MOBCOLL

wonderland

Software Design Description v1.1

Sinan Küçükköseler

Zeliha Şentürk

Cansu Tetik

Murat Türe

17.01.2013

Table of Contents

1. Introduction

1.1. Purpose

This Software Design Document describes the architecture and system design of MOBCOLL The purpose of this document is to help the reader visualize the project presented. It explains the purpose and features of the system, the interfaces of the system, what the system will do and the constraints under which it must operate. This document will provide a direct approach to the development of this project. This document covers the development process up until the implementation phase also is the final deliverable before true implementation.

1.2. Scope of the Project

MOBCOLL will be a mobile application that is designed for item collection management. The goal of the project is to provide users the ability to manage their collections via their mobile devices with Android OS with the features of keeping track of missing and newly released items and checking the item sets, interact with other collectors with lending and borrowing facilities.

One of the main benefits of MOBCOLL is that it provides a good solution to complete the item sets and reach the desired items, as they are of great importance to collectors. Moreover, the application has a user-friendly interface that makes organizing the collections via mobile devices available for collectors.

1.3. Overview of the Document

The next chapter of this document gives an overview of the system by explaining the problem, technologies used, brief description of the application and context diagram.

The third chapter describes system architecture with subchapters architectural design, decomposition description and design rationale. Major subsystems and data flow are explained in that section.

The fourth chapter data design includes information about data structures and provides the data dictionary.

In the fifth chapter, component design is given with pseudocodes and explanations.

The sixth chapter the intended user-interface design is described with sample figures and discussion of screen objects and actions associated with those objects is given.

Requirements matrix is provided in the seventh chapter of this document.

1.4. Definitions and Abbreviations

1.4.1. Definitions

Term	Definition
Image Processing	Any form of signal processing applied to images (with the aim of recognition in this project)
Operating System	A collection of software that manages computer hardware resources and provides common services for computer programs

1.4.2. Abbreviations

Term	Definition
OS	Operating System
ADT	Android Development Tools
SDK	Software Development Kit

2. System Overview

2.1. Technologies Used

The mobile application part of MOBCOLL project will be developed by using Java through Eclipse. The server will be implemented in Python. SQLite will be used for the database management and communication with the database. Since MOBCOLL is an application for Android – powered devices, ADT and Android SDK will be used.

2.2. Application Overview

MOBCOLL is a mobile application with server and database connection developed to be used in devices with Android OS. Basically, it will have functions as data managing, image recognition, web based searching and web based operations specific to item collecting.

2.3.Context Diagram

MOBCOLL is designed for collectors. It helps the users share their collections with others, communicate with them and reach their collection's condition easily.

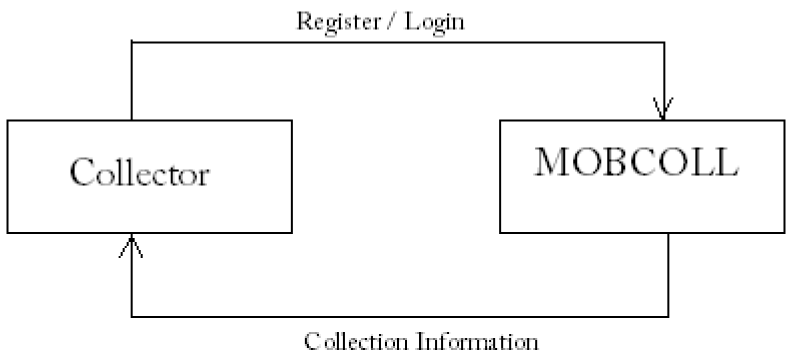


Figure 1 - Context Level DFD of MOBCOLL

3. System Architecture

3.1.Architectural Design

MOBCOLL is a mobile application which communicates with the online server to carry out collection keeping requirements. System architecture can be divided into three major parts; mobile client, server and the database.

3.1.1. Client

Client module provides the user interface and makes the connections with server. Mobile user interface is organized by the client. It abstracts the server from human errors and user misuses. For example, the input validation and the encryption of the information sent to the server are done by the client. Client also keep record of the own collection of the user in a local database to reduce the traffic to the server.

3.1.2. Server

Server is the one of the most important module in this system. It provides web services for the communication with the clients. Also, it serves as a bridge between the database and the clients. Database can only be reached by the server, thus security of the server is crucial for the safety of the users' information. Authentication is performed by the server.

3.1.2.1. Input Output

Post parameters sent to server are given in the server side part of the pseudecodes.

Creating a Collection:

```
ClientSide:
   get(collectionName)
        sendToServer(collectionName,currentUserID)

   if( result = waitForResult() )
                show “Collection created successfully”
        else
                show “Collection could not be created”

Server Handler:
        //Post Parameters:
        //collectionName,currentUserID
        collectionName,currentUserID  = waitForRequest()
       q =
prepareAddCollectionQuery(collectionName,currentUserID)
       result = executeQuery(collectionName)
      replyRequest(result)
```

Fetching Collections of a User:

```
Client Side:
    sendToServer(currentUserID)
    collections = waitForResult()
    show(collections.name)

Server Handler:
    //Post Parameters:
    //userID
    userID = waitForRequest()
    q = prepareCollectionsOfUserQuery(userID)
   collections = executeQuery(q)
replyRequest(collections)
```

Adding Item To Collection

```
Client Side:
    sendToServer(currentCollectionID,cardID)
    result = waitForResult()
    if(result)
        show("Successfully Added")
    else
        show("Could not be Added")

Server Handler:
    //Post Parameters:
    //currentCollectionID, cardID
    userID = waitForRequest()
    q = prepareCollectionsOfUserQuery(userID)
   collections = executeQuery(q)
    replyRequest(collections)
```

Fetching Cards of a Collection:

```
Client Side:
    sendToServer(currentUserID)
    collections = waitForResult()
    show(collections.name)

Server Handler:
    //Post Parameters:
    //userID
    userID = waitForRequest()
    q = prepareCollectionsOfUserQuery(userID)
    collections = executeQuery(q)
    replyRequest(collections)
```

Web Services:

3.1.2.1.1. Login

User name and password is sent as post parameters to this service. Server creates the query and executes it in the server database. Result is sent back to the client.

3.1.2.1.2. Register

New user name, password and email is sent as post parameters to this service. Server makes the query according to the input and executes it in the server database.

3.1.2.1.3. Logout

Server gets the current user id from the session information in its main memory. It removes that session information.

3.1.2.1.4. Search

Search string is sent as a post parameter to this service. Server creates the query with a ‘LIKE’ keyword to ensure fast searching, and executes it in the database.

3.1.2.1.5. Card

Card ID is sent as a post parameter to this service. Server fetches the card information and sends it to the client using the server database connection.

3.1.2.1.6. Collections of a User

Current user id is sent to the server as a post parameter. Server creates the query and executes it in the database. The ID’s of the collections that the current user has, is sent to the client.

3.1.2.1.7. Collection Info

Collection id is sent to the server as a post parameter. The query for getting the cards in a collection is created with respect to the input id and the result is sent back to the client.

3.1.2.1.8. Create Collection

Collection name and the current user id are sent to the server as post parameters. The query for creating a new collection is created with respect to the input and the result is sent back to the client.

3.1.2.1.9. Add Card to a Collection

Active collection id and card id are sent to the server as post parameters. The query for adding a new card to a collection is created and after execution, the result is sent back to the client.

3.1.3. Database

Database, by nature, is the module that holds the user and collection information. Its security is abstracted by the server. Database will be in responsible of fast access of the data by using optimized data storage techniques.

