جامعة الأخوين

ⵜⴰⵙⴷⴰⵡⵉⵜ ⵏ ⵓⵅⴰⵡⴰⵢⵏ

AL AKHAWAYN UNIVERSITY

SCHOOL OF SCIENCE AND ENGINEERING

PECS MESSAGING

A MOBILE APPLICATION (ANDROID)

Capstone Design

November 2015

Benchekroun Youssef

Supervisor: Dr. Naeem Nisar Sheikh

Approved by the Supervisor

ACKNOWLEDGEMENTS

This capstone project would not have been possible without the help of many people. Hereafter, I would like to thank them for their help, support, and/or advice they have given me during the semester.

First, I would like to thank my capstone project supervisor, Dr. Naeem Nisar Sheikh for his guidance, help, and the many discussion sessions that gave me courage and enthusiasm to design and build this application.

Secondly, I would like to thank my family and my friends for their constant support and amazing ideas they provided me with

Finally, I would like to thank Al Akhawayn University for this opportunity to work on this project.

Contents

Table of Figures

Abstract

This capstone project incorporates the well-known Picture Exchange Communication System into a messaging application to help people suffering from the Autistic Spectrum Disorder better communicate with others. This paper presents the requirements analysis, the design, and implementation of a mobile application using the REST technology. This project also serves the purpose of learning new technologies and discovering new concepts. It will serve as an introduction to mobile development using the Android operating system, and to learn more about web services using the REST architecture along with the technologies related to it.

Introduction

People suffering from the Autistic Spectrum Disorder find the most basic everyday tasks very difficult to perform due to their inability to communicate normally with others. Throughout time, many attempts have been conducted to facilitate the learning of a new language and try to help ASD children and grownups to communicate more easily with other people. The most famous system nowadays is called the Picture Exchange Communication System (PECS).

PECS methodology is very simple and clear. Objects are represented as pictures in cards that are used to express the needs and wants of ASD sufferers. This system has been proven to be very efficient in teaching the basics of a language and helping autistic people communicate with others in a social context.

This protocol is mainly divided into six phases of increasing difficulty:

- **Phase I - How to communicate:** This phase is mainly to teach the student the ground rules of PECS. They learn how to pick up cards representing the object they are interested in. For example, a child that desires an apple will learn how to pick up the card representing that object and give it to his teacher.
- **Phase II – Distance and persistence:** During this phase, the student learns how to initiate the conversation when his teacher is not around him.
- **Phase III – Discrimination between symbols:** Throughout this phase, the student learns to make choices between different objects. For instance, a student is asked what his favorite food is when he is given two cards of apples and oranges.
- **Phase IV – Sentence Structure:** This is a crucial step in which the student learns how to formulate useful sentences where he expresses his wants and needs.

- **Phase V – Answering a Direct Question:** During this phase, the student learns how to answer questions like ‘What do you want?’ using sentences like ‘I want candies’.
- **Phase VI – Commenting:** The students are taught to answer more frequently to questions such as ‘what do you see?’ or ‘what is it?’ and so on. They are more and more able to comment on different questions asked of them.

Project Scope & Purpose

The idea of this project is to implement PECS into a mobile application that supports messaging features in order to emphasize the communication aspect of the system.

The project's outcome will be an Android application for smartphones and tablets. Therefore PECS can be easily accessible when the family is traveling or far away from the physical PECS cards. It could be easily implemented in schools that offer programs for ASD children and they can easily access thousands of words in a matter of clicks.

The purpose of this messaging ASD application is to take PECS to a whole new level: anybody anywhere in the world can access this resource to help children and adults suffering from the Autistic Spectrum Condition.

The added value of this project is that it offers the ability to chat and communicate through its messaging features. In fact, the application helps them improve the fundamental communication skills that they are going to need in a real world situation.

Project Plan

This section of the report describes the methodology and the planned time for the completion of each stage.

Methodology

Prior to starting the project, my supervisor and I agreed to adopt the following approach.

First, we proceeded with analyzing the different aspects and information we initially know of the project. This implies to start with a general understanding of PECS and how it is usually used.

The next step was to do a requirements and specifications analysis to determine the functional and mandatory functionalities we want to implement under this project. Then, we determined the constraints and non-functional requirements we wanted to implement in this project. Finally, we also needed to gather the user interface requirements to be more specific about how the GUI of the application will look and feel like. User interface is always important but in our application it is of paramount importance as many sufferers of ASD are known to get easily frustrated with interfaces or functionalities that are not exactly the way they anticipate it or are used to. (National Institute of Mental Health, 2011)

The next step is the design of the application. We decided to split our application into two modules. The first being the web service that will mainly include the implementation of the various functions that constitute the actual application. The second module is the Android client that is mostly user interface interacting with the web service. Consequently, to design the application, we needed to design, as a first step, the web service.

Software Development Methodology

Regarding choosing a software engineering process model, we decided to use the Waterfall Model because it distinguishes and clarifies every step before moving to the next one.

One of the reasons that helped us take this decision is the amount of necessary learning involved in each technology. Therefore, we needed to make sure to have all the necessary information before moving to the next one, so we can spend more time learning and experimenting with the different technologies involved in the phases of the engineering process methodology.

Another reason of why we have chosen the Waterfall model is because we have decided to split the server side from the client side. Thus, using this model allowed us define all the necessary requirements to use in each side ahead of time and the technologies we need.

Project Timeline

Milestones	Week(s)
Finding a supervisor and choosing a capstone project	Week 1, 2
Feasibility study	Week 3
Requirement analysis	Week 4
Design	Week 5, 6, 7
Implementation	Week 8, 9, 10, 11
Final report	Week 12
Final presentation	Week 13

Table 1. Project Timeline

Personal Expectations from this Project

This section describes the primary goals of this capstone project, and their difficulties.

1	*Learning and building an application for Android.*	**MEDIUM**

Objective 1: This capstone project represents a real opportunity to learn the basics of the Android development, therefore, the basics of mobile development. I consider this objective a medium difficulty because it is mostly based on the JAVA programming language which I already have studied in the university. Nevertheless, the Android development introduces many unique libraries and APIs that are interesting to understand and work with.

2	*Learning how to work with web services and take benefits from them.*	**MEDIUM**

Objective 2: Web services are nowadays more and more common in the web, and with the help of this project, I can learn more about them, and deepen my knowledge about the various technologies in this sector. The difficulty is medium because I already have been introduced to web services in the previous semester during my Internet Technologies class. Hence, this is also an opportunity to learn more and apply my already acquired knowledge on the subject.

3	*Learn a new concept about Artificial Intelligence.*	**HIGH**

Objective 3: As far as the application is concerned, the way symbols are selected can be optimized as the number of PECS's images increases. Therefore, doing some research and learning how this problem can be solved imposed itself in my mind. Hence, I decided to learn

new concepts in Artificial Intelligence that might help me solving and optimizing the way of picking a symbol.

4	*Contribute to the outside world, and help solving a problem.*	**LOW**

Objective 4: The capstone semester represents the last semester I have to spend at the university, and the opening to the outside word. It was obvious to make my capstone project interesting by helping solving a real world problem and make it one of my greatest achievements in the near future.

1. Feasibility Study

1.1. Economic Feasibility

We intend to develop and make available this application for free. Thus, the project is economically feasible as the only requirement for a user is a functional smart phone with the Android operating system.

1.2. Existing Solutions

There are multiple applications online that propose teaching how to use the Picture Exchange Communication System; however, there is none on the Google Play Store that proposes messaging feature to allow autistic children to learn and communicate with each other. We give a brief description of the various PECS-related applications we found.

'PexPix' (Android):

This application teaches ASD children to communicate in a chosen language. It provides a large database of images to which the user can record his voice speaking the symbol or use the by default recorded voice in order to teach the child how to speak. In other words, it teaches people suffering from ASD how to communicate with real language by using PECS.

One of the advantages of this application is the ability to record a voice speaking the symbol. This functionality allows the parents/teacher to either use the default language or to record the objects using another language. Therefore, it allows the application to be flexible with any language. However, 'PexPix' does not provide any messaging feature that allows the end user to communicate and exchange messages with other users. It only works offline, and only teaches the basics of PECS. Furthermore, the application is developed only for Android, but not for iOS as well.

'Proyect@ PECS' (Android):

This application helps autistic children to learn how to construct and structure sentences by providing a large amount of PECS symbols. It also categorizes the PECS into different sections to facilitate the selection of pictures and the phrasing of sentences.

The advantage of this application is its ability to provide a clear screen of selection with many categories to choose from. The application was developed around the idea of helping ASD people to formulate sentences; however, it does not provide any means to communicate with others, nor does it provide an iOS version.

'PECS Phase I', 'PECS Phase II','PECS Phase III','PECS Phase IV+' (iOS):

These applications are divided into the phases of the Picture Exchange Communication System. Each application represents a specific phase, and focus on teaching people suffering from ASD each phase of the system. This allows each application to provide a range of functionalities specifically destined to a phase of the system.

The problem with these applications is the fact that they are paid application only available on iOS. This limits dramatically the number of users as there are more Android users nowadays than iOS, and because a user needs to buy each application separately on the Apple App Store. In addition, these applications do not come with messaging features as they are intended only for teaching PECS.

