to production-level Android application development experience. The end-product is developed in a way that it could shortly be available for public use.

5.2 Future Development

Undoubtedly there are many more aspects of this application that could be developed further and become a project on their own. The most intriguing of those probably is an image recognition extension on the server-side of our application that would be specifically developed to recognize the different types of incidents, categorize them and forward them to the appropriate authority. An extension of this kind could also take advantage of the descriptions provided in the incidents. It could look for specific keywords in them and use those to make the image recognition part more accurate. One more aspect of the already implemented application that could be developed is a web-client site in order to give to the users the ability to review their previous reports or other statistical details that could be interesting after a certain amount of data has been gathered. Furthermore, an interesting aspect could be the creation of a custom administrator page for each of the departments and the authorities concerned. This way the officers would have the chance to resolve those incidents more efficiently as well as to provide a live tracking feed on the evolution of that process. That live feed could be implemented by reporting the current action that is taking place at each moment regarding each incident. Finally, when the incident is resolved, it could be marked as such. One more intriguing perspective of that development could be the statistics like the percentage of incidents resolved by each department or the average time that was used for those incidents to get resolved. Those statistics could be used also in order to predict incidents in specific areas and to give the chance to the municipality to be ready to encounter them. Moreover, later, that could be used as a measure of efficiency of a municipality. One last way in which this application could be evolved is a cooperation with Google's Google Maps. Some of the public incidents, especially the most urgent ones, usually tend to have a huge impact on the traffic of the cities. If someone could report an incident in real time and in a way that it can be verified (a function already provided by our application) the Google Maps application could spread notifications to the citizens that are going to pass by that location either by passively notifying them about the incident or even by actively suggesting a different route to them so they can avoid the traffic jam.