3.2.Decomposition Description

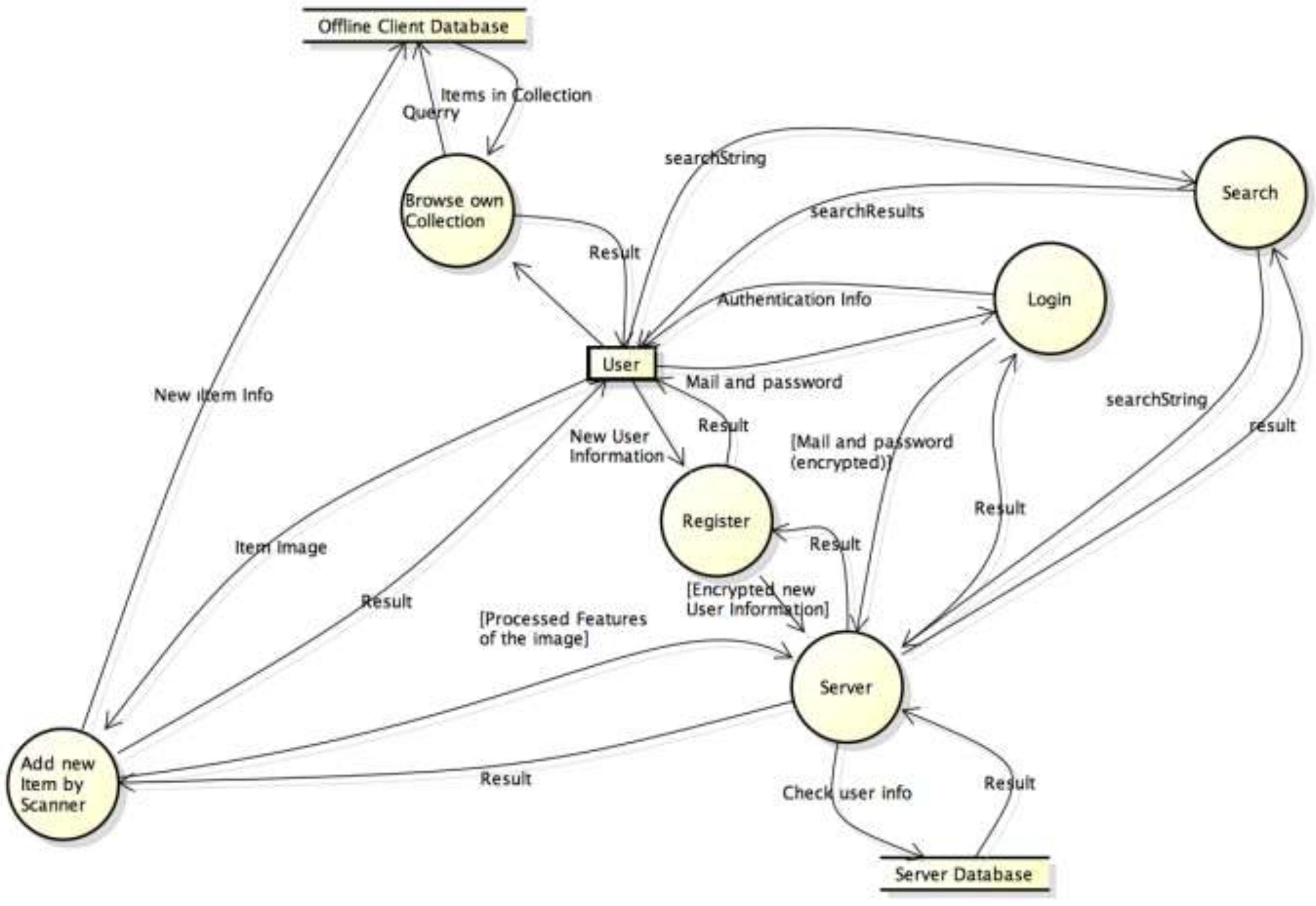


Figure 2 - Data Flow Diagram

In the above data flow diagram, the data that travels through the system is shown. User is the outside actuator which gives inputs and receives outputs through the user interface. The first processes that user sends info, are implemented in the client application. After the data is validated and encrypted, they are sent to the server. Server connects to the database and fetches the output to be sent to the client. Finally, after every operation output result is shown to the user through the client's user interface.

3.2.1. Login Functionality

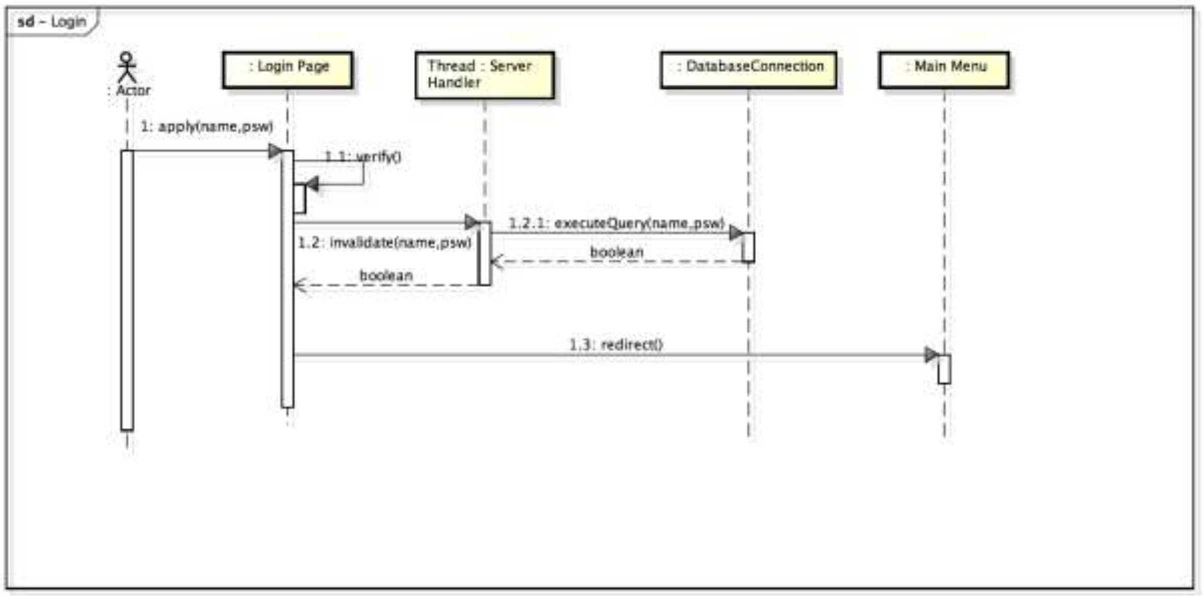


Figure 3 – Login

Login operation starts when the user enters its user information and clicks login button. User interface takes this information, validates and encrypts its and sends to the server handler. Server creates a handler when receives the request, handler uses the database connection to check the user information and sends the result back to the client.

Moreover, the user will be able to select login via Facebook or Twitter accounts, by tapping the corresponding social network icon button. As the user chooses to login via social network account, user interface sends this information to the application server. Then, server validates the user data using the corresponding social network's API.

If the result is success, user interface redirects the user to the main menu.

3.2.2. Search Functionality

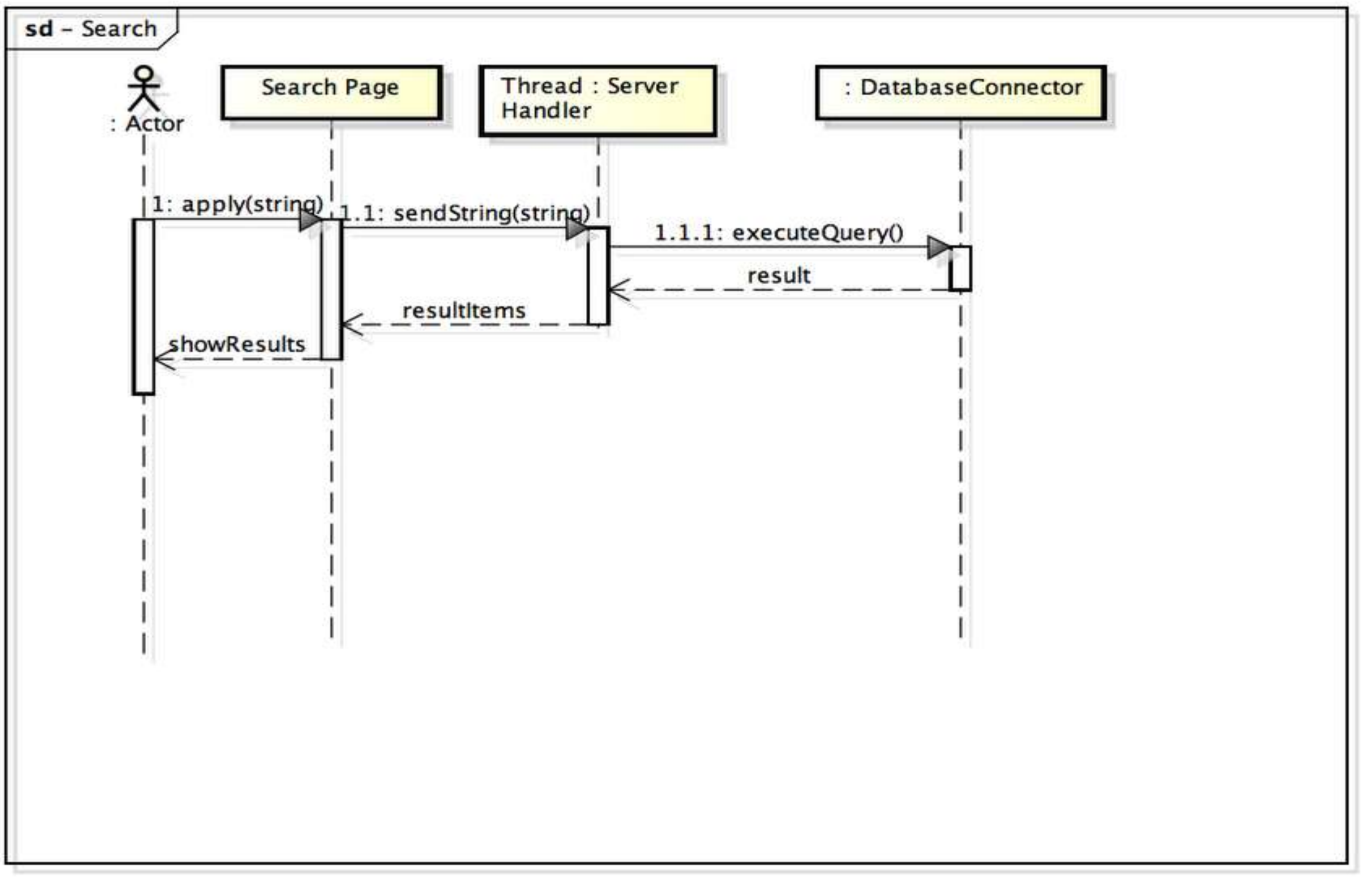


Figure 4 – Search

Search operation can be done from the main menu. Like every other operation, data flow through the client to the database. Then the result of the query gets back to the user interface and is shown in the main menu. This result includes collection items and any detailed information regarding the items.