1.3. Relation to analog PECS

In the above survey, we have emphasized the fact that none of the existing PECS-related software applications have message capability. We have done this for the following reason: one of the uses of the analog card-based PECS is to facilitate for an ASD sufferer to be able to communicate with his/her caretakers, family members, and friends. In light of this, it makes

good sense to develop a digital-based PECS to allow the same facilitation through modern communication and information devices.

1.4. Technological Analysis

The application will require a stable internet access as it is a messaging application. In fact, for the same reason that we emphasized a good user interface, namely the tendency of ASD to expect consistency and stability in their environment, it must be emphasized that reliable and fast internet connection will be quite essential for this tool to have relevance for the ASD community.

In order to achieve good performance, the application will be a native app downloadable from the Google Play Store once released.

2. Requirements Analysis

2.1. Functional Requirements

No.	Requirements	Input	Output
1.	The user shall be able to register in the system	Username Password Email	New account
2.	The user shall be able to login through a username and a password	Username Password	Login / Error
3.	The user shall be able to see their profile	-	-
4.	The user shall be able to modify their personal information	-	-
5.	The user shall be able to search for a contact	Search query	Results
6.	The user shall be able to add new contacts	Friend request	Accept / Refuse
7.	The user shall be able to delete a contact	-	Delete contact
8.	The user shall be able to see other contacts' profiles	-	-
9.	The user shall be able to begin a new conversation with a selected contact	-	Messages
10.	The user shall be able to follow a tutorial to learn how to use PECS symbols	-	-
11.	The user shall be able to dismiss the tutorial and start using the messaging app without prior learning of PECS.		
12.	The user shall be able to browse a large variety of PECS symbols and choose the one he wants	-	-
13.	The user shall be able to group many PECS symbols to represent a full sentence.	-	-
14.	The user shall receive normal messages or messages of PECS symbols	-	-

15.	The user shall be able to use PECS's symbols to communicate with a selected contact	-	-
16.	The user shall be able to switch back and forward to a normal keyboard or a PECS keyboard	-	Letter keyboard / PECS
17.	The user shall be able to logout	-	Logout
18.	The user shall be able to request a new password if he ever forgets the actual one	Request new password	New password
19.	The user shall be able to get and see push notifications whenever an event happen	New event	Push notification

Table 2. Functional Requirements

2.2. Non Functional Requirements

No.		Requirements
Product Requirements		
20.	Usability	The user interface should be easy to use and understand
21.	Performance	The system should be quick at responding to the various requests of the user
22.	Space	The mobile application shouldn't take a lot of space to fit the Android criteria.
23.	Reliability	The system should satisfy the users' needs and be accessible 24/7
Organizational Requirements		
24.	Delivery	The project should be delivered with all the progress made at the end of the fall semester.
25.	Implementation	The mobile application will be developed with the Java programming language.

Table 3. Non Functional Requirements

2.3. Interface Requirements

The interface requirements represent a very important aspect of our project, since we are dealing with people suffering from Autistic Spectrum Disorder (ASD). Therefore, we should take into consideration how the application look and feel in order to deliver a quality product to our target users.

No.	Requirements
26.	The login interface should be clear and simple with the name of the application in the top of the screen.
27.	The home screen should have three tabs: one for the friend list, one for the request, and one for the settings.
28.	The messaging interface should contain a history of the last 20 messages sent, and a button to start composing a message.
29.	The symbols should be categorized and displayed in a grid to facilitate the selection process.
30.	The application should provide a simple view to add friends by their emails.
31.	The application should provide different options for the selection: either one click, hold click, or double click. Diversity in this domain will be helpful owing to the varied physical handicaps of ASD sufferers.

Table 4. Interface Requirements

3. Design

3.1. Entity Relational Diagram

Once we have gathered the requirements, we designed the Entity Relational Diagram (ERD) as shown below. It is composed of four tables that represents the entities of our project. In addition, it also illustrates the relationships between tables using the Crow's Foot notation.

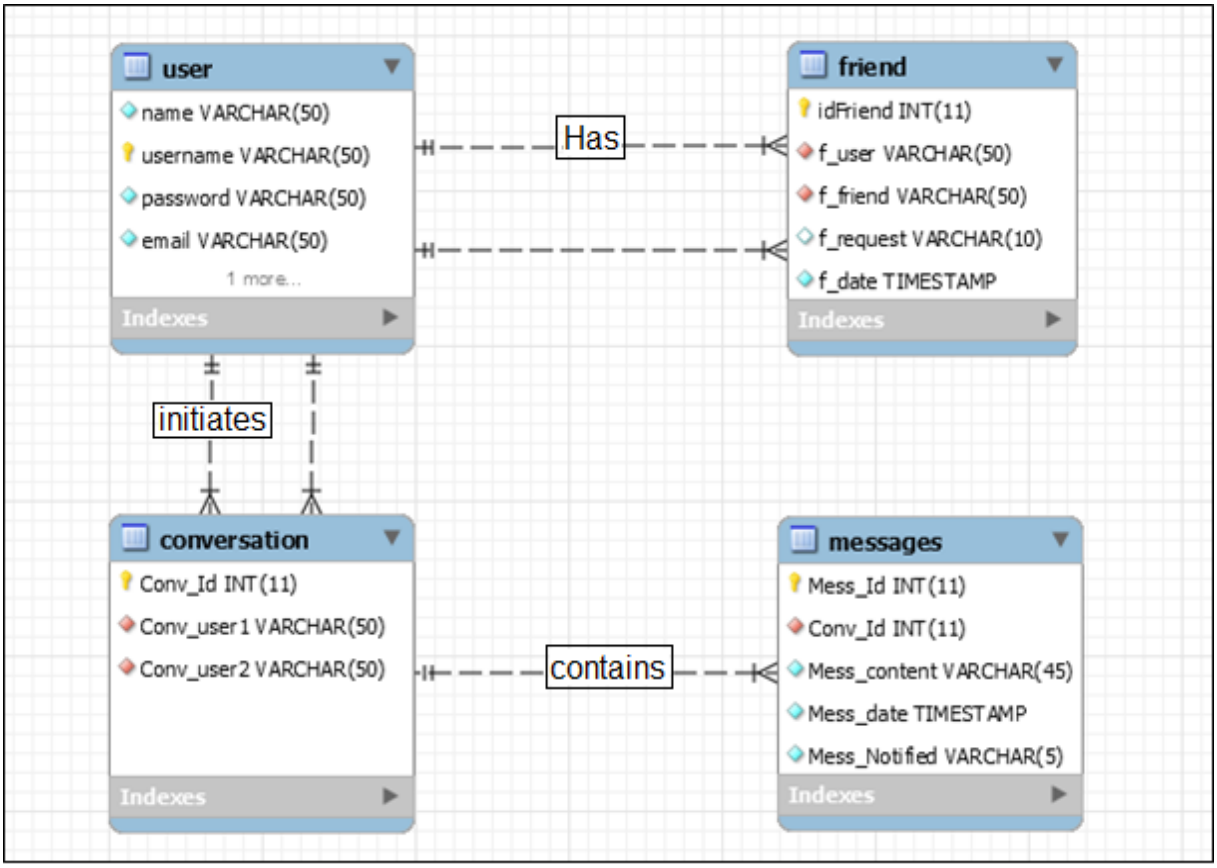

Figure 1. Entity Relational Diagram

3.1.1. Description of the entities

This section describes each entity of the previously shown ERD.

User: the table ‘user’ refers to the users using the applications. Each user in the system is identified by a username and a password and has additional information such as his/her full name and email. The username will allow the system to identify which user is trying to connect and the password will be used to proceed with the connection.

Friend: the table ‘friend’ refers to the list of friends of each user. The table contains a primary key ‘idFriend’ that identifies the friendship, and two foreign keys to the table ‘user’ that represent the friends. The first foreign key indicates the user using the system, and the second foreign key indicates the friend with whom the user is a friend with. Consequently, when a user adds a new friend, the table will add two entries: the first one that indicates with whom you become friend, and another to indicate you as a friend to the other user. This way, it will be much easier to fetch for friends when a user logs in by only selecting the table for his friends. In addition, the table also include a request field which used to send requests to user. If the field is ‘true’ then the friendship exists, if it is ‘false’, then the entry is deleted and the friendship is removed. The friend list is mandatory as it represents the first step to start a conversation.

Conversation: the table ‘conversation’ represents the conversations between two users. It has an identifier, and the two users communicating with each other. This table has a relationship with the table ‘messages’ because a conversation is composed of many messages. This design will help us better organize conversations between users, and I will be much easier fetching messages of a specific conversation since we will only fetch by conversation ID rather than fetching by source and then by destination. Thus, when a user clicks on a contact, a request will be made to the server with the conversation ID. This way the server will only select the

messages of the conversation and sends them back to the user. Afterwards, the client will format the messages and displays them to the user.

Messages: the table 'messages' represents the different messages sent and received. This table refers the table 'conversation', specifies the date at which a message is sent, and has a field to determine if the recipient of the message has received the message. Additionally, this table has a content field that represents the messages itself. . Furthermore, the pictures of the system will be stored locally in the client side, which makes it easier to send and receive messages by simply communicating the IDs. The content of a message contains the IDs of all the symbols sent, so as when the clients receives the messages, it replaces the ID by the actual images before displaying it to the user.

Basically, each picture in the system is defined by a unique global ID that users use as part of their messages. In fact, every time we add a new set of pictures to our system, we give to each picture an ID, and the more symbols we have the larger out collection gets.