6. ANNEX

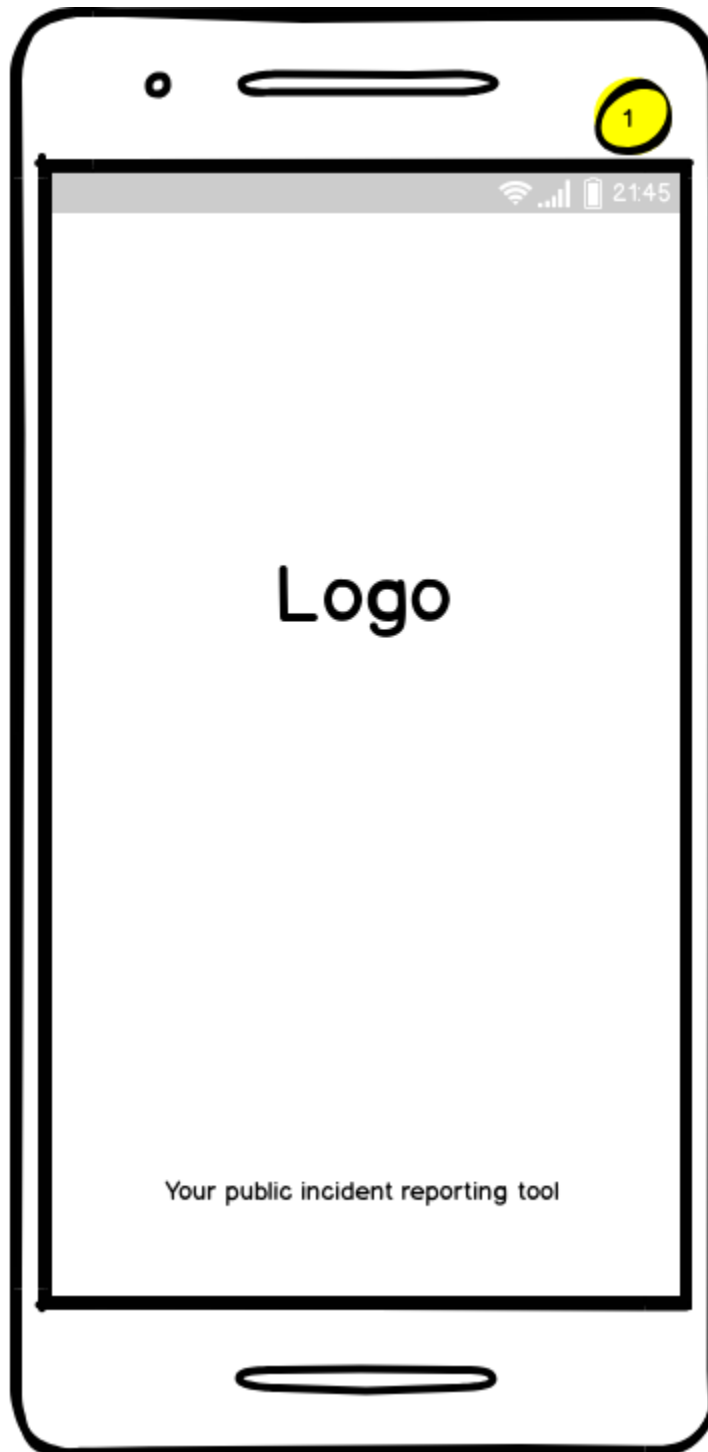


Illustration 20: Intro loading page wireframe

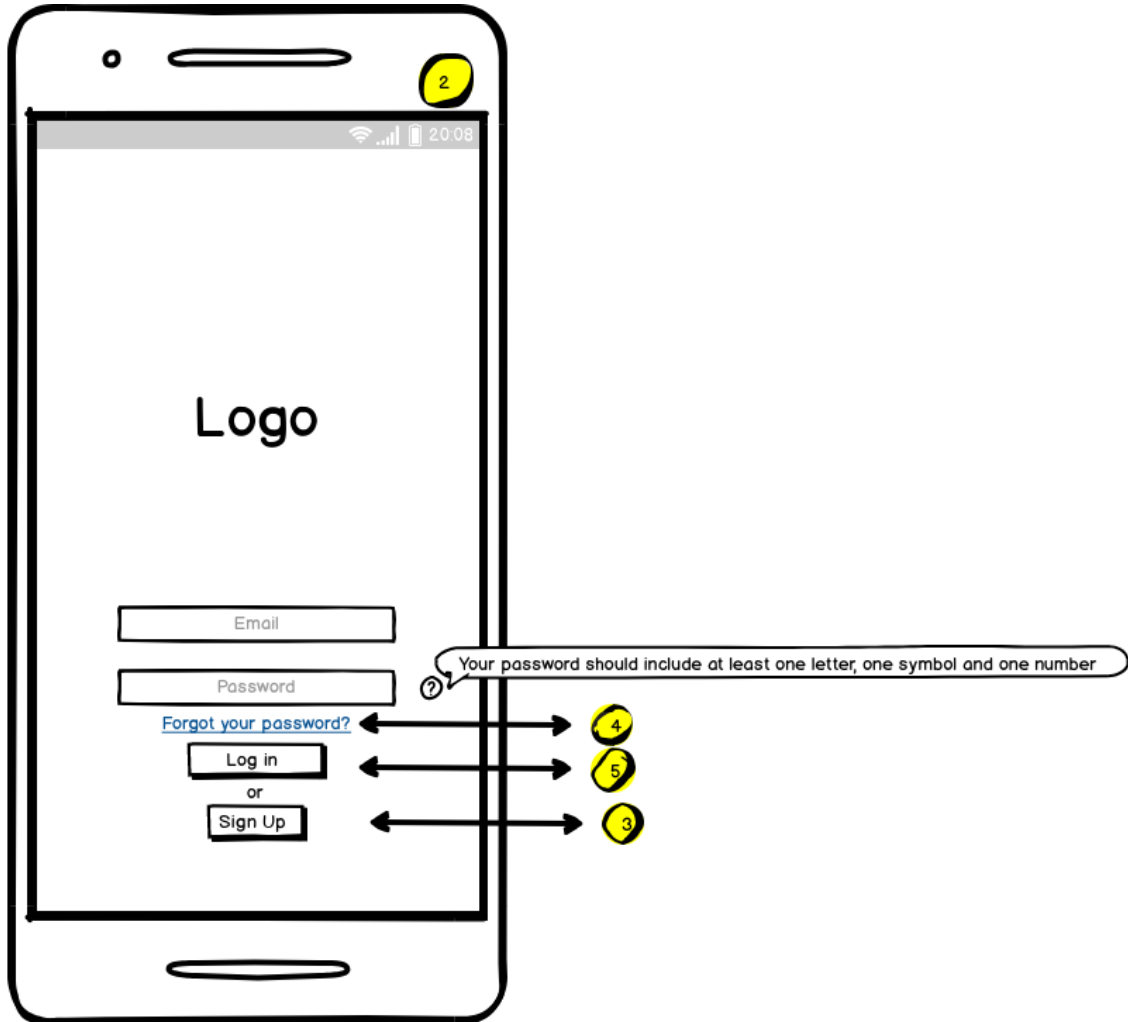


Illustration 21: Log in wireframe

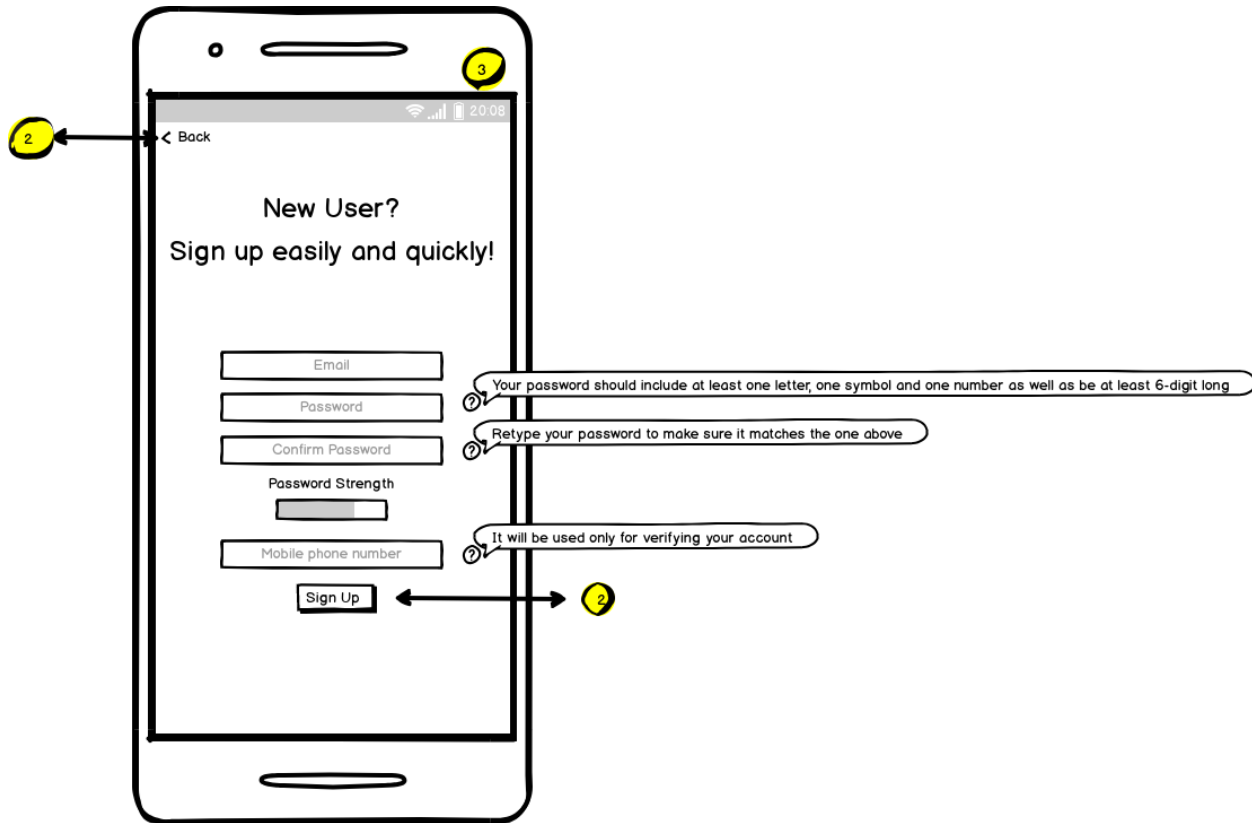