3.2.3. Register Functionality

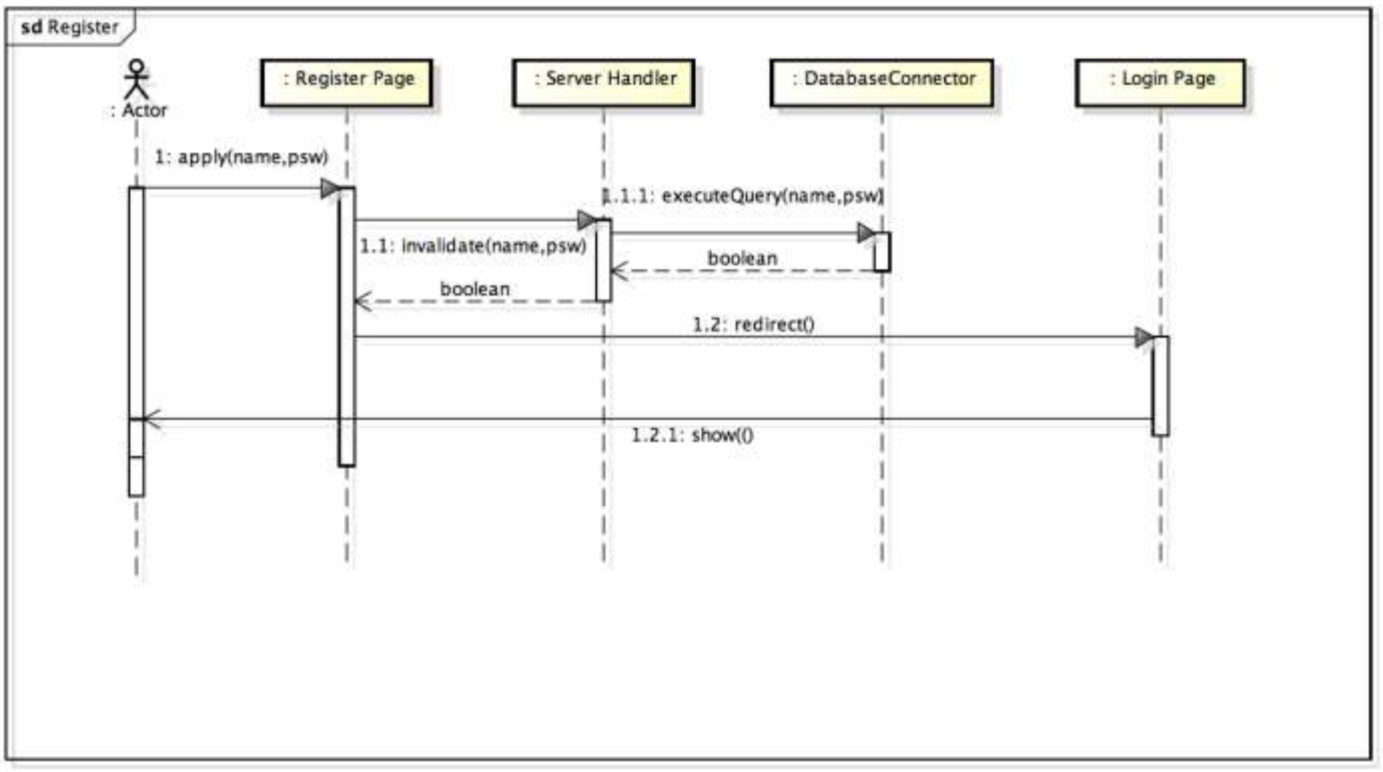


Figure 5 – Register

When user opens the application first time, s/he must register a new account. Through the register page, application gets the new user information, encrypts and sends it to the server. Server accepts the request and the newly created handler thread executes the query. If the new user info is unique then the handler sends success result to client. Then user interface redirects the user to the login menu. Moreover, the user will be able to select register via Facebook or Twitter accounts, by tapping the corresponding social network icon button on the register page. As the user chooses to register via social network account, user interface sends this information to the application server. Then, server validates the user data using the corresponding social network's API.

3.2.4. Scan Functionality

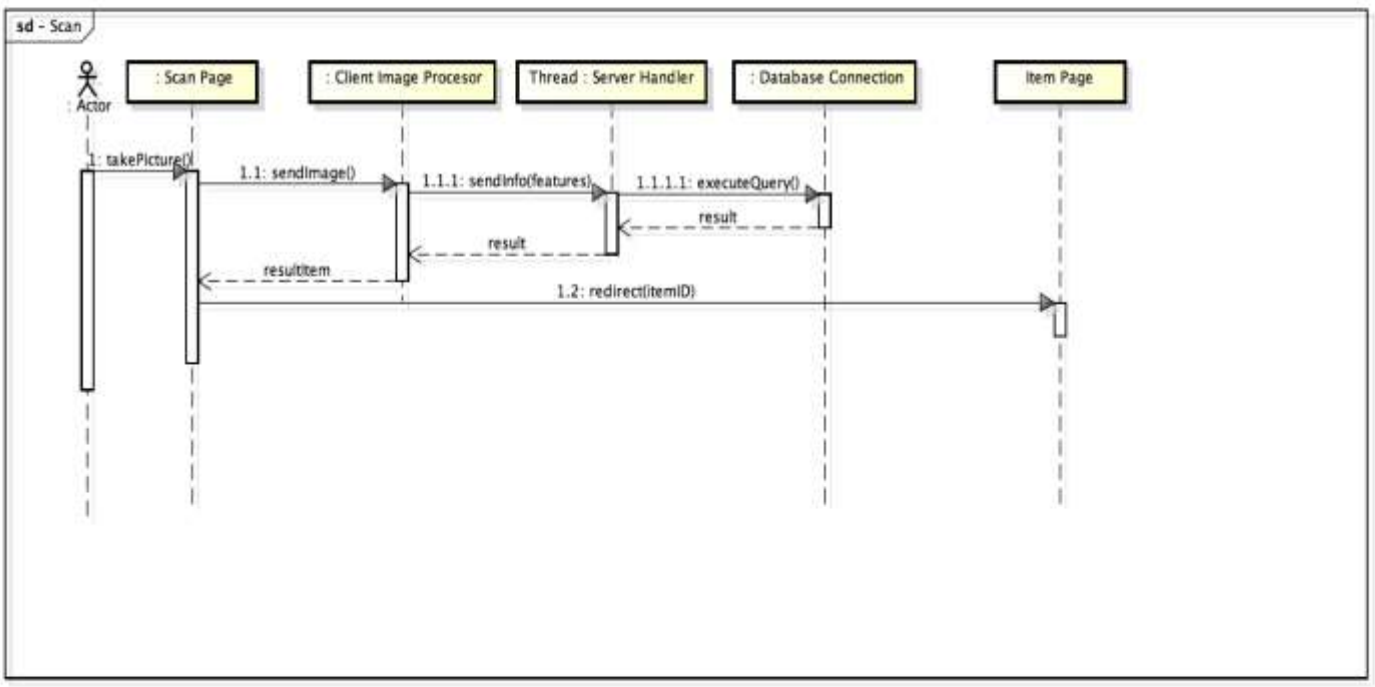


Figure 6 – Scan

Scan operation is done in scan page, which look like an ordinary camera page. User takes the picture. Client extracts the image information to be processed in the server. Then if any matching item is found, the user is redirected to the corresponding item page. It handles requirement 4.2.4.

3.2.5. Lend and Borrow Functionality

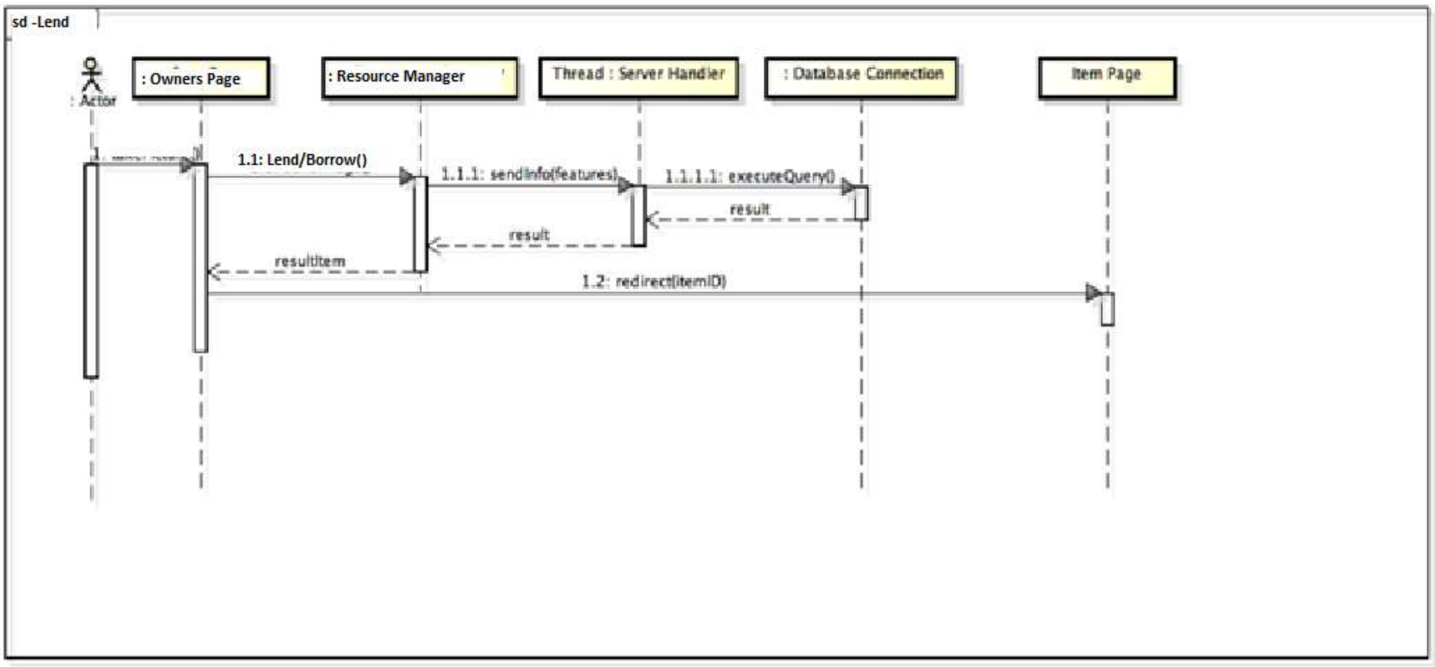


Figure 7 – Lend and Borrow Functionality

User can send borrow requests to other collectors in “Owners Page” by the corresponding methods. Resource manager of the application in the client-side will send this method info to the database and related results will return and can be seen in the user interface of the application by the user.