Additionally, the server side will only keep the last twenty messages sent rather than keeping the entire conversation. This allowed us to better manage our tables, and quickly retrieve the desired conversations. On the other hand, the client side will locally store the entire conversation should the user needs to check the full history.

Finally, the server will maintain collections of pictures that the client can download at any given time. Initially, the client comes with a predefined set of pictures, but as the client uses the application, he/she can get access to a larger variety of pictures through our server. Furthermore, the pictures can be downloaded in sets that are thematically organized ("fruits", "sports", "outdoor activities", "domestic tasks", etc.), and if a picture is referred and the local client does not have it, an appropriate "don't have this picture" icon will be displayed, along with option to download it from the server.

3.2. Conceptual Architecture

The Conceptual Architecture allows us to draw a general and complete overview of all the processes supported by the application.

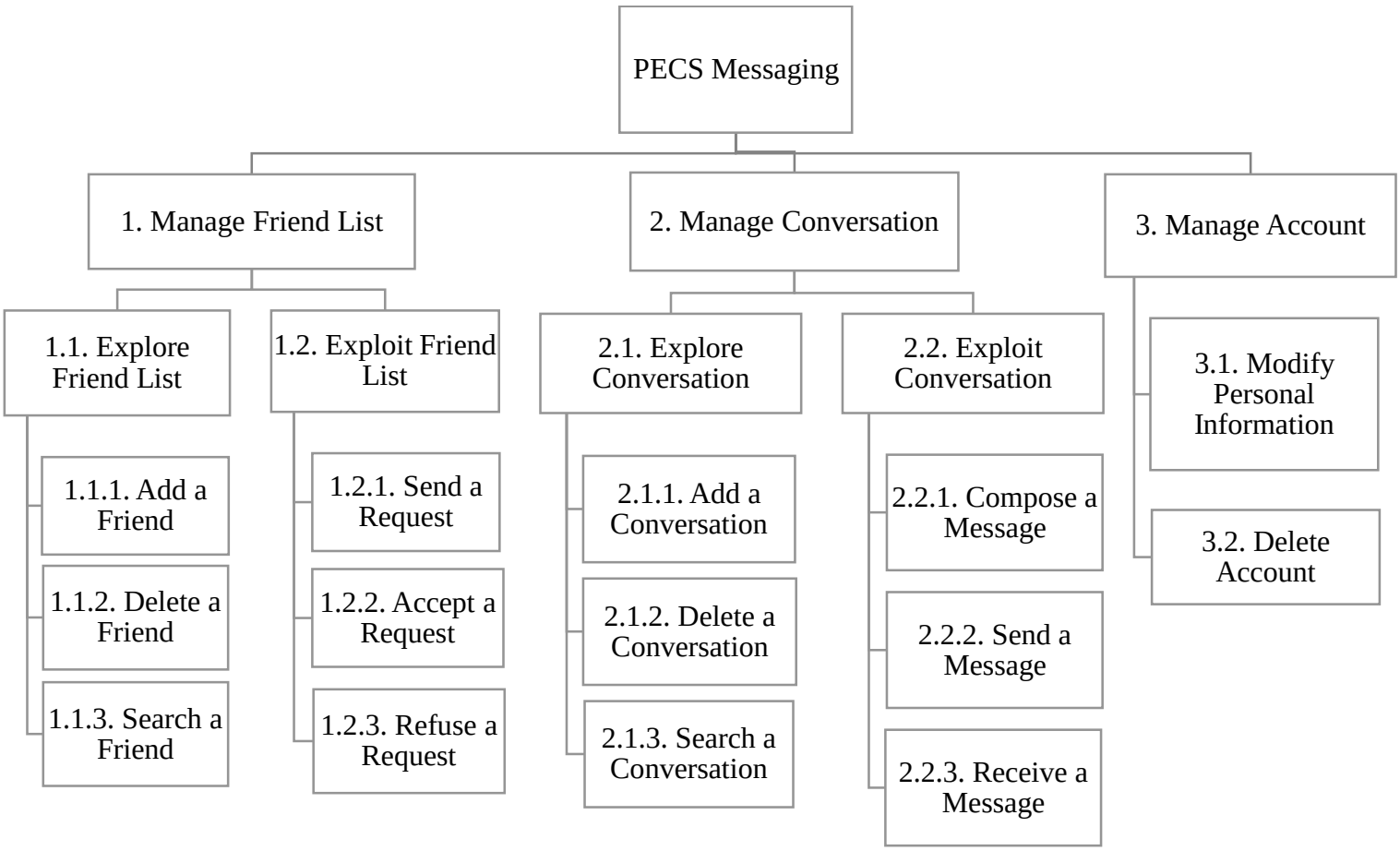

Figure 2. Conceptual Architecture

3.3. Use Case Diagram

The Use Case Diagram represents the interaction between the actors and the system. In our project, we have only considered one actor, which is the end user.

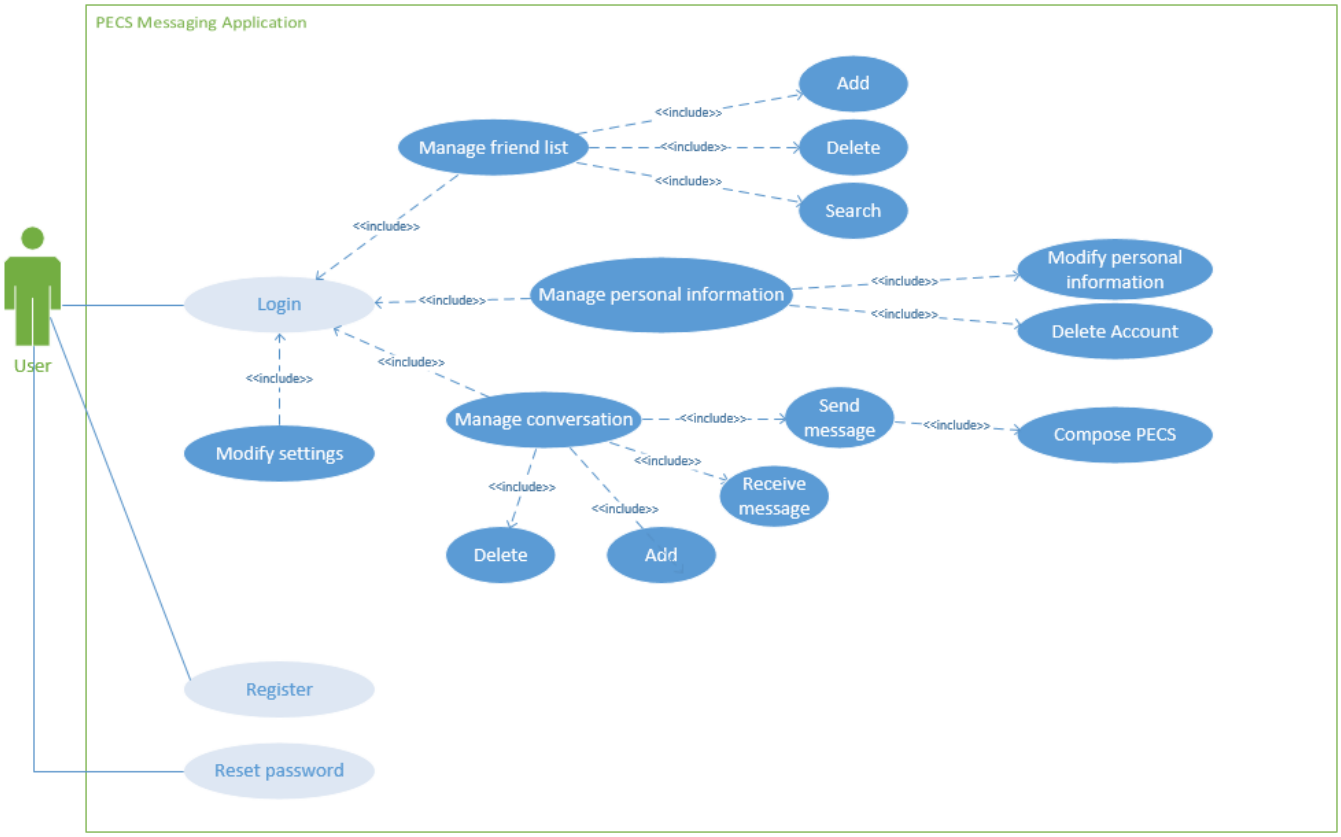

Figure 3. Use Case Diagram

3.3.1. Use-Case Specifications

The user is allowed to perform the actions represented in the following use cases:

3.3.1.1. Login to the application

Use case name	**Login**
Use case ID	01
Use case description	The user will login to the system in order to access the functionalities of the application.
Actors	User
Precondition	A user friendly interface that allows the user to enter his/her username and password.
Main flow	The system will verify if the login credentials.
Post condition	Either the user will be allowed the access to the system or an error message will be displayed.

3.3.1.2. Add friend

Use case name	**Add Friend**
Use case ID	02
Use case description	The user is allowed to add new friends to his friend list.
Actors	User
Precondition	The user should be logged in.
Main flow	The user sends friend request to the added friend.
Post condition	The system should display a success message and redirect the user to his/her friend list.

3.3.1.3. Search friend

Use case name	**Search friend**
Use case ID	03
Use case description	The user is allowed to search friends in the friend list.
Actors	User
Precondition	The user should be logged in. The user should have at least one friend.
Main flow	The system executes a search method on the friend list.
Post condition	The system should display the desired friend, or an empty result if the friend does not exist.