Illustration 22: Sign up wireframe

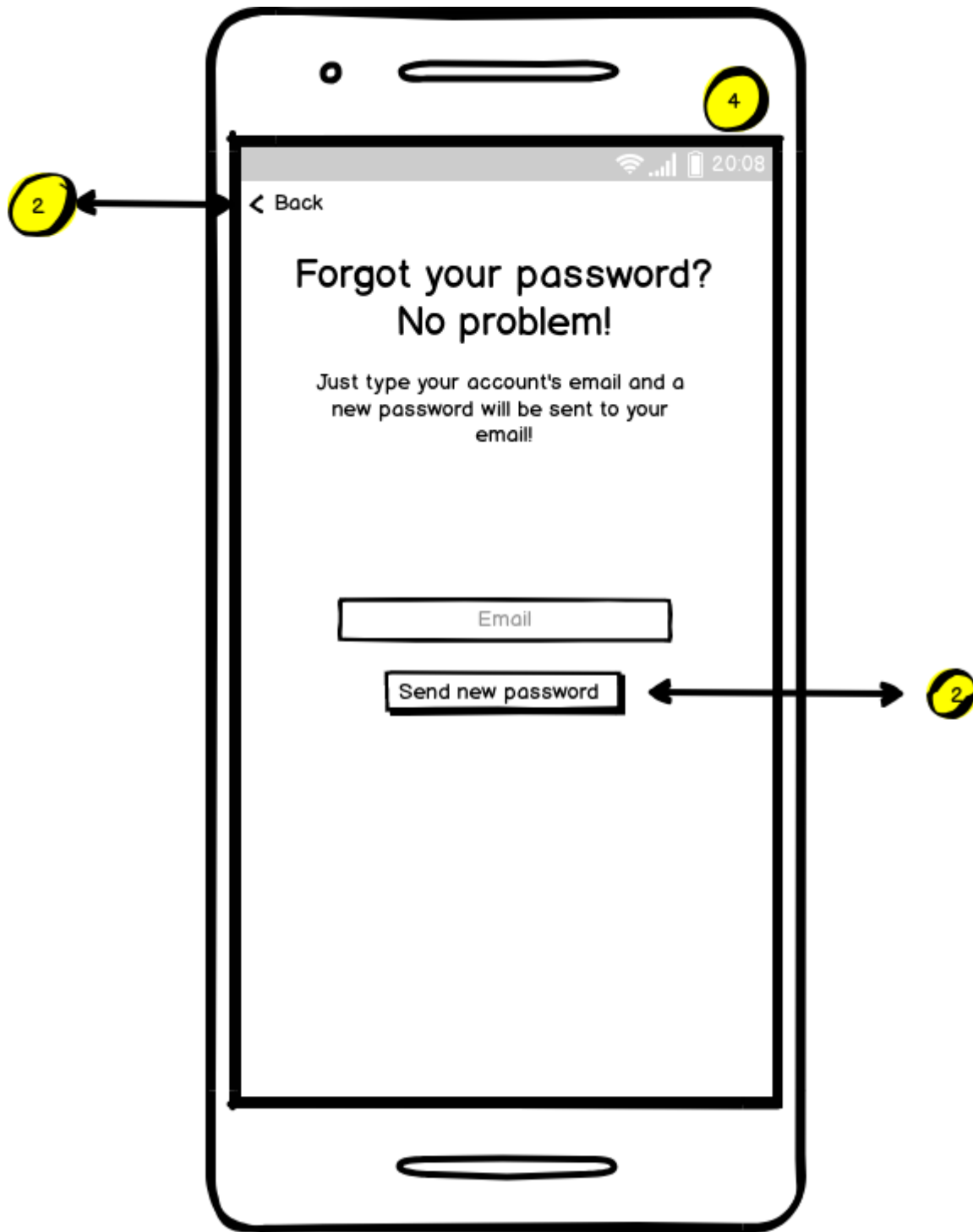


Illustration 23: Forgot your password? wireframe

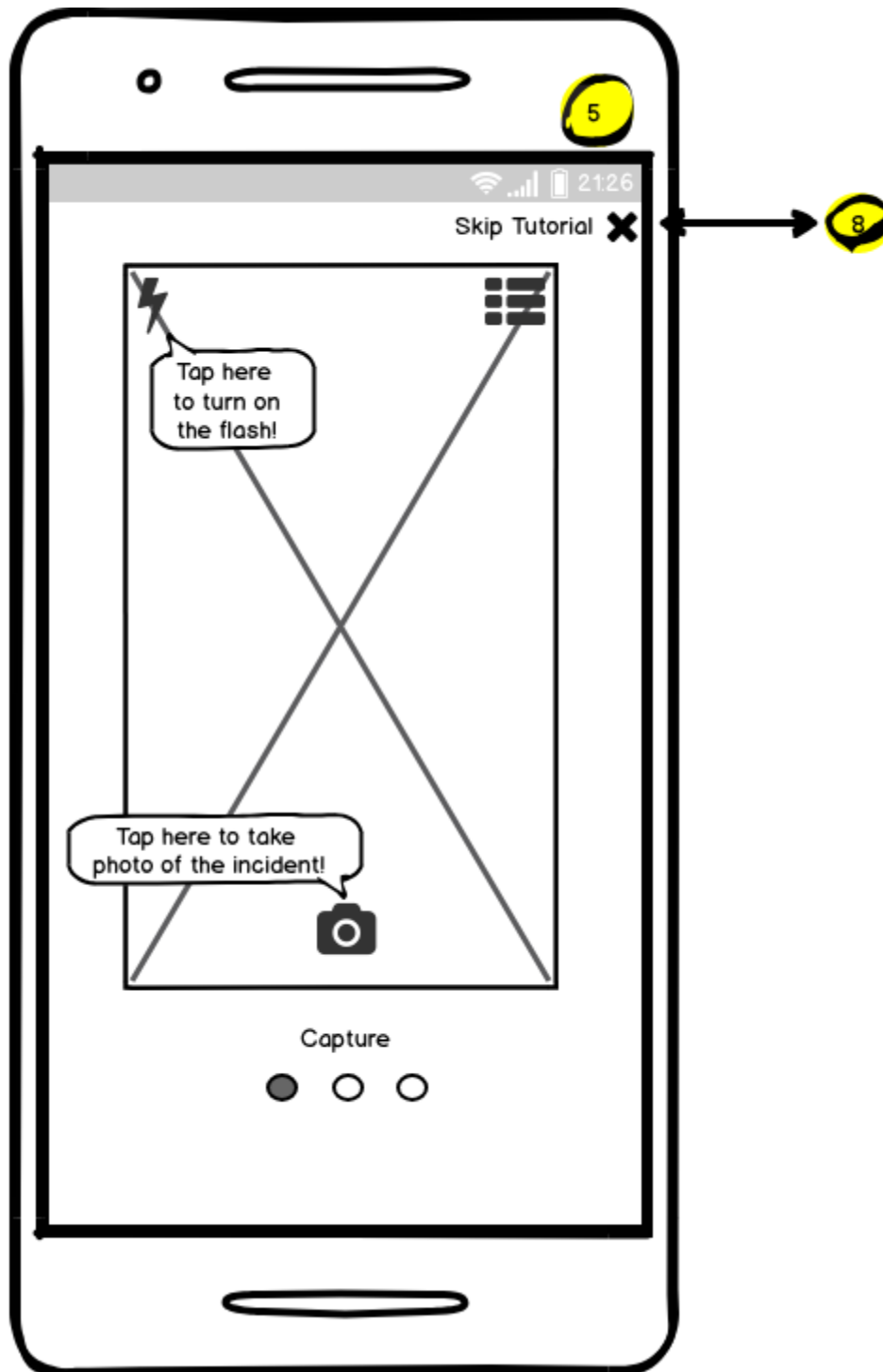


Illustration 24: Home wireframe during first user entrance showing a