3.2.6. Add Cards to a Collection Functionality

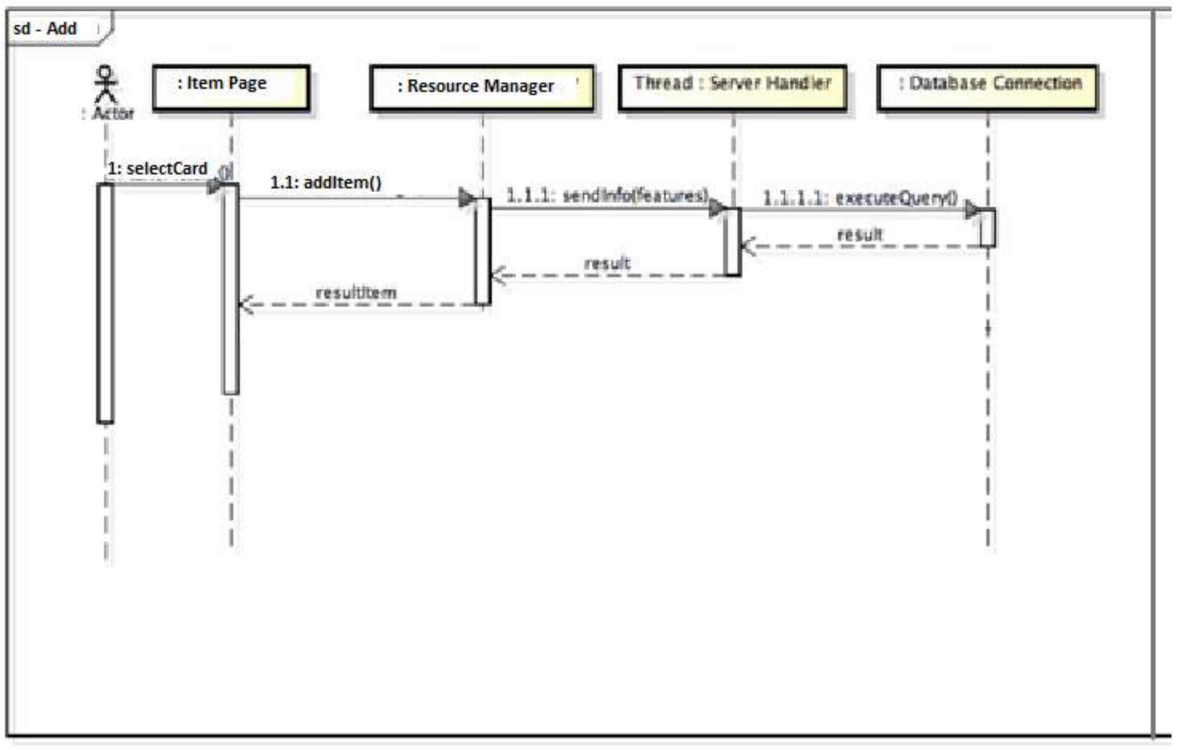


Figure 8 – Add Cards to a Collection

User can add items to their collection by selecting an item from the interface. Data flow is similar to the previous functionalities; it includes Resource Manager and Database Server working consecutively.

3.2.7. Deployment Diagram

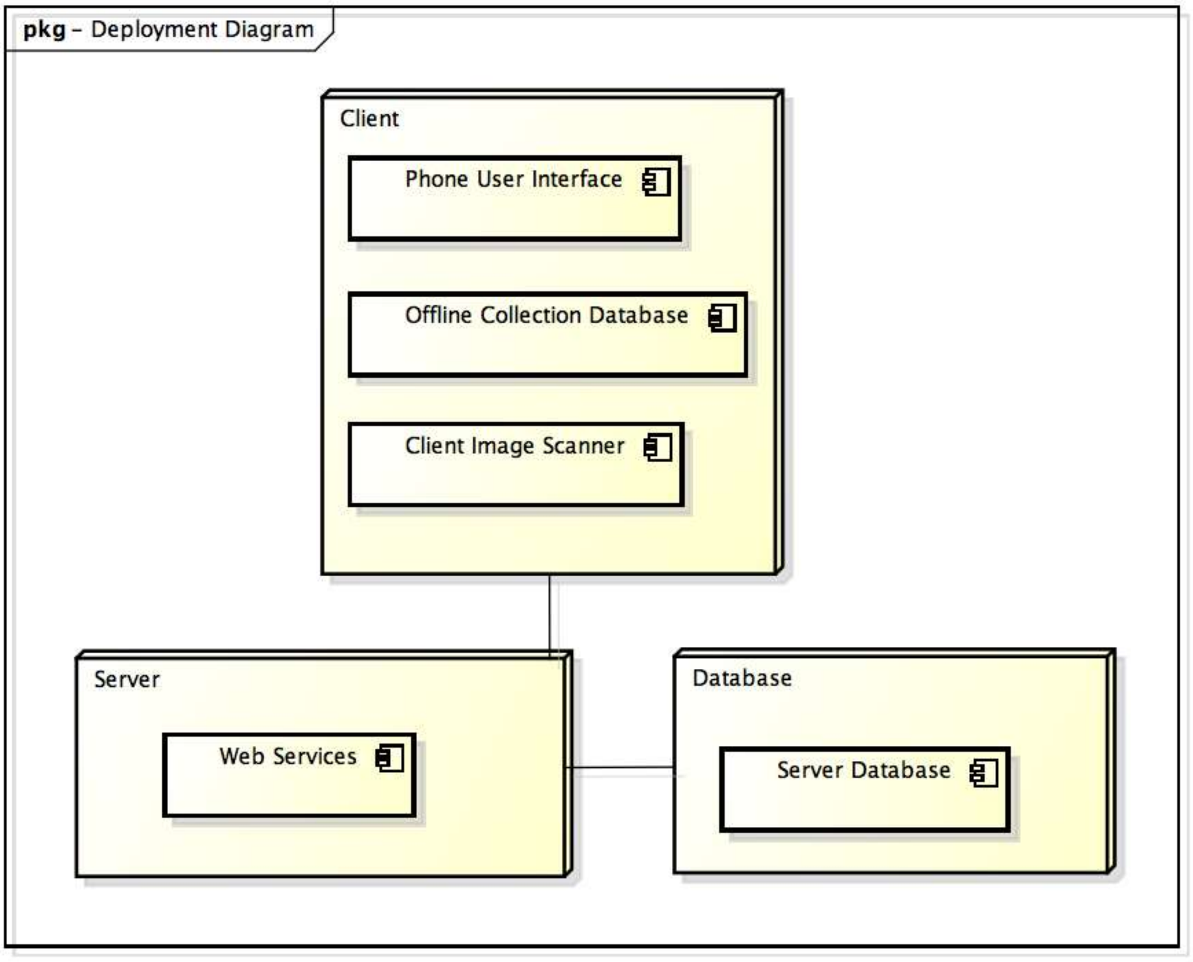


Figure 9 - Deployment Diagram

3.3.Design Rationale

The ultimate purpose of this application is to be responsive as possible and easy to use. Thus limiting the server's payload and connection bandwidth is essential. In this design, client module does every preprocessing, like input validation.

As the most frequent operation done by the user is browsing own collection, keeping a copy of its own collection in the client application storage will also reduce the server's response time.

Main tradeoff of the above feature is that the local database has to be updated at some dynamic time intervals for showing the updated collections.

Abstracting the database from the client by the server makes the database less reachable and increases security.

4. Data Design

4.1.Data Description

4.1.1. Database Relations

The data storage is a crucial part of the application since a collection data management is the core aim that the application is targeting.

The application data is stored in relational database on our database server using SQLite. The Entity – Relationship Diagram of the database is given below.

To start with describing the schema; it has the 'Collector' entity which will hold the information of the application users such as username, password and e-mail as shown. And there are the 'Item' and 'Item Set' entities which contain the information of the collection items and the sets that consists of some particular collection items respectively.

Finally there is 'Has' relation which describes the possession of a particular item of a particular collector. Furthermore, there is the 'Item Transfer' relation which holds data entries of users borrowing or lending collection items which they own as shown with the 'has' relation. Finally, there is the 'Contains' relation which stands for that some particular items come together to create an item set that exists in our application data.