3.3.1.4. Delete friend

Use case name	**Delete friend**
Use case ID	04
Use case description	The user is allowed to delete friends from in the friend list.
Actors	User
Precondition	The user should be logged in. The user should have at least one friend.
Main flow	The system deletes the friendship entry in the database.
Post condition	The system refreshes the friend list.

3.3.1.5. Modify personal information

Use case name	**Modify personal information**
Use case ID	05
Use case description	The user is allowed to modify his own personal information.
Actors	User
Precondition	The user should be logged in.
Main flow	The system updates the information.
Post condition	The system should display a success message and refresh the screen.

3.3.1.6. Delete account

Use case name	**Delete account**
Use case ID	06
Use case description	The user is allowed to delete his account.
Actors	User
Precondition	The user should be logged in.
Main flow	The system deletes the user data from the database, and deletes the account.
Post condition	The system is redirected to the login activity.

3.3.1.7. Add conversation

Use case name	**Add conversation**
Use case ID	07
Use case description	The user is allowed to initiate a conversation with a friend in the friend list.
Actors	User
Precondition	The user should be logged in. The user should have at least one friend.
Main flow	The system creates a new conversation in the database.
Post condition	The system is redirected to the messaging interface.

3.3.1.8. Compose message

Use case name	**Compose message**
Use case ID	08
Use case description	The user is allowed to compose a PECS message.
Actors	User
Precondition	The user should be logged in. The user should have at least one conversation. The system should contain at least one PECS.
Main flow	The application formats the message and enters it in the database.
Post condition	The message is displayed in the screen.

3.3.1.9. Receive a message

Use case name	**Receive a message**
Use case ID	09
Use case description	The user can receive new messages.
Actors	User
Precondition	The user should be logged in. The user should have at least one conversation.
Main flow	The system fetches the database for new messages. If any, the system marks them as notified.
Post condition	The system displays the message.

3.3.1.10. Delete conversation

Use case name	**Delete conversation**
Use case ID	10
Use case description	The user is allowed to delete conversations.
Actors	User
Precondition	The user should be logged in. The user should have at least one conversation.
Main flow	The system deletes the conversation with all its messages from the database.
Post condition	The system redirects the user to his friend list.

3.3.1.11. Modify settings

Use case name	**Modify settings**
Use case ID	11
Use case description	The user is allowed to modify the settings.
Actors	User
Precondition	The user should be logged in.
Main flow	The application is updated with the new settings.
Post condition	The settings are activated and saved.

3.3.1.12. Register

Use case name	**Register an account**
Use case ID	12
Use case description	The user can create an account.
Actors	User
Precondition	Registration interface.
Main flow	The system registers the new user in the database.
Post condition	The system is redirected to the login interface.

3.3.1.13. Reset the password

Use case name	**Reset the password**
Use case ID	13
Use case description	The user is allowed to ask for a new password.
Actors	User
Precondition	Reset password interface. A valid email address.
Main flow	The system reset the password and sends it to the user email address.
Post condition	The system is redirected to the login activity.

3.4. Sequence Diagram

The sequence diagram illustrates the process and the sequence associated with a specific action. It helps us visualize the data flow and how the action is carried out through the different methods and functionalities of our application.

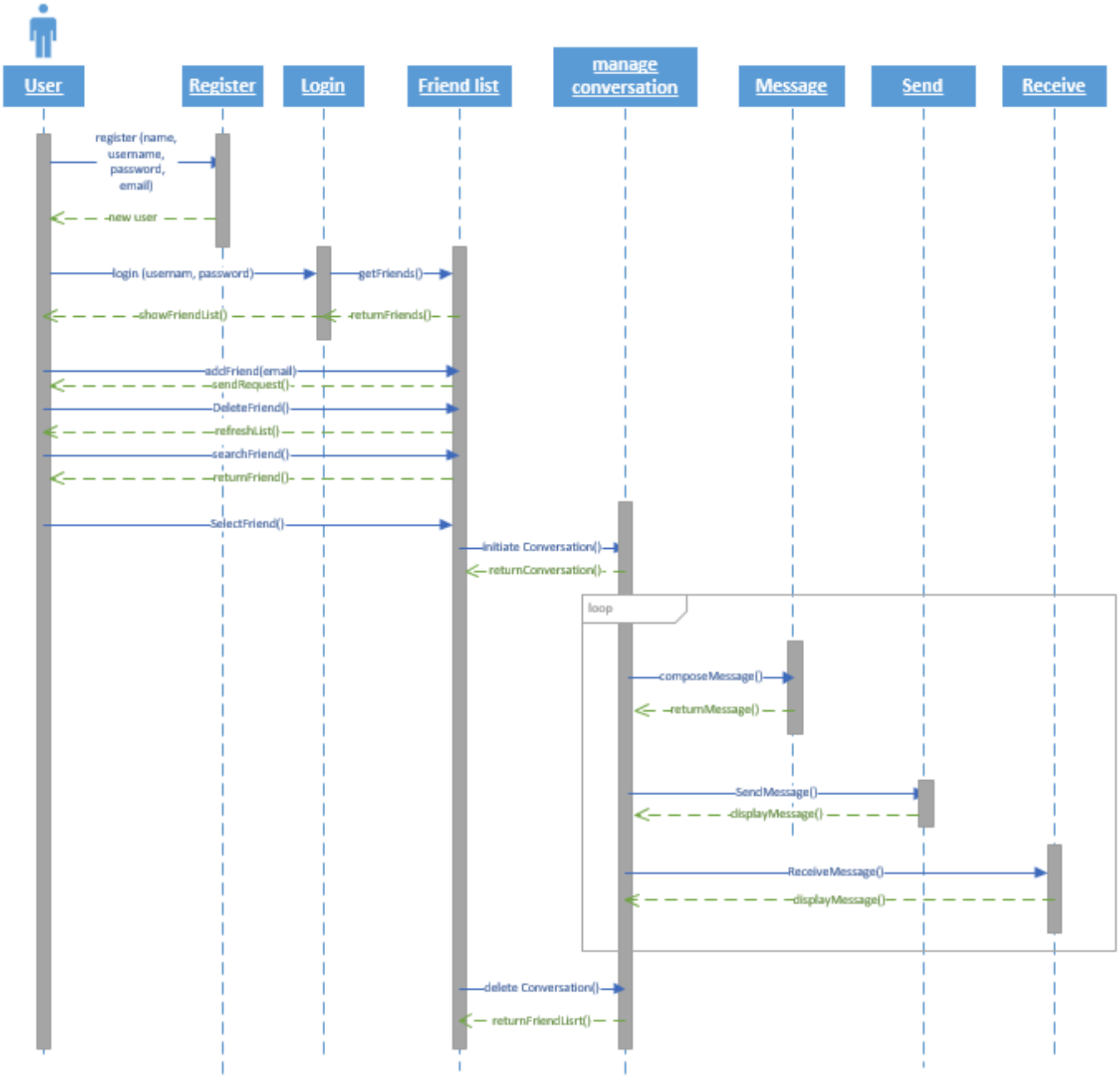

Figure 4. Sequence Diagram

3.5. System Architecture

3.5.1. MVC Architecture

When we talk about web services and web technologies, we usually refer to the MVC architecture. It is a very popular architecture that distinguishes three main roles of a project: the Model, the View, and the Controller.

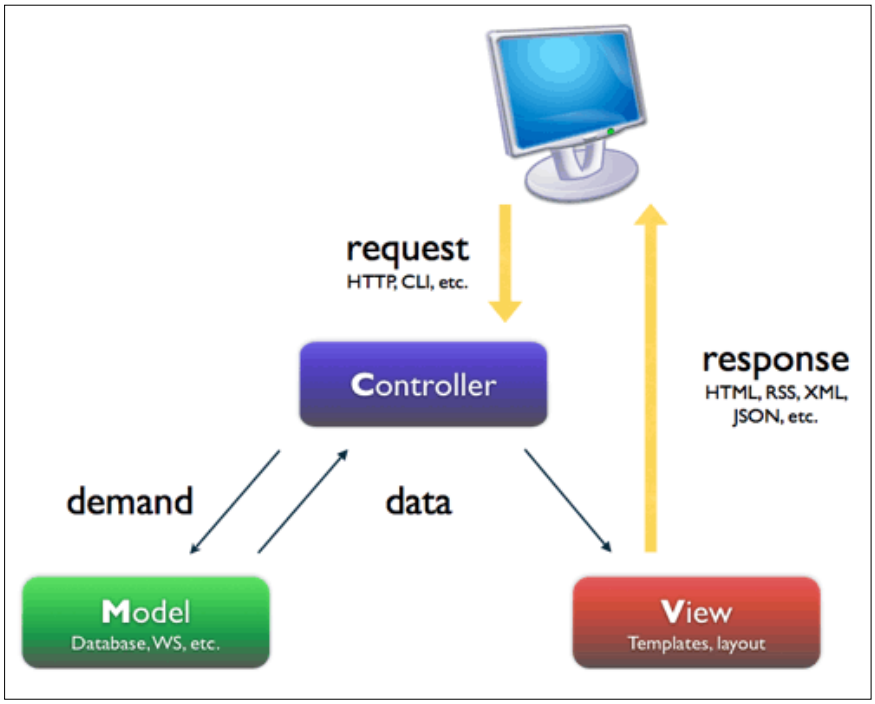

Figure 5. MVC Model

The Model represents the logic of the program. It is where all the calculations are made and all the results are generated. Its primary role is to satisfy a request and come up with an appropriate result as a piece of information to send back to the user.