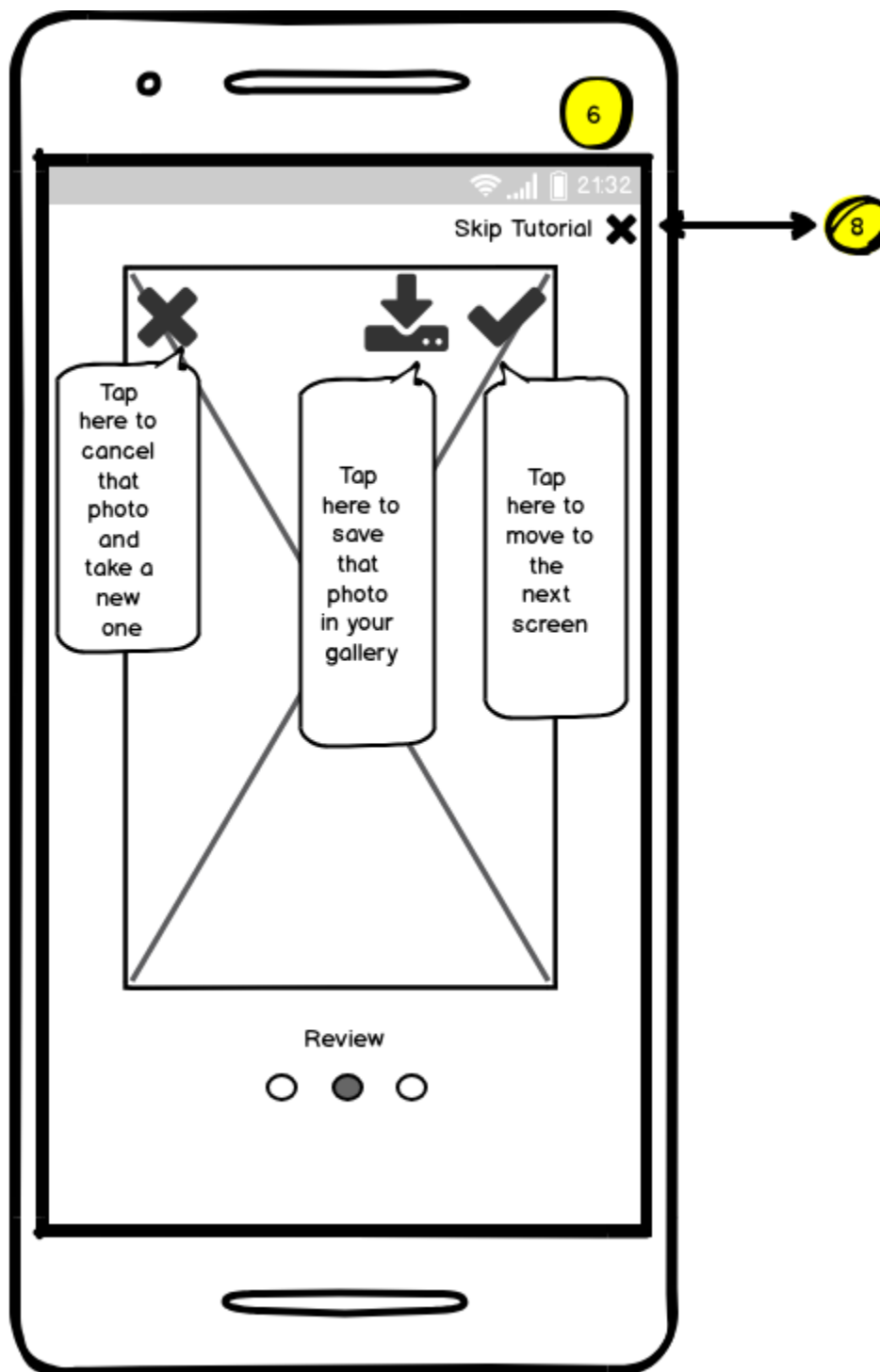


Illustration 25: Home wireframe during first user entrance showing a

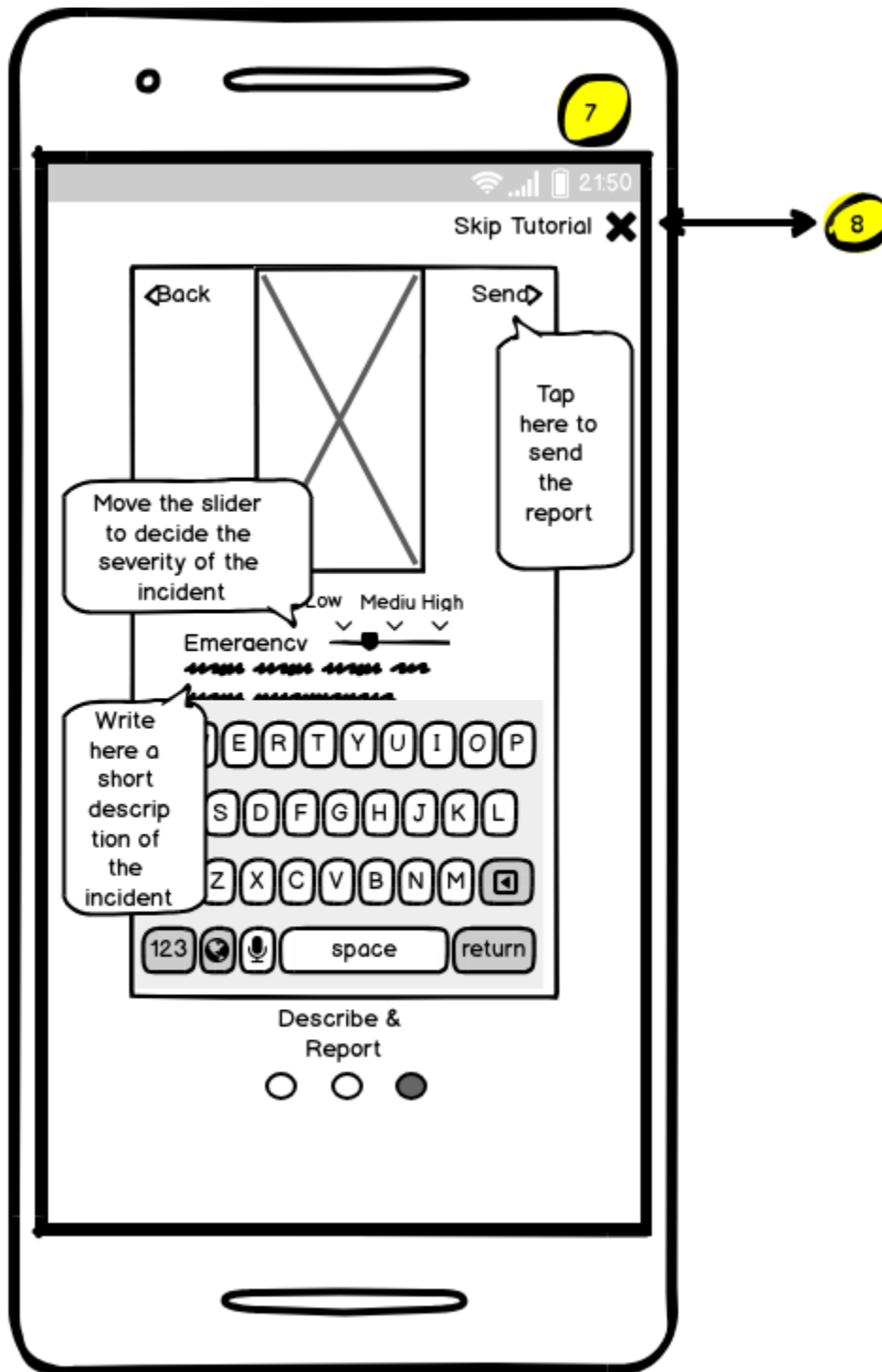


Illustration 26: Home wireframe during first user entrance showing a