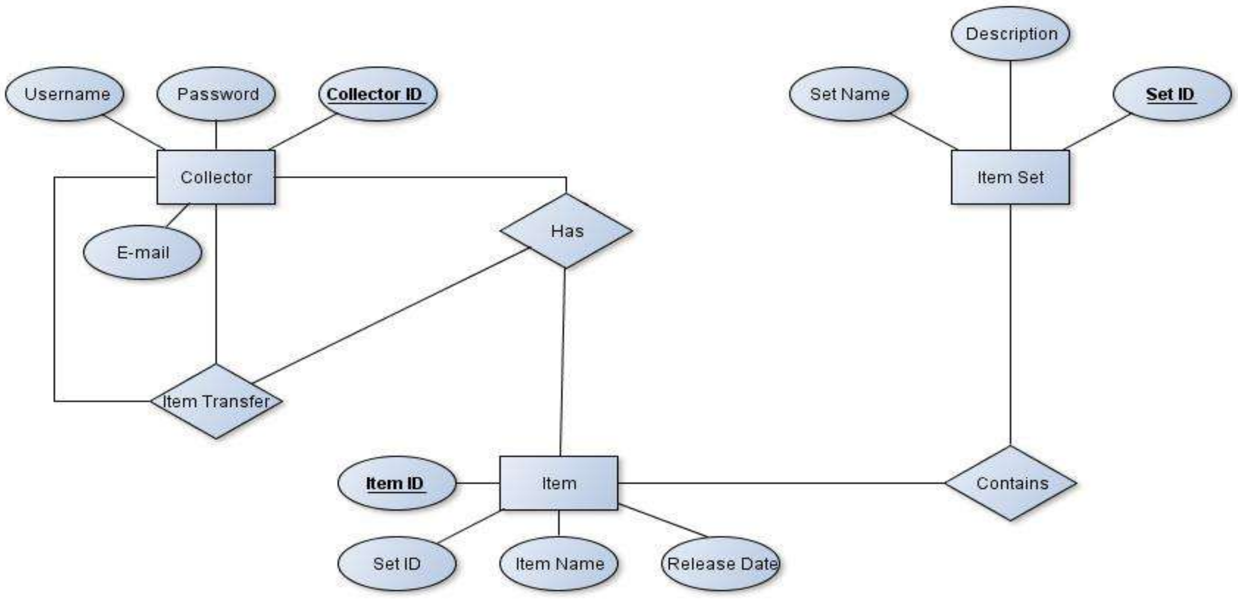


Figure 10 - ER Diagram

4.1.2. Class Overview

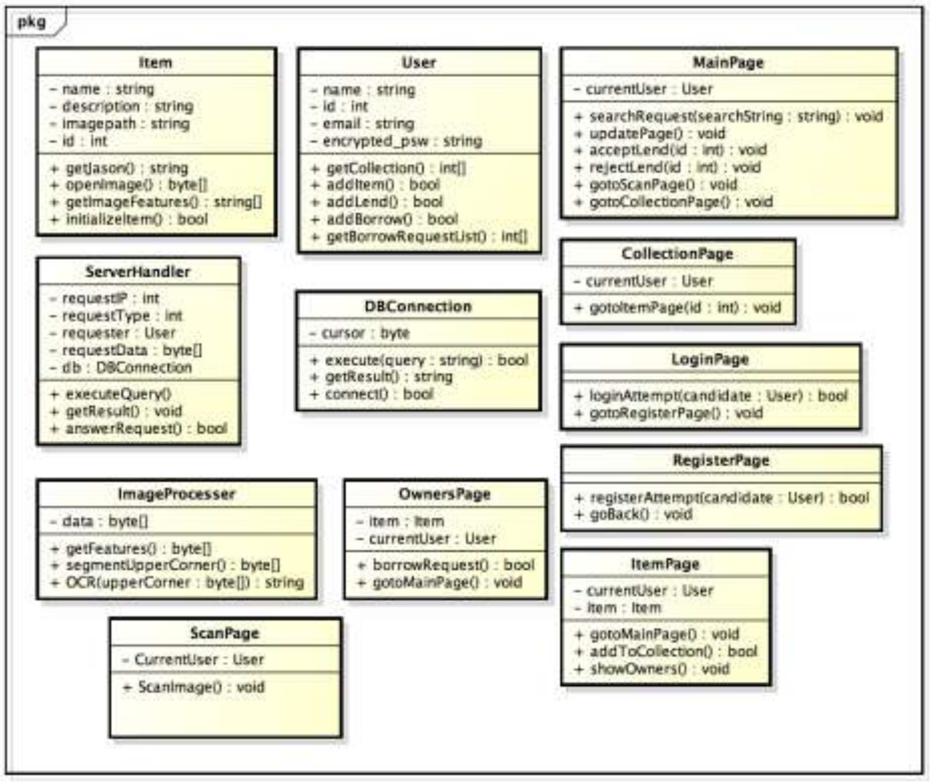


Figure 11 - Class Diagram

- Item class represents the items in the database. It is instantiated at both the server and the client.
- User represents the collectors in the application. It can be used in both the server and the client. User information is hold here.
- RegisterPage class is used in the client side. It covers the requirement 4.2.1 .
- LoginPage is used in the client side. It covers the requirement 4.2.2 . If the user can log on successfully, this class initialize the currentUser variable which is used in throughout the application.
- ServerHandler is the thread that main server creates when a new request for the web service comes. It uses the DBconnection to query the database.
- MainPage handles the operations in the main page. Also the borrow requests come here for the user to perform accept or reject operations. Searching for items and collectors is done in this page.
- DBConnection holds the cursor for the database. It has privilege to perform queries. Also, it returns the result of the queries.
- ImageProccesor class implements the feature detection from an image. It handles the requirement 4.2.4.
- ItemPage show a particular item. It is used in the client application. It gives the user operations like add to collection (requirement 4.2.6) and view owners of that particular item.

4.2.Data Dictionary

The entity types, fields belong to them and the type and description of the fields are given in the table below.

Entity	Field Name	Type	Description
Collector	User ID*	Integer	It is the unique id which every user has their own
	Username	Text	It is the unique username of every user which they will use within the actions of the application
	Password	Text	It is the encrypted text field which user must provide during logging-in
	E-mail	Text	It is the personal e-mail address of the user
Item	Item ID*	Integer	It is the unique id of an item that exists in the application data
	Set ID	Integer	It is the unique id of the set which the current item belongs to
	Item Name	Text	It is the name of the current item
	Release Date	Date	It is the date of which the item is released
Item Set	Set ID*	Integer	It is the unique id of an item set within the application
	Set Name	Text	It is the name of the item set
	Description	Text	It is the detailed description of the item set which the users may be interested in

5. Component Design

There are five major components which made up the MOBCOLL system.

- Web Services
- Server Database
- Phone User Interface
- Offline Collection Database
- Client Image Scanner

5.1.Web Services

The first major component of the system is the web services. Web services, which are implemented by HTTP protocol, established a reliable communication between the server and the client. Server will accept requests from the client from port 80 and after evaluating the request, will send the output to the client.

Clients will use GET or POST HTTP actions to request or send some data to the server.

The layout of the forms in these different requests (login , register etc..) will be set in the same way in both server and the client. Also, these forms will match with the input structures in the user interface.

5.2.Server Database

The server database is the most essential data storage of this application. It stores the most up to date information. Every request and change comes to the server as predefined queries from the server. Only administrators have the privilege to make changes and view the whole database.

The photos of the items, which can be used to scan or show to the users, are kept in the operating system’s disk storage and the file paths corresponding to them will be kept in the database.

5.3.Phone User Interface

Client application in the phone provides the user interface. The main purpose of this component is to be as responsive as possible to the user. The activities , which corresponds to different pages in the user interface, consists of Login , Register , Main Menu , Collection , Item and Scan pages.

Main menu can be reached after the user log on and it is used for easily getting into other pages and also searching for items is performed in the main menu. User interface will be specified with more detail in User Interface Design section.

5.4.Offline Collection Database

This offline database will reside in client storage. It will be used for actions regarding user's own collection. It will be updated whenever user makes some changes about its own collection.

5.5.Client Image Scanner

User can scan items to get information about that item in a fast way. Extracting information from that image will be done in the client. Otherwise, sending the image to the server would be very slow.

Optical character recognition and image segmentation techniques will be implemented in this component.

5.6.Component Functionality

The algorithms used for implementing functionalities, which have been shown in the section 3.2, are specified below in pseudocode:

5.6.1. Register

```
Client Side:

get(username,password)

if( validate_input(username,password) )

        sendToServer(encrypt(username,password))

        if( waitForResult() )

                redirect to registerPage

        else

                show wrong name or password

else

        show error



Server Handler:

//Post Parameters:

//name, password, email

data = waitForRequest()

decrypt(data)



if( executeQuerry() )

        replyRequest(true)

else

        replyRequest(false)
```

5.6.2. Login

```
Client Side:

get(username,password)

if( validate_input(username,password) )

        sendToServer(encrypt(username,password))

        if( waitForResult() )

                redirect to mainPage

        else

                show wrong name or password

else

        show error



Server Handler:

//Post Parameters:

//name, password

data = waitForRequest()

userInfo = decrypt(data)



if( executeQuerry(userInfo) )

        replyRequest(true)

else

        replyRequest(false)
```

5.6.3. Scan Image

```
Image Scanner:

get(rawImage)

    sendToServer(extractFeatures(rawImage))

    if( foundItem = waitForResult() )

        redirect to itemPage(foundItem)

    else

        show no items found



Server Handler:

//Post Parameters:

//featureList

imagedata = waitForRequest()



if( foundtem = executeQuerry(imagedata) )

    replyRequest(foundItem)

else

    replyRequest(null)
```

5.6.4. Search for Item

```
Image Scanner:

get(searchString)

    sendToServer(searchString)
```

```
        if( foundItems = waitForResult() )
                show foundItems
        else
                show no items found


Server Handler:
//Post Parameters:
//searchString
searchString = waitForRequest()
foundtems = executeQuerry(searchString)
replyRequest(foundItem)
```

6. User Interface Design

6.1.Overview of User Interface

When a collector wants to launch MOBCOLL for the first time, the application directs to the login / register page and the collector needs to fill the required information (user name, e-mail and password) to register to MOBCOLL database. The user stays logged in unless he/she logs out. In collection page, the collector can browse his/her owned items, and can add items from the collection in the item page. Moreover, one can acquire information regarding the item properties from this page. In scan page, the collector activates the camera to scan an item and see the matching results.