The View allows the display of the information to the user. A graphical user interface that format the results and information into a nice interface to the user. In addition, the View is also the place where new requests are created. For instance: a web page.

The Controller is the middle man. It takes the request from the View and send them to the Model. Once the Model processes the request and come up with a result, the Controller take the generated piece of information and send it back to the View. The View then takes the information, formats it and displays it to the user.

3.5.2. System Architecture

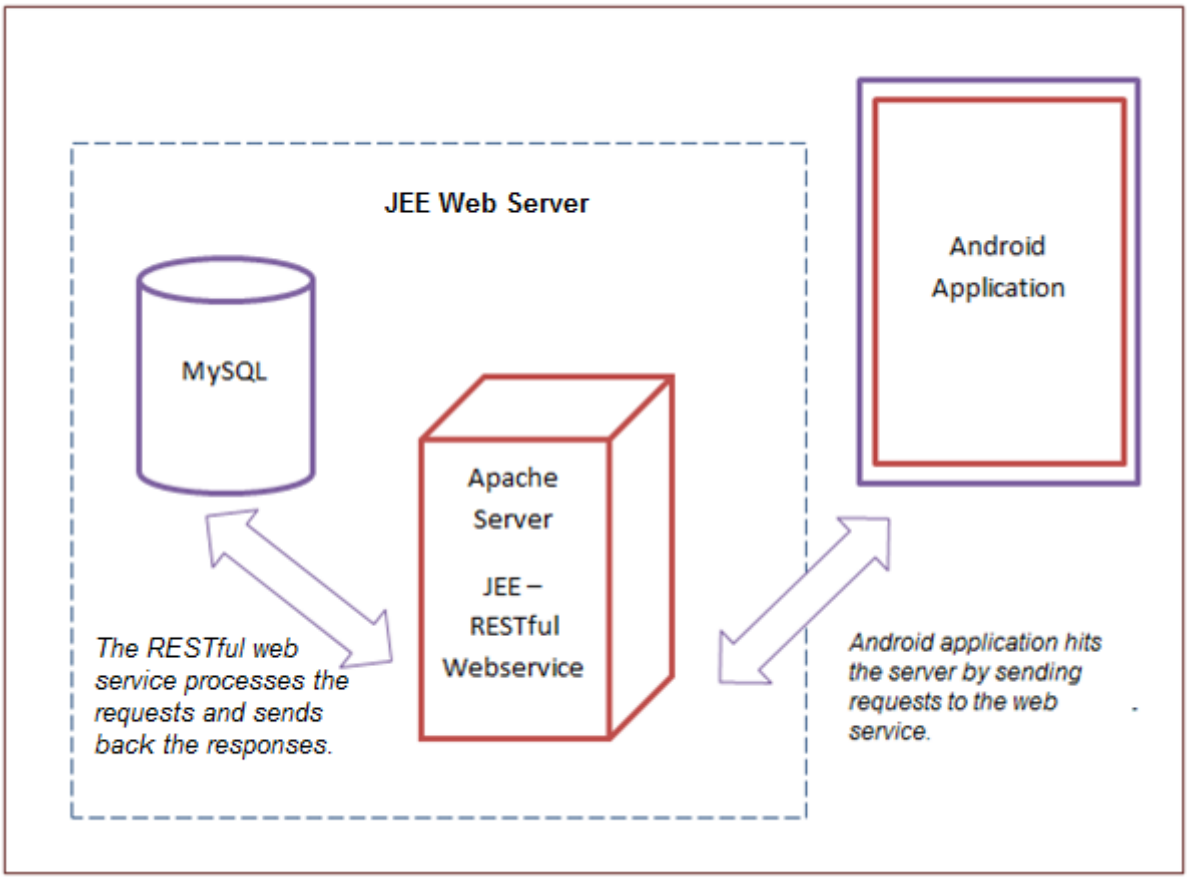

Figure 6. System Architecture

As we mentioned before, the architecture we have chosen is to have the server side module and the client side module.

The server side is the web server implementing all the functionalities of our project. It implements a RESTful web services that will be consumed by REST clients. The web service

implements the models and the interaction with the databases. The models are where the data is treated and processes whereas the database is where the data is stored.

The client side is the Android application. The client deals with the graphical user interface and how data is formatted to the end user. It also includes the controller that sends requests to the web server and receives the corresponding responses. For the purpose of this project, we only have to build an Android application, but in fact, the client could be any device meant to consume a RESTful web service, such as a browser.

3.5.3. What is REST?

According to Oracle's definition, Representational State Transfer (REST) is "is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induces desirable properties, such as performance, scalability, and modifiability that enable services to work best on the Web". This means that REST deals with data as resources, and therefore, they are accessed using simple URLs. In addition, a RESTful web service uses only the HTTP protocol and therefore allows the user to format the requests and the responses using any format they feel comfortable with. This can vary from HTML, XML, JSON, etc.

In our situation, we are going to use simple URLs to access our web service through the HTTP protocol. Once the request is processed, we will use the JSON format to send back the response. The client will then parse the file, format the response, and displays it to the user.

3.5.4. What is JSON?

The JavaScript Object Notation (JSON) is a human readable text in which data objects are represented as a pair of attributes and their values. It is mostly used with asynchronous communication between a client and a server. Here the client can be a browser, a stand-alone

application, or like in our case, an Android application. Usually, applications uses JSON as a better alternative to the classical way of using Extensible Markup Language (XML).

3.5.5. JAX-RS

JAX-RS is an Application Programming Interface (API) created specifically to develop web services that takes advantage of the REST architecture. It is considered a powerful API because it takes advantage of the Java annotation system to decorate and indicate the needed resources and the actions to perform on them.

3.5.6. Jersey Framework

Many frameworks take advantage of the JAX-RS API, and the Jersey Framework is one of them. It mainly provides support from the API and implements the necessary classes and methods that are required for developing RESTful web services. In our application, we will use the Jersey framework to implement our web service and the different functionalities the application offers.

3.6. REST vs SOAP

This section presents a small comparison of the two types of web services: REST and SOAP (Simple Object Access Protocol).

#	SOAP	REST
1	A XML-based message protocol	An architectural style protocol
2	Uses WSDL for communication between consumer and provider	Uses XML or JSON to send and receive data
3	Invokes services by calling RPC method	Simply calls services via URL path
4	Does not return human readable result	Result is readable which is just plain XML or JSON
5	Transfer is over HTTP. Also uses other protocols such as SMTP, FTP, etc.	Transfer is over HTTP only
6	JavaScript can call SOAP, but it is difficult to implement	Easy to call from JavaScript
7	Performance is not great compared to REST	Performance is much better compared to SOAP - less CPU intensive, leaner code etc.

Table 5. SOAP vs REST

Both technologies are worth knowing and studying. One should know their strengths and weaknesses. For my case, I am already familiar with the SOAP type of web services; therefore, it was an obvious choice when I first started this project. For this reason, I preferred to choose REST to learn more about it and be able to work with it should I ever need to.

Another reason of why I chose REST is because of its easiness of learning compared to the SOAP technology. It mainly uses simple calls to the service via URL which makes a huge difference. This combined with the ability to choose whether to work with JSON or XML to send and receive data made it very convincing as a choice.

3.7. Technology Enablers

During the development of the application, we will be using two primary integrated development environments (IDEs):

- Eclipse JEE edition: this IDE will be used to develop the server side of the project.
- Android Studio: This IDE will be used to develop the client side, which is an Android client application.

SERVER SIDE

- Apache Server
- JAX-RS API
- Jersey Framework
- MySQL

In addition, we will use the Google Chrome extension 'Advance REST Client Application' to test the different functionalities of the web server.

CLIENT SIDE (Android)

- Android API for Android Studio
- Android Virtual Machine
- Android Async HTTP Handler library
- ShowcaseView library

The library 'Android Async HTTP Handler' will allow us to send HTTP requests asynchronously, and similarly receive responses from the web server asynchronously.

The library ShowcasView will allow us create tutorials for when the application is installed for the first time in a device, we can help the user and guide him/her into using the

application. This way we can showcase all the functionalities of the application, so the user becomes aware of them and knows how to use them.

4. Web Service Development

4.1. Using Jersey REST framework

In this section, we will demonstrate how we used the Jersey REST framework to build our web service.

The Framework allows us to associate each resource with a specific URL path using Java annotation. Therefore, when a user desires to access the said resource, he/she only needs to use the related URL.

- `@Path` annotation is responsible for assigning a path to a resource.

Similarly, we can assign the access method (GET, POST, PUT, and DELETE) to resources to specifically indicate that the resource is only accessible through the indicated HTTP method.

- `@GET, @POST, @PUT, @DELETE` annotations are responsible for defining how a resource can be accessed.

The framework also provides a simple java annotation to indicate which type of text file to use for the output. Therefore, we can easily chose to work with JSON or another type. For our project, we decided to work with JSON as it will be easy to parse and work with both in the server side and in the client side of the application.

- `@Produces(MediType.<yourType>)` annotation is responsible for selecting a media type as JSON where the output will be stored.