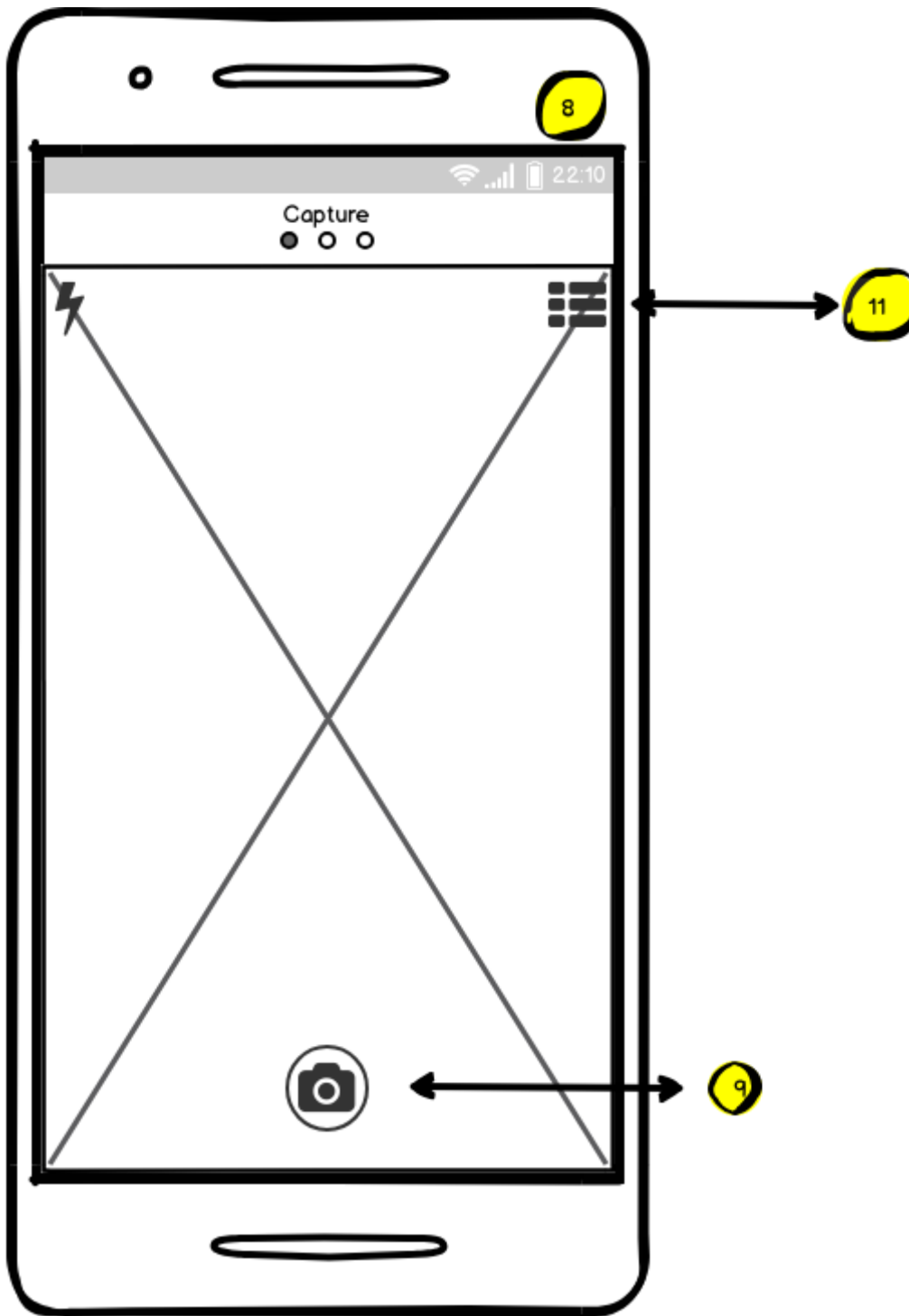


Illustration 27: Home wireframe every time after the first time the user enters

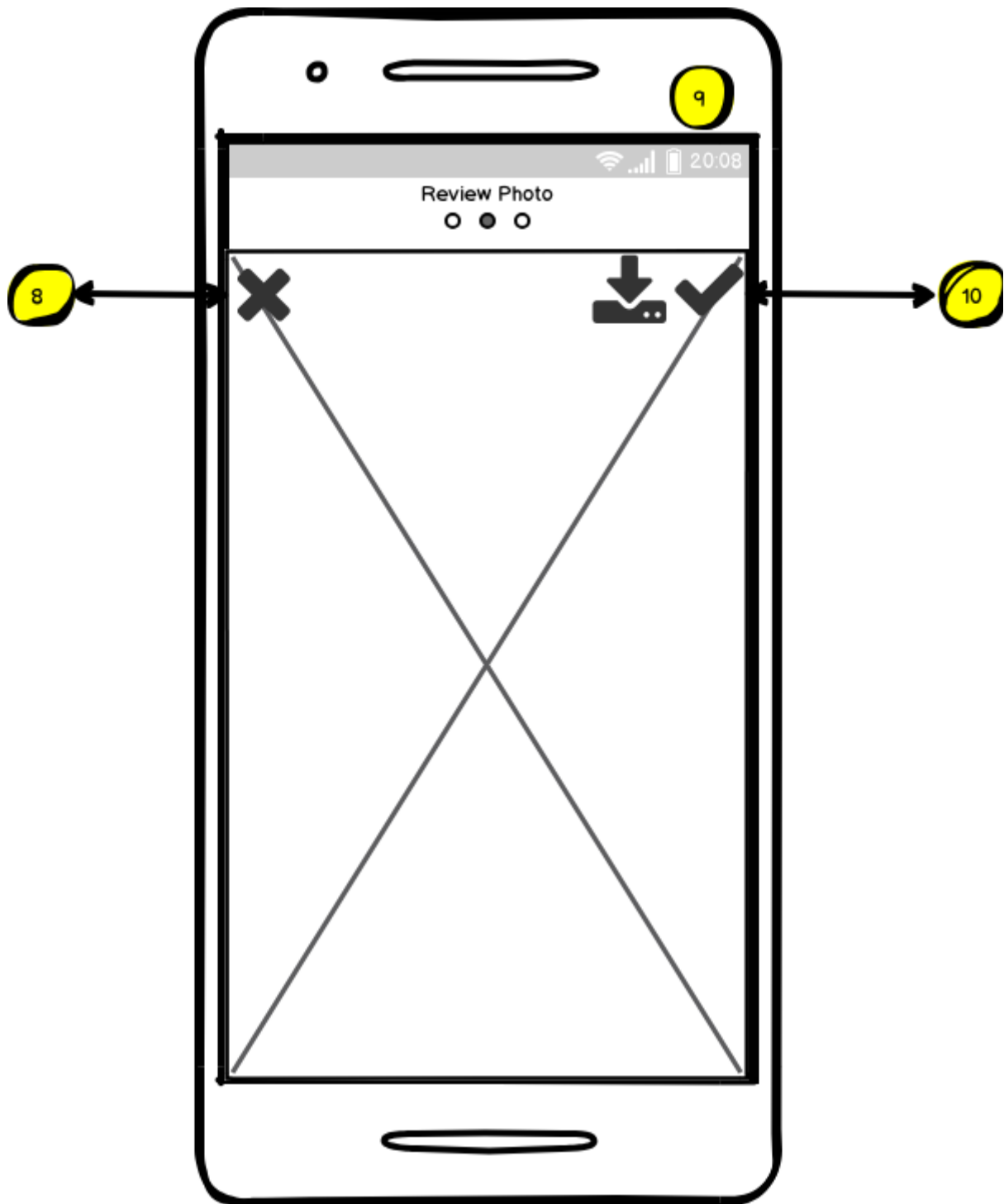


Illustration 28: Incident image preview wireframe