6.2.Screen Images

6.2.1. Login / Register Page

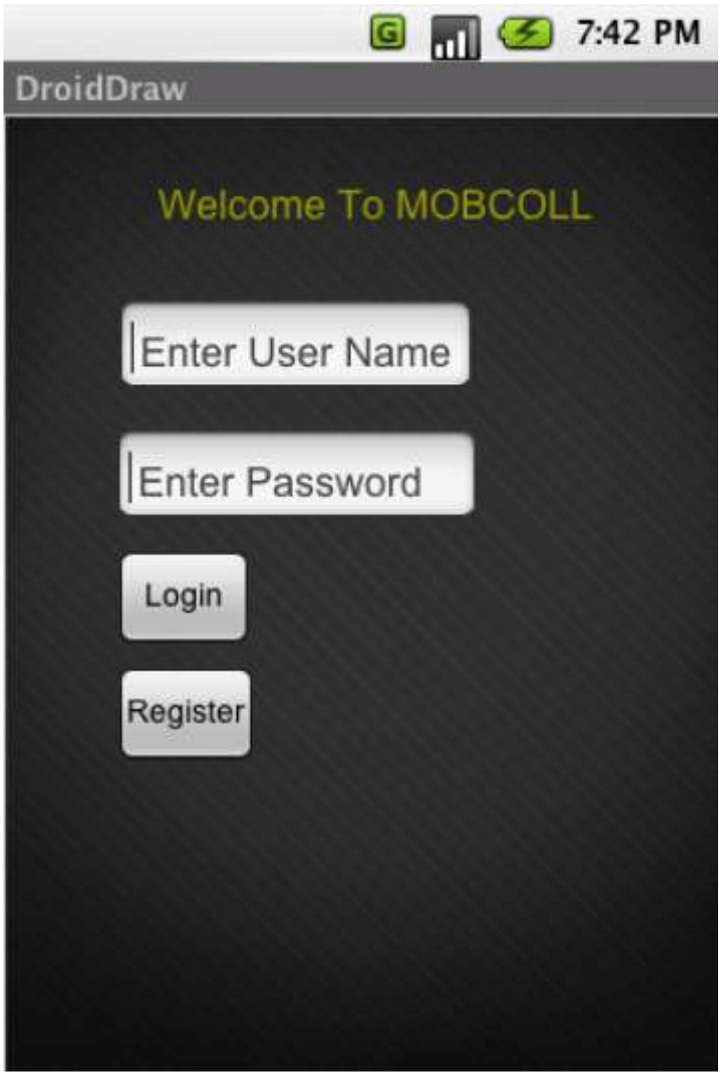


Figure 12 - Login / Register Page

This page enables users to log in or register to the MOBCOLL application.

6.2.2. Register Page

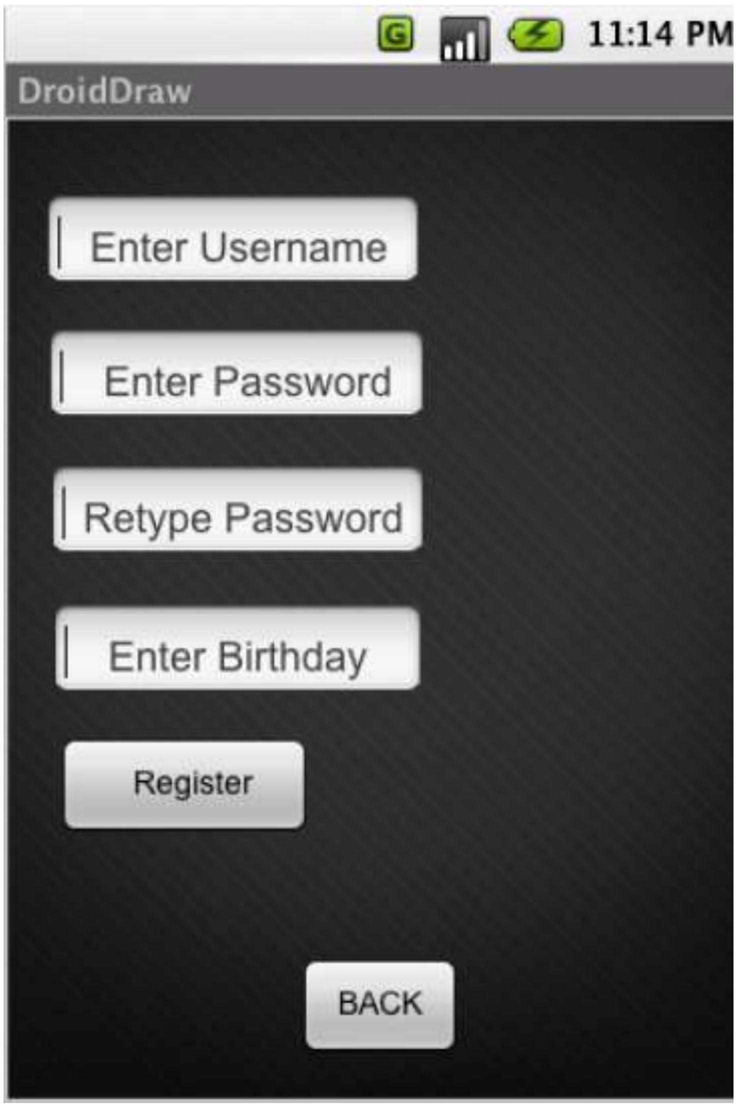


Figure 13 - Register Page

If the user is not registered, he/she can register through this page.

6.2.3. Main Page

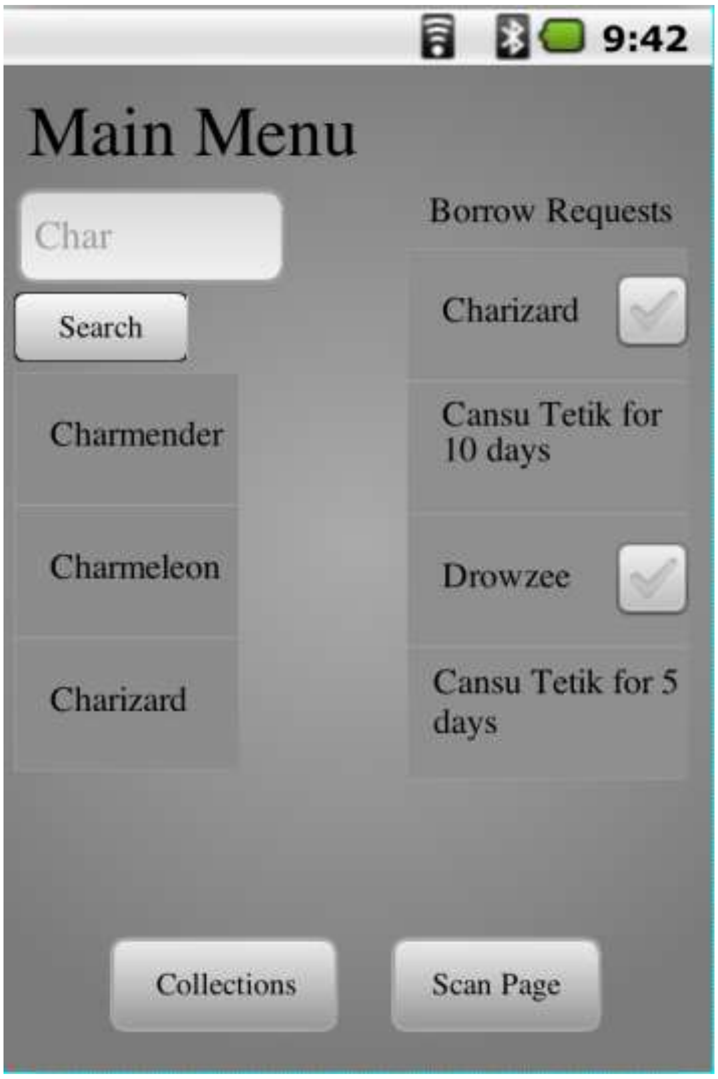


Figure 14 - Main Page

From this page users can search for other collectors or for items, can accept or reject borrow requests if there are any. They can also see their collections and scan an item and logout, through the buttons.