Since a RESTful web services is invoked by using simple URL, the Jersey Framework implements yet another easy annotation to specify parameters that are needed to some methods.

- `@QueryParam(<nameOfParam>)` annotaion is specifically used when there is a need to specify a parameter in the URL path of the desired resource.

Login.java

```
//Path: http://localhost/<appln-folder-name>/login
@Path("/login")
public class Login {
    // HTTP Get Method
    @GET
    // Path: http://localhost/<appln-folder-name>/login/dologin
    @Path("/dologin")
    // Produces JSON as response
    @Produces(MediaType.APPLICATION_JSON)
    // Query parameters are parameters: http://localhost/PECS/login/dologin?username=abc&password=xyz
    public String doLogin(@QueryParam("username") String uname, @QueryParam("password") String pwd){
        String response = "";
        if(checkCredentials(uname, pwd)){
            //response = Utility.constructJSON("login",true);
            JSONObject obj = new JSONObject();
            try {
                obj.put("tag", "login");
                obj.put("status", new Boolean("true"));
                obj.put("name", dbLogin.user.getName());
                obj.put("username", dbLogin.user.getUsername());
                obj.put("password", dbLogin.user.getPassword());
                obj.put("email", dbLogin.user.getEmail());

                response = obj.toString();
            } catch (JSONException e) {
                // TODO Auto-generated catch block
            }
        }else{
            response = Utility.constructJSON("login", false, "Incorrect Email or Password");
        }
    return response;
    }
```

Figure 7. Login web service

Register.java

```
//Path: http://localhost/<appln-folder-name>/register
@Path("/register")
public class Register {
    // HTTP Get Method
    @GET
    // Path: http://localhost/<appln-folder-name>/register/doregister
    @Path("/doregister")
    // Produces JSON as response
    @Produces(MediaType.APPLICATION_JSON)
    // Query parameters are parameters: http://localhost/<appln-folder-name>/register/doregister?name=pqrs&username=
    public String doLogin(@QueryParam("name") String name, @QueryParam("username") String uname, @QueryParam("passw
        String response = "";
        //System.out.println("Inside doLogin "+uname+"  "+pwd);
        int retCode = registerUser(name, uname, pwd, email);
        if(retCode == 0){
            response = Utility.constructJSON("register",true);
        }else if(retCode == 1){
            response = Utility.constructJSON("register",false, "You are already registered");
        }else if(retCode == 2){
            response = Utility.constructJSON("register",false, "Special Characters are not allowed in Username and
        }else if(retCode == 3){
            response = Utility.constructJSON("register",false, "Error occured");
        }
        return response;

    }
```

Figure 8. Register web service

Send.java

```
@GET
@Path("/dosend")
@Produces(MediaType.APPLICATION_JSON)
public String dosend(@QueryParam("M_Source") String source,
        @QueryParam("M_Destination") String destination,
        @QueryParam("M_Content") String content ){

    String response = "";
    if(checkMessage(source, destination, content)){
        response = Utility.constructJSON("Message_sent", true);
    }else{
        response = Utility.constructJSON("Message_sent", false, "Incorrect Source or Destination entered");
    }
    return response;
}
```

Figure 9. Send Web Service

Receive.java

```
@GET
@Path("/doreceive")
@Produces(MediaType.APPLICATION_JSON)
//the username referes to the destination user receiving the message
public String doreceive (@QueryParam("username") String username){
    String response =" ";
    if(fetchNewMessage(username)){
        //response = Utility.constructJSON("login",true);
        JSONObject obj = new JSONObject();
        try {
            obj.put("tag", "New Message");
            obj.put("status", new Boolean("true"));
            obj.put("id", dbReceive.message.getID_Message());
            obj.put("source", dbReceive.message.getM_Source());
            obj.put("destiation", dbReceive.message.getM_Destination());
            obj.put("content", dbReceive.message.getM_content());
            obj.put("date", dbReceive.message.getM_date());

            response = obj.toString();
        } catch (JSONException e) {
            // TODO Auto-generated catch block
        }
    }else{
        response = Utility.constructJSON("New Message", false, "No new message");
    }
    return response;
}
```

Figure 10. Receive web service

4.2. Testing with Advance REST Client Application

Before we can move to the client development, we need to test the different methods of our web service. Thankfully, there exists a Google Chrome extension by the name of 'Advanced REST Client Application' that allows us to send requests and receive the corresponding responses.

The client allows us to specify the path, the parameters if there are any, and even the HTTP method. Once the request is sent, it will display to the screen the full HTTP request and the response alongside with the output media if it is specified in the web service.

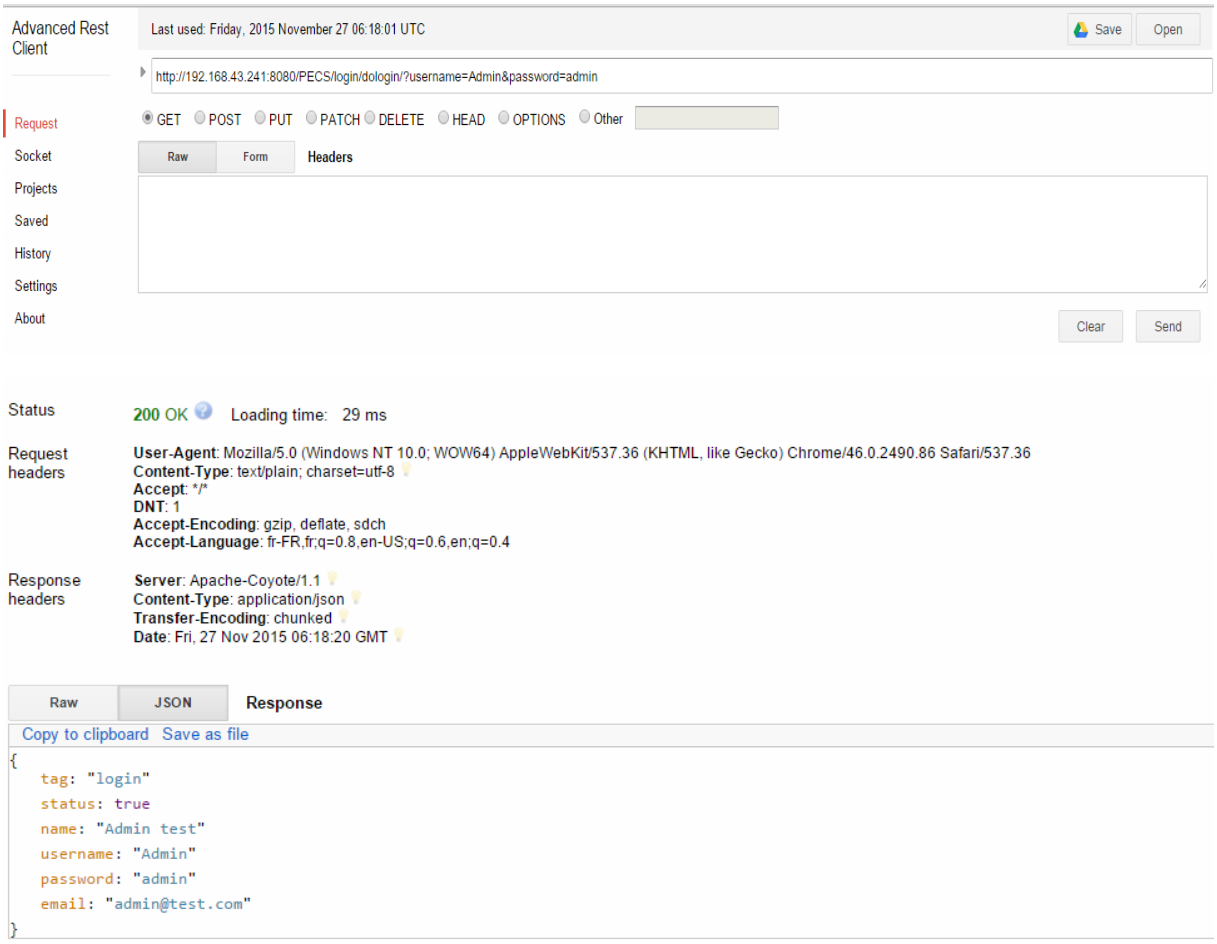

Figure 11. Advanced REST client app

Using this testing methodology, we could test all the functionalities of our web server before developing the Android client.

5. Android Client Development

Before we start the development of the Android client, we need to define an important concept in Android architecture which is the Activity Manager. In a typical programming language, the first function to execute is usually the main; however, the Android platform uses a different way to initiate a program: Activities.

An Activity is a component meant for interaction with the user. It can have its proper user interface as the login Activity, or it can work with no interface, like a radio activity. Each activity within an application has specific actions associated with it. For instance, a login activity will have a nice graphical user interface with textboxes, so the user can enter his username and password. The action in this case is to authenticate the user; therefore, once logged in the user is redirected to another activity that also have its proper actions.

In every Android application, there exists one special Activity which is often called the Main Activity. This is the first Activity to launch when we start the application.

As explained earlier, an activity can have its graphical user interface, and this interface is just a simple XML file containing elements of the graphics, such as buttons and textboxes.

The following is the XML file of the login plus the activity:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/bg_login"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="10dp" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:orientation="vertical"
        android:paddingLeft="20dp"
        android:paddingRight="20dp" >

        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="10dip"
            android:text="PECS Messenger"
            android:gravity="center"
            android:textSize="30sp"
            android:textColor="@color/white"/>

        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            />
```

Figure 12. Login XML

We can see that the login xml file is composed of graphical elements, such as scroll views, linear layouts, and text views.

Each element can have many properties that goes from resizing it or completely changing its graphics, such as position, size, height, width, etc.