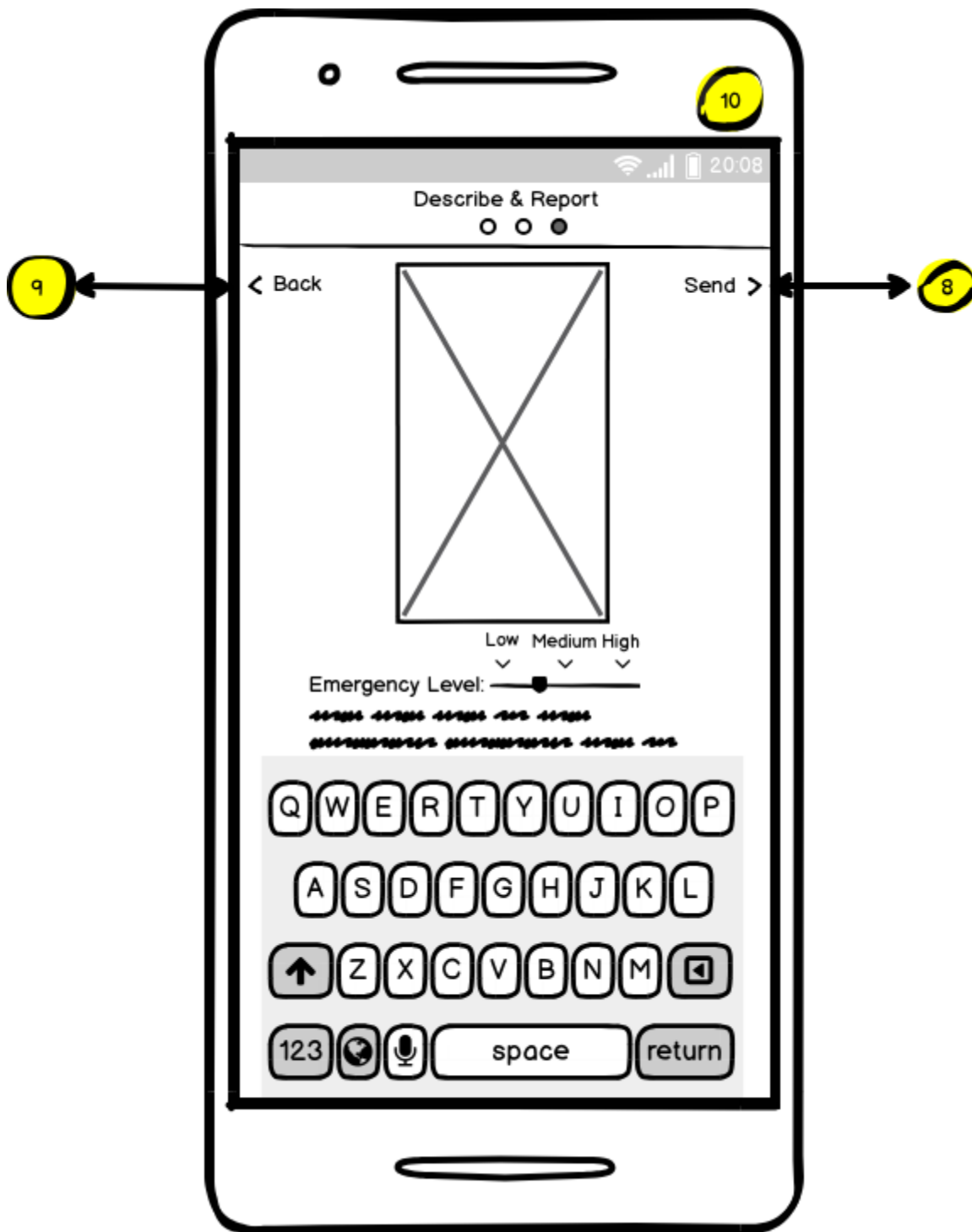


Illustration 29: Create incident report wireframe

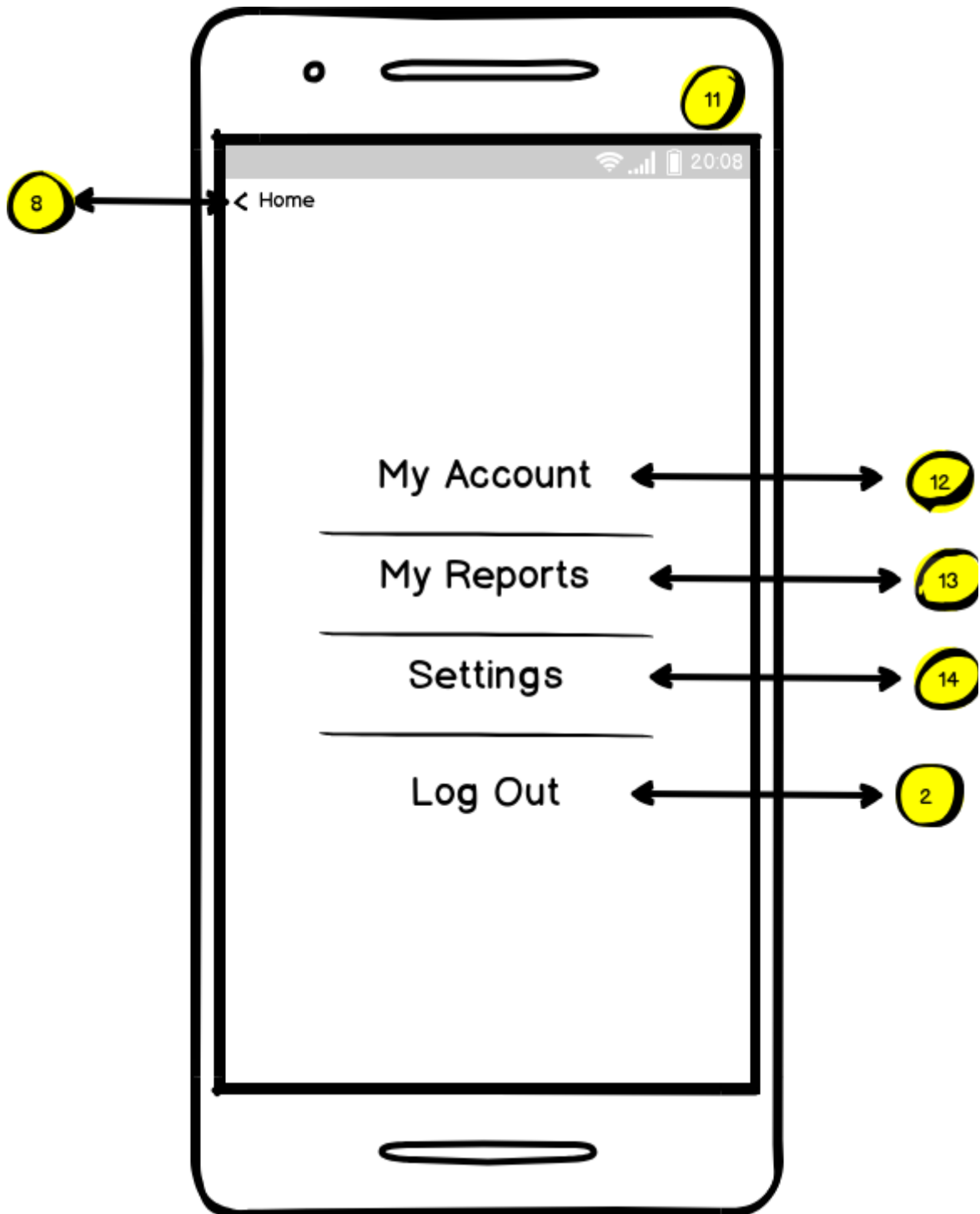


Illustration 30: Menu wireframe; the user can tap either to each of the four available views in