6.2.4. Scan Page

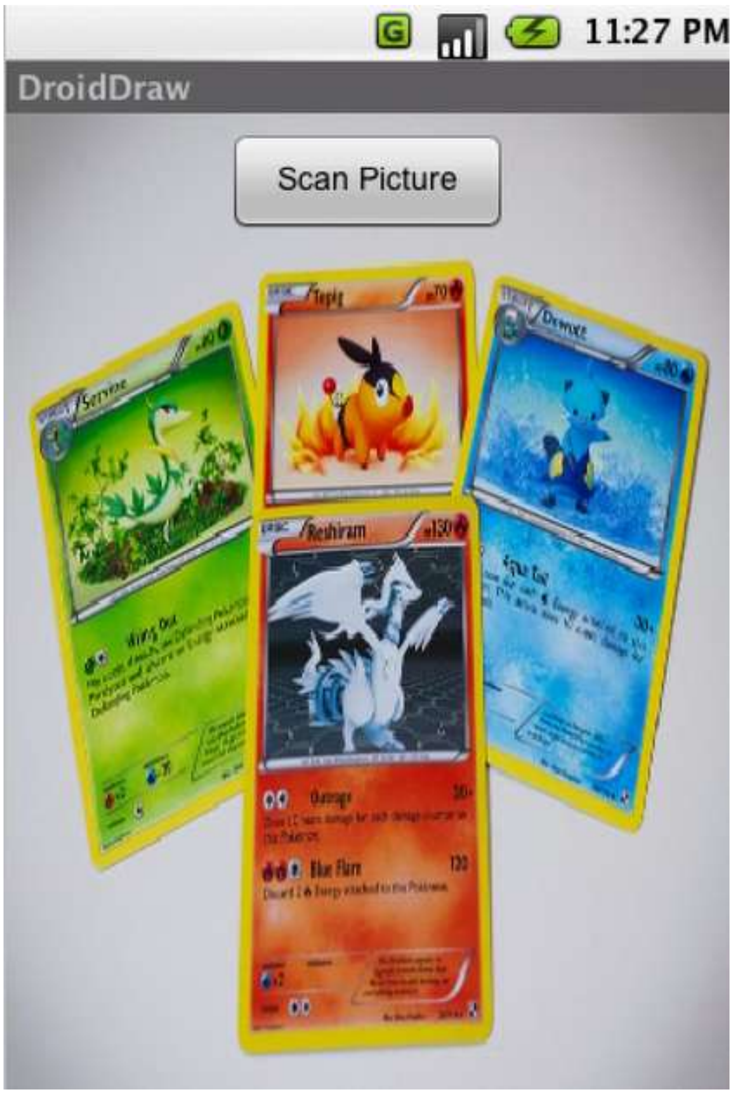


Figure 15 - Scan Page

This page lets users scan a picture and search for it in the database.

6.2.5. Collection Page

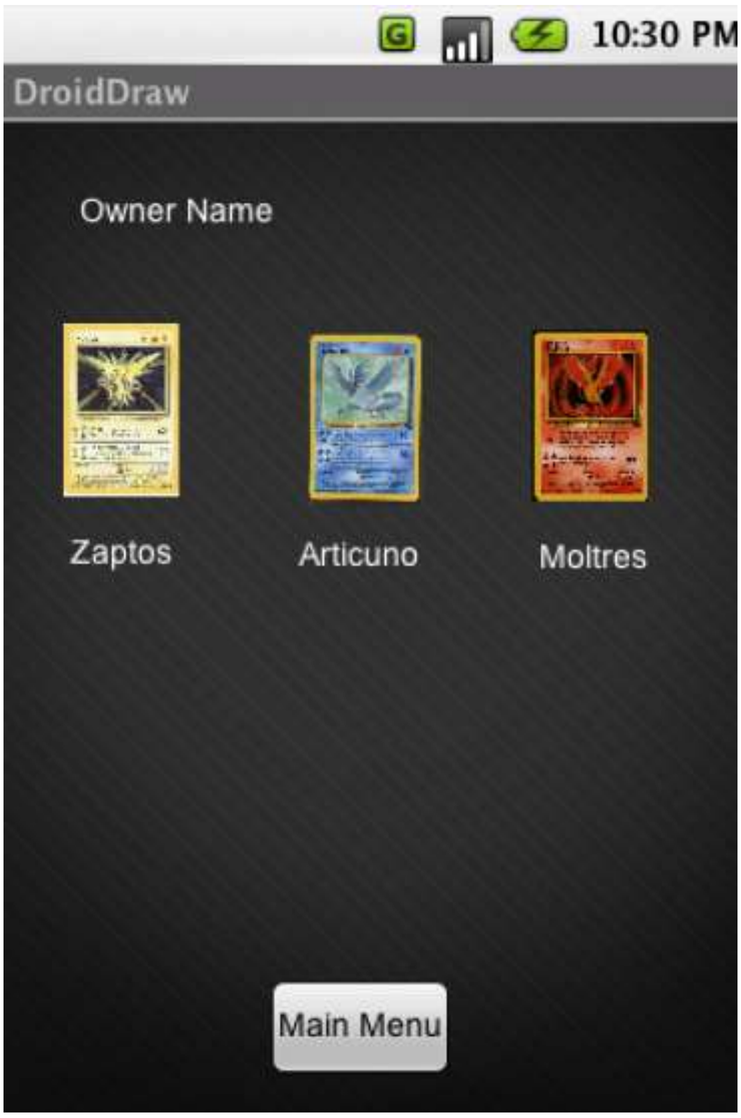


Figure 16 - Collection Page

Users can see their collection items (up to 6 per page), and swipe up and down to see the rest of their owned items.

6.2.6. Item Page

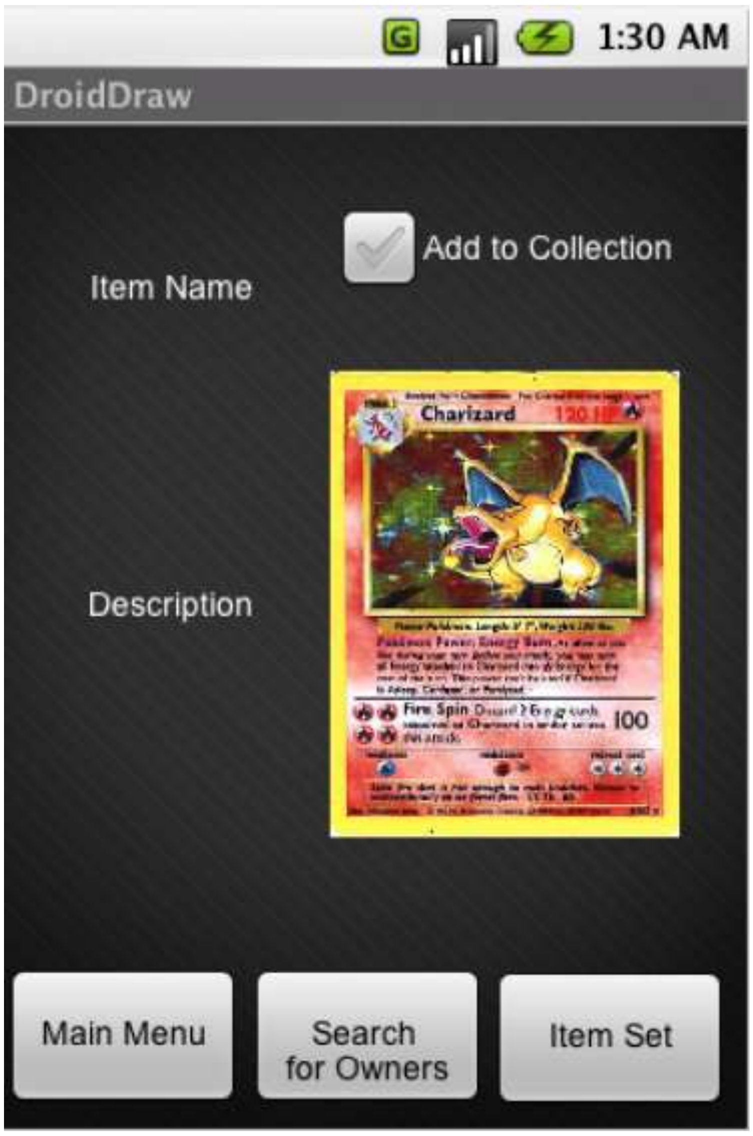


Figure 17 - Item Page

In Item Page, the name and brief description of the item can be seen, as well as an option to add the item to the collection. Users can also go to main page and search for the owners who have that item.

6.2.7. Item Set Page

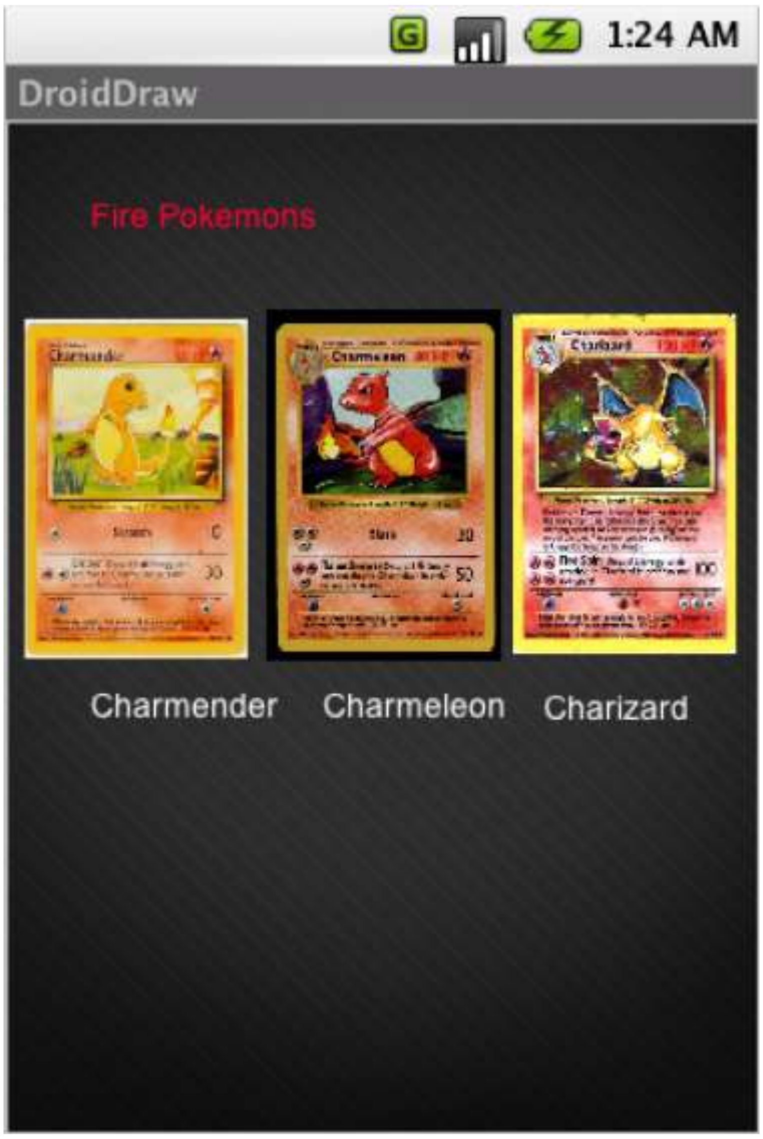


Figure 18 - Item Set Page

In this page, users can see the items of a set.