```java
public class LoginActivity extends Activity {

    // Progress Dialog Object
    ProgressDialog prgDialog;
    // Error Msg TextView Object
    TextView errorMsg;
    // Email Edit View Object
    EditText usernamET;
    // Passwprd Edit View Object
    EditText pwdET;

    UserLocalStore userLocalStore;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        // Find Error Msg Text View control by ID
        errorMsg = (TextView)findViewById(R.id.login_error);
        // Find Email Edit View control by ID
        usernamET = (EditText)findViewById(R.id.loginUsername);
        // Find Password Edit View control by ID
        pwdET = (EditText)findViewById(R.id.loginPassword);
        // Instantiate Progress Dialog object
        prgDialog = new ProgressDialog(this);
        // Set Progress Dialog Text
        prgDialog.setMessage("Please wait...");
        // Set Cancelable as False
        prgDialog.setCancelable(false);

        userLocalStore = new UserLocalStore(this);
    }
```

Figure 13. Login activity

In order for a java class to become an activity, it must extend the Activity Manager. There exists many types of Activities -- some are more adapted to specific scenarios like having a list or a grid view. For our case, we only work with the normal Activity.

The first important thing to do when creating an activity is to override the function `onCreate()` because it is the first method to execute when an Activity starts. In this method, we can initialize the different GUI elements, so we can work with them later on. In addition, we can also call other methods should we ever need so. This method is where the Activity

start and therefore it is also the right place to initialize any variables or graphical elements if we ever intend to use them later.

5.1. Android Async HTTP Handler

The Android Asynchronous HTTP Handler allows us to send asynchronous requests to a web server and of course receive asynchronous responses. For the purpose of our project, and since we are building a RESTful web service that can be consumed by a REST client, we have to use asynchronous handling since it takes advantage of the REST architecture.

In order to send a request, we need first to gather the parameters that are required by the web service method we are trying to call from our client.

In order to do so, the Android Asynch HTTP Handler allows us to create an object of type `RequestParams` in which we put the parameters we need, as shown below.

```
public void loginUser(View view){
    // Get Email Edit View Value
    String username = usernamET.getText().toString();
    // Get Password Edit View Value
    String password = pwdET.getText().toString();
    // Instantiate Http Request Param Object
    RequestParams params = new RequestParams();
    // When Email Edit View and Password Edit View have values other than Null
    if(Utility.isNotNull(username) && Utility.isNotNull(password)){
        // Put Http parameter username
        params.put("username", username);
        // Put Http parameter password
        params.put("password", password);
        // Invoke RESTful Web Service with Http parameters
        invokeWS(params);
    } else{
        Toast.makeText(getApplicationContext(), "Please fill the form, don't leave any field blank", Toast.LENGTH_LONG).show();
    }
}
```

Figure 14. Login Async Task

Once we have all the required parameters, we can invoke the function that takes care of sending the request and receiving the response.

```
public void invokeWS(RequestParams params){
    // Show Progress Dialog
    prgDialog.show();
    // Make RESTful webservice call using AsyncHttpClient object
    AsyncHttpClient client = new AsyncHttpClient();
    client.get(Utility.CONNECTION_SERVER+"PECS/login/dologin",params ,new AsyncHttpResponseHandler() {
        // When the response returned by REST has Http response code '200'
```

Figure 15. Login WS part A

As we can see in the example, we initiate a new object of type AsyncHttpClient and we pass the parameters we gather earlier as an input.

However, it also requires overriding two other methods. The first is called `onSuccess()`. This method is invoked when the request is sent successfully, and we receive the desired response. Within this method, we implement the desired functionalities when we receive the response. For instance, when invoking the login method of the web service, we receive the information of the user logging in. In this case, we chose to save this information when the operation succeed. On the other hand, when the operation fails, the second overridden function is called instead. The `onFailure()` function is where we implement the handling of the error that caused the operation to fail. In our case, when the operation of logging in fails, we display the error message.

```
client.get(Utility.CONNECTION_SERVER+"PECS/login/dologin",params ,new AsyncHttpResponseHandler() {
    // When the response returned by REST has Http response code '200'

    @Override
    public void onSuccess(int i, cz.msebera.android.httpclient.Header[] headers, byte[] bytes) {
        User returnedUser = null;
        // Hide Progress Dialog
        prgDialog.hide();
        String response = new String(bytes);
        try {
            // JSON Object
            JSONObject obj = new JSONObject(response);
            // When the JSON response has status boolean value assigned with true
            if(obj.getBoolean("status")){
                Toast.makeText(getApplicationContext(), "You are successfully logged in!", Toast.LENGTH_LONG).show();
                // store user information locally
                String name = obj.getString("name");
                String username = obj.getString("username");
                String password = obj.getString("password");
                String email = obj.getString("email");
                returnedUser = new User(name, username, password, email);
                //System.out.println(returnedUser);
                userLocalStore.storeUserData(returnedUser);
                userLocalStore.setUserLoggedIn(true);
                // Navigate to Home screen
                navigatetoHomeActivity();
            }
            // Else display error message
            else{
                errorMsg.setText(obj.getString("error_msg"));
```

Figure 16. Login WS part B