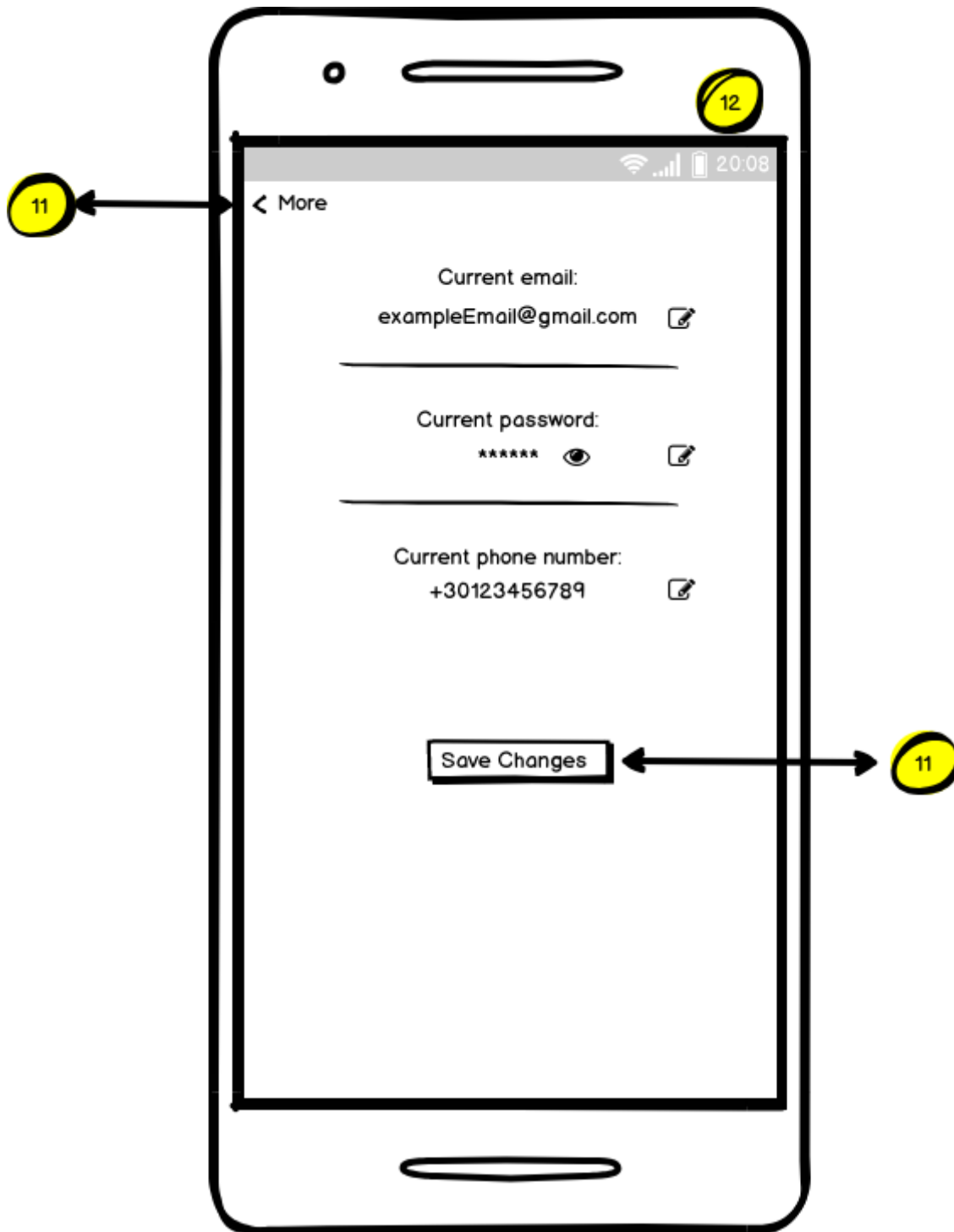


Illustration 31: My account wireframe

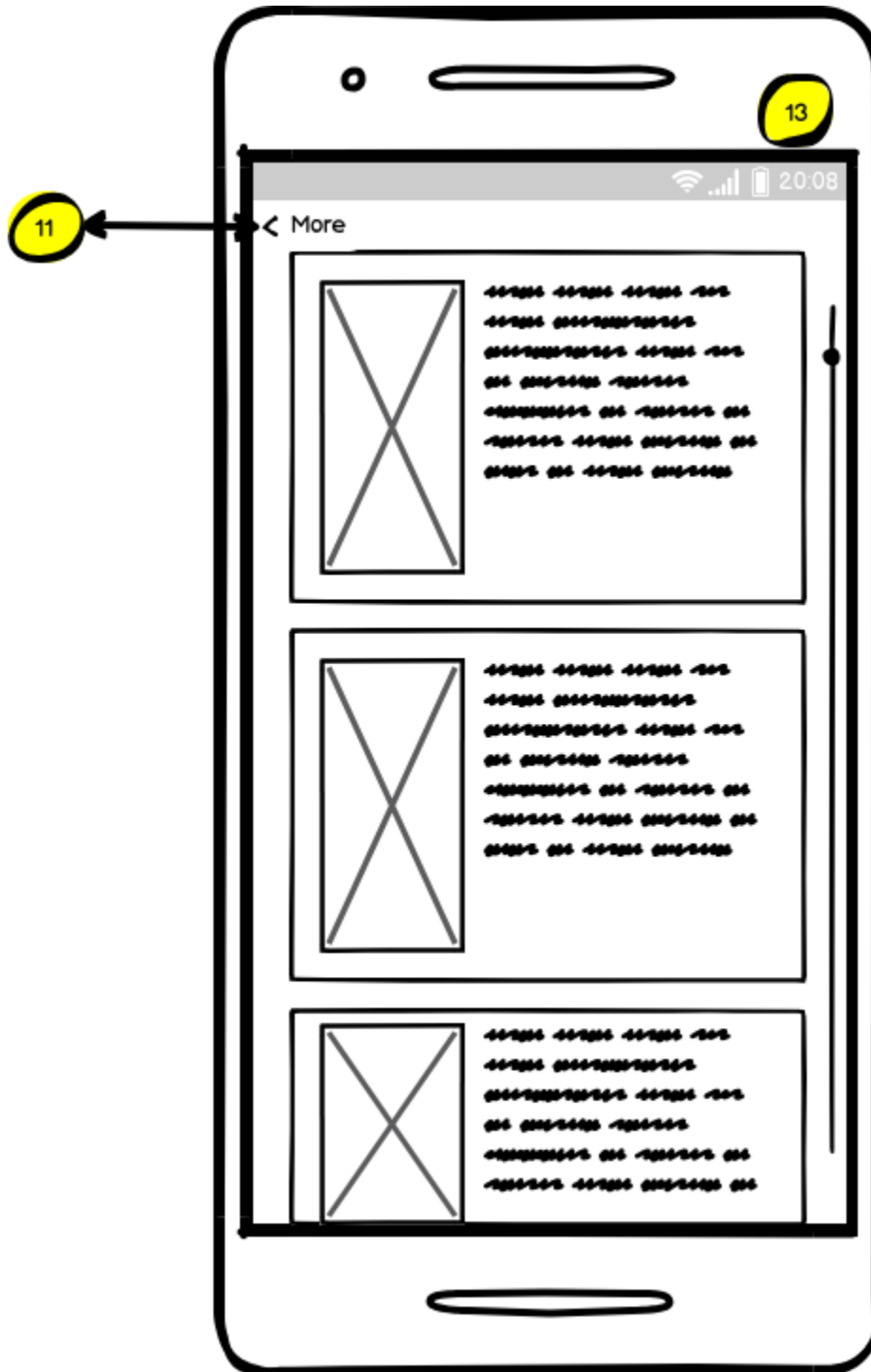


Illustration 32: My reports wireframe

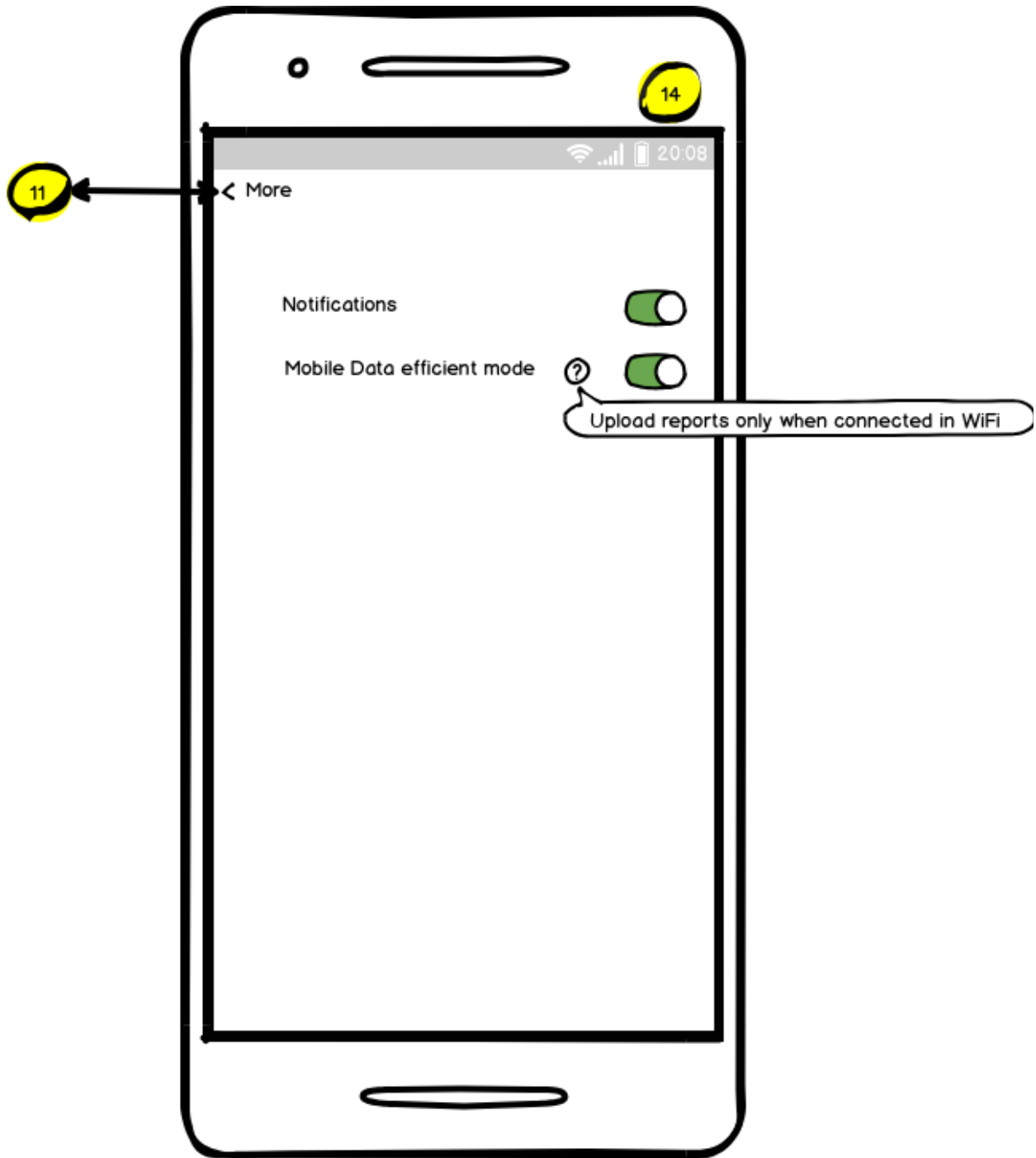


Illustration 33: Settings wireframe

TABLE OF TERMINOLOGY

Ξενόγλωσσος όρος	Ελληνικός Όρος
Android application	Εφαρμογή Android
Framework	Πλαίσιο
Interface	Διεπαφή
Programming languages	Γλώσσες προγραμματισμού
Software architectural style	Στυλ αρχιτεκτονικής προγραμματισμού
Authentication policies	Πολιτικές αυθεντικοποίησης
Server	Εξυπηρετητής
Client	Πελάτης

ABBREVIATIONS - ACRONYMS

NKUA	National Kapodistrian University of Athens
CIR	Citizen Incident Reporting tool
REST	Representational State Transfer
HCI	Human-Computer Interaction
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
MVT	Model-View-Template
ΕΚΠΑ	Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
API	Application Programming Interface
UI	User Interface
XML	Extensible Markup Language
JSON	JavaScript Object Notation
GMS	Google Mobile Services
UCD	User Centered Design
ASCII	American Standard Code for Information Interchange
URI	Uniform Resource Identifier
DTL	Django Template Language

REFERENCES

- [1] Athanasios A. Filippidis “CIR – A Citizen Incident Reporting tool” March 2019; <https://github.com/ThanasisFilippidis/CIR---your-Citizen-Incident-Report-android-application> [Accessed 9/3/2019]
- [2] Neologix “Incident Management Mobile App” <http://www.nlindia.com/incident-management-app.html> [Accessed 26/2/2019]