6.2.8. Owners Page

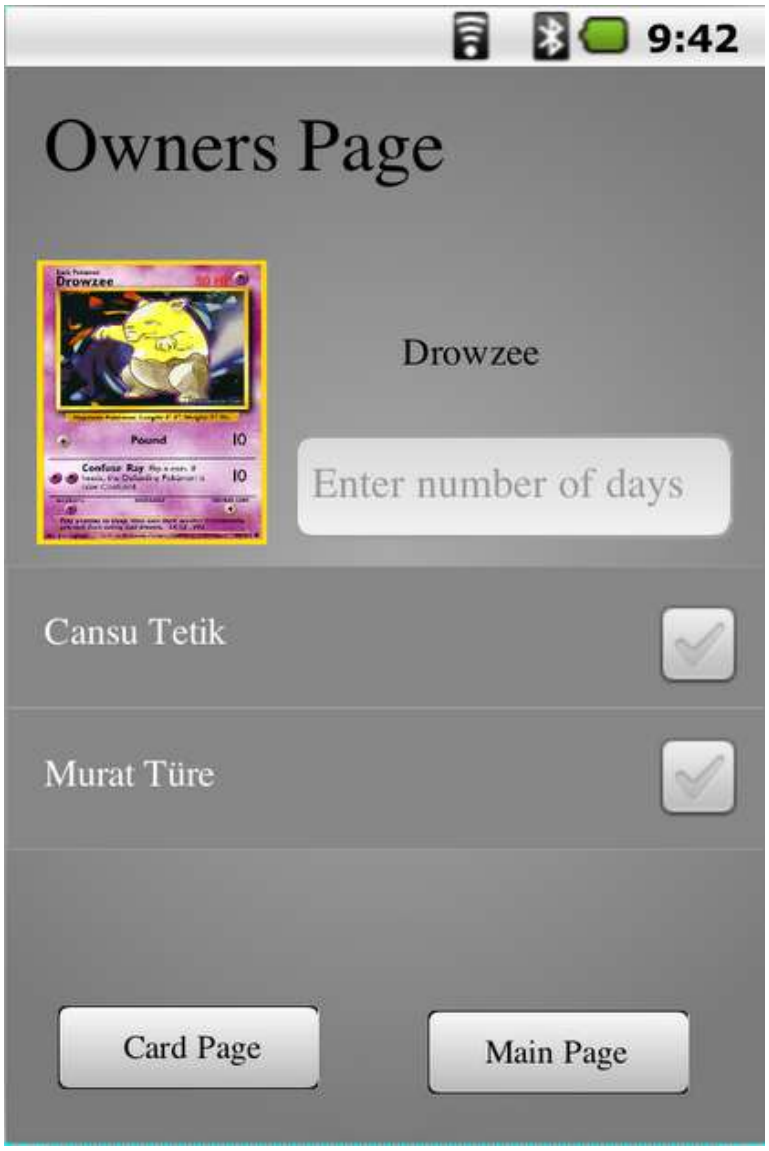


Figure19 - Owners Page

From this page users can make borrow requests for a specific item from the owners list, or they can go back to the Main Page.

6.2.9. Missing Items Page

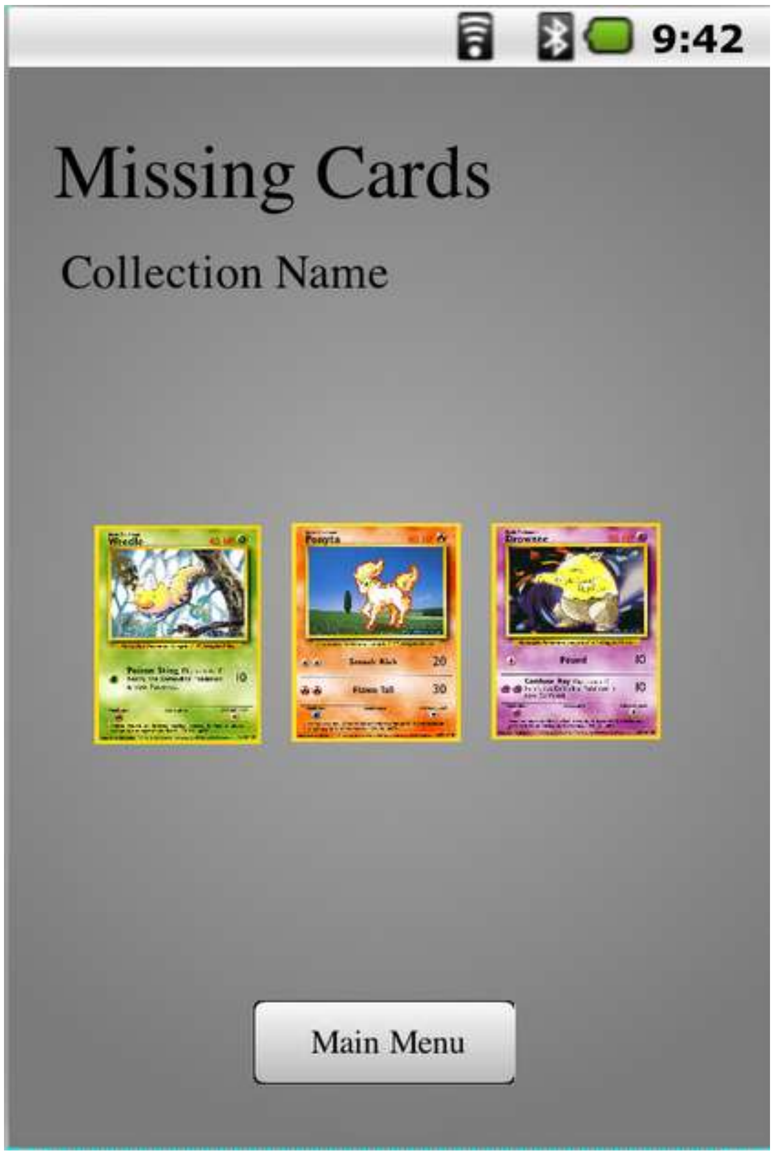
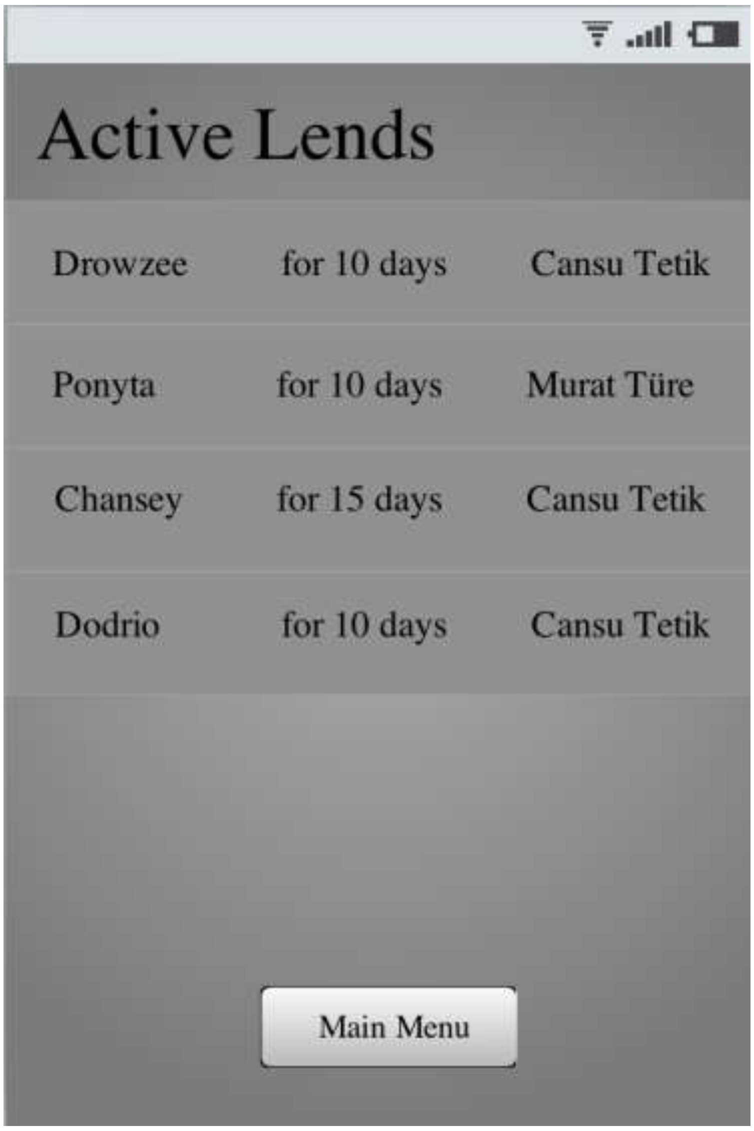


Figure 20 – Missing Items Page

From this page, users can see the missing items for a specific set.

6.2.10. Active Lends Page