```
    @Override
    public void onFailure(int statusCode, cz.msebera.android.httpclient.Header[] headers, byte[] bytes, Throwable throwable) {
        // Hide Progress Dialog
        prgDialog.hide();
        // When Http response code is '404'
        if(statusCode == 404){
            Toast.makeText(getApplicationContext(), "Requested resource not found", Toast.LENGTH_LONG).show();
        }
        // When Http response code is '500'
        else if(statusCode == 500){
            Toast.makeText(getApplicationContext(), "Something went wrong at server end", Toast.LENGTH_LONG).show();
        }
        // When Http response code other than 404, 500
        else{
            Toast.makeText(getApplicationContext(), "Unexpected Error occcured! [Most common Error: Device might not be connected to Ir
        }
    }
});
}
```

Figure 17. Login WS part C

In this project, we decided that the Android client will represent the graphical user interface to our web service implemented in separately on a web server. Therefore, the Android client is only displaying information to the user, and sending and receiving data to the web server.

Therefore, every time there is a need to contact the web server, we will use the Android Asynchronous HTTP Handler to send requests and receive responses as explained and illustrated earlier.

5.2. Results

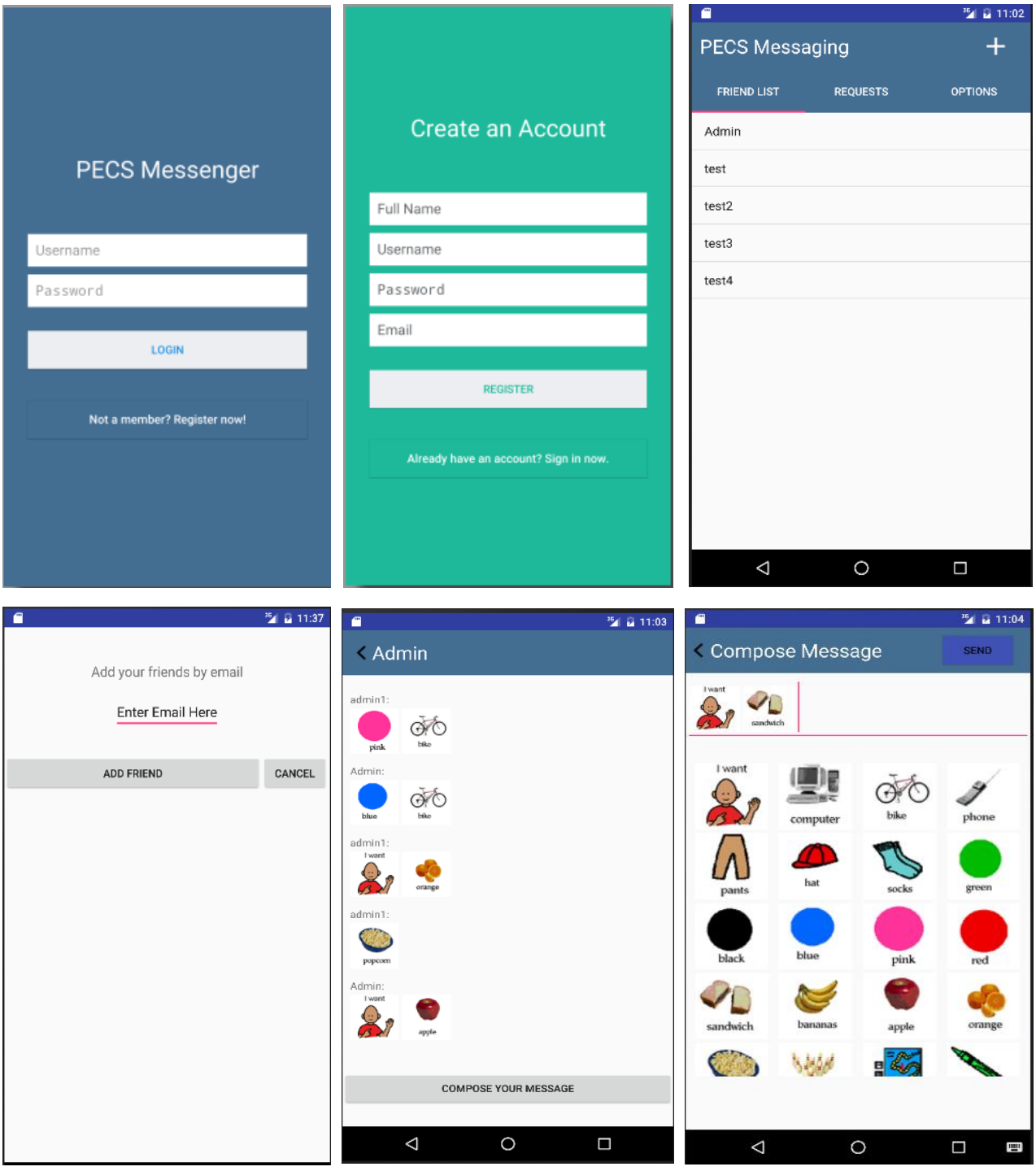

Figure 18. Application screenshots

6. STEEPLE Analysis

During the development of the project, we took into account different parameters commonly known as STEEPLE:

- **Societal:** The application addresses a social problem which is helping all the people suffering from the Autistic Spectrum Disorder (ASD) to learn speech and better communicate with other human beings.
- **Technical:** The application also facilitates the learning by incorporating PECS in a mobile application. This will not only encourage the learning of basic communication skills, but also to use mobile devices and how to use them.
- **Environmental:** The application does not present any environmental issues.
- **Ethical:** The application does not harm or lead to any unethical behavior. The application is designed to enhance the life quality of individuals suffering from ASD.
- **Political & Legal:** The application has no political concern, nor does it break any Moroccan or International law.
- **Economic:** This project is just a proof-of-concept, but if I were given a chance to develop it fully, we will make it available freely to the ASD community.

7. Future Work

Even though the majority of functionalities were implemented, the application is still a proof-of-concept and there is still room for modifications and optimization.

Currently, the PECS database has approximately a few dozens of the picture cards which is still small compared to the huge number of pictures available online. Therefore, populating the database with more pictures is one of the parts that still need some work.

Additionally, we can add optional functionalities such as voice recording to cover a larger range of customers. This will also allow the user to record the voicing of the object in any language, and therefore have a direct impact on learning the language and communicating with other people.

Predicting the next symbol to use within the composition of a message was initially part of the requirements; however, due to time constraints, this functionality was not developed. As far as I am concerned, I believe that this functionality can bring a lot to the table, especially when the software grows and incorporate more and more PECS. The more pictures there is in the system, the longer it will take to fetch for the desired symbol, which ultimately can cause frustration to people suffering from Autistic Spectrum Disorder.

8. Conclusion

Working on this capstone project was a real opportunity and a fruitful experience to learn even more than I have already studied during my past years at the university.

At the beginning of this experience, I planned to achieve three objectives. The first was to learn the Android environment and how to build an Android application. The only knowledge I had at the time was limited experience with the programming language Java which helped me even more than I expected. At the end of the project, I can say that I become a better programmer in terms of Java and in building Android applications. However, I still need to learn more as I always have this feeling of not knowing enough.

The second objective I had fixed for myself was to deepen my knowledge about web services. During the design phase of the project, I learned that there are two types of web services: SOAP and REST. Having already studied a little bit about the SAOP web services, the choice was obvious into choosing to work with REST. By the end of this project, I become aware of both technologies, so I can say that the objective was achieved.

Finally, the last objective that I set to myself was to learn a new concept in the Artificial Intelligence as we could also add a nice feature in the software that predicts the next symbol to use in a sentence. Unfortunately, because of time constraint, I could not implement this part of the project; however, I still managed to read about it and discuss it with my supervisor.

9. References

[1] Android Developers (2015). *Get Started with Android Studio* [Online]. Available at:

http://developer.Android.com

[2] Bondy, A.S. and Frost, L. (1985). *What is PECS?* [Online]. Available at:

http://www.pecsusa.com/pecs.php

[3] Apache (2015). HTTP Server Project. [Online] Available at:

http://httpd.apache.org/

[4] Baumann, M. (2014). *PexPix Autism*, retrieved from:

https://play.google.com/store/apps/details?id=appinventor.ai_matthewcb4.PecsPics

[5] Smith, J. (2012). *Android Asynchronous Http Client*, [Online]. Available at:

http://loopj.com/Android-async-http/

[6] Jersey (2015). *RESTful Web Services in Java*, [Online]. Available at:

https://jersey.java.net/

[7] National Institute of Mental Health (2011). *A Parent's Guide to Autism Spectrum Disorder* [Online]. Available at: http://www.nimh.nih.gov/health/publications/a-parents-guide-to-autism-spectrum-disorder/index.shtml

[8] Oracle (2013). *The Java EE 6 Tutorial*, [Online]. Available at:

https://docs.oracle.com/javaee/6/tutorial/doc/docinfo.html

[9] Pyramid Educational Consultants (2015). *PECS IV+*, [Online]. Available at:

https://itunes.apple.com/us/app/pecs-iv+/id919593979?mt=8

[10] TechTarger (2010). *What is Unified Modeling language?* [Online]. Available:

http://searchsoftwarequality.techtarget.com/definition/Unified-Modeling-Language

[11] Universidad de Valparaíso (2014). *Proyect@ PECS*, [Online]. Available at:

https://play.google.com/store/apps/details?id=air.TestPECS

[12] W3Schools (1999). *JSON Introduction* [Online]. Available at:

http://www.w3schools.com/json/

Appendix A

A.1 Picture Collection

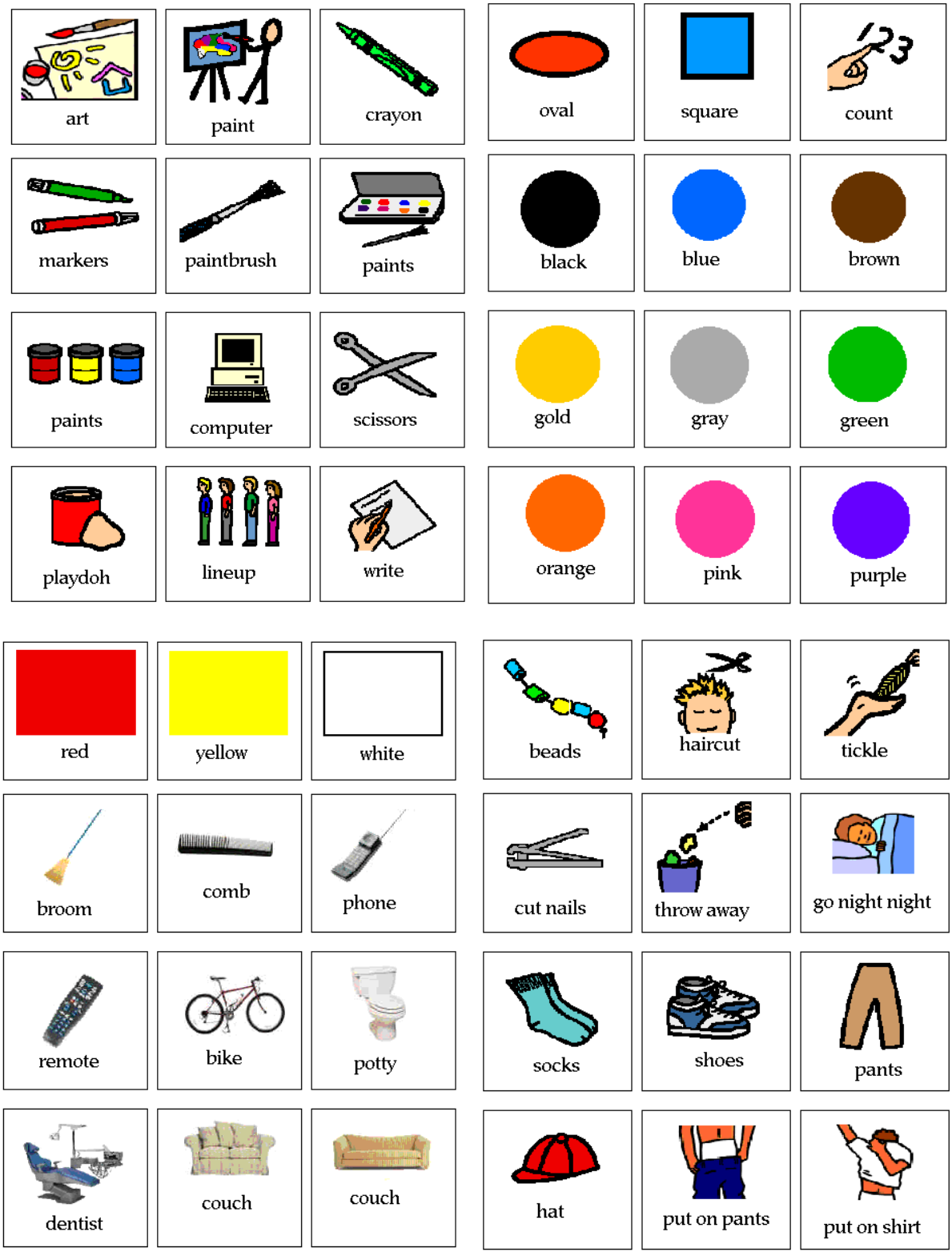

Figure 19. Pictures Collection