- [3] emAPPetizer “1st Incident Reporting” <https://play.google.com/store/apps/details?id=com.emAPPetizer.first&hl=en> [Accessed 26/2/2019]
- [4] Ashley Karr Wireframes Defined April 2014; <http://interactions.acm.org/blog/view/wireframes-defined> [Accessed 1/3/2019]
- [5] Amit Ashwini “What is Django and why is it so popular” November 2017 <https://medium.com/swlh/what-is-django-and-why-is-it-so-popular-2b225620cca0> [Accessed 4/3/2019]
- [6] Square Open Source “Picasso” <https://square.github.io/picasso/> [Accessed 6/3/2019]
- [7] Square Open Source “Retrofit” <https://square.github.io/retrofit/> [Accessed 6/3/2019]
- [8] Google “Google Mobile Services” <https://developers.google.com/android/reference/com/google/android/gms/location/package-summary> [Accessed 8/3/2019]
- [9] Google “Camera 2 API” <https://developer.android.com/reference/android/hardware/camera2/package-summary> [Accessed 2/3/2019]
- [10] Interaction Design Foundation “User Centered Design” <https://www.interaction-design.org/literature/topics/user-centered-design> [Accessed 12/3/2019]
- [11] Google “Google Play” <https://play.google.com/store> [Accessed 13/3/2019]
- [12] “ASCII Table and Description” <http://www.asciitable.com/> [Accessed 14/3/2019]
- [13] “Global mobile OS market share in sales to end users from 1st quarter 2009 to 2nd quarter 2018” <https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/> [Accessed 15/3/2019]
- [14] “Report on the Inconvenience of life” <https://play.google.com/store/apps/details?id=kr.go.seoul.seoulSmartReport> [Accessed 15/3/2019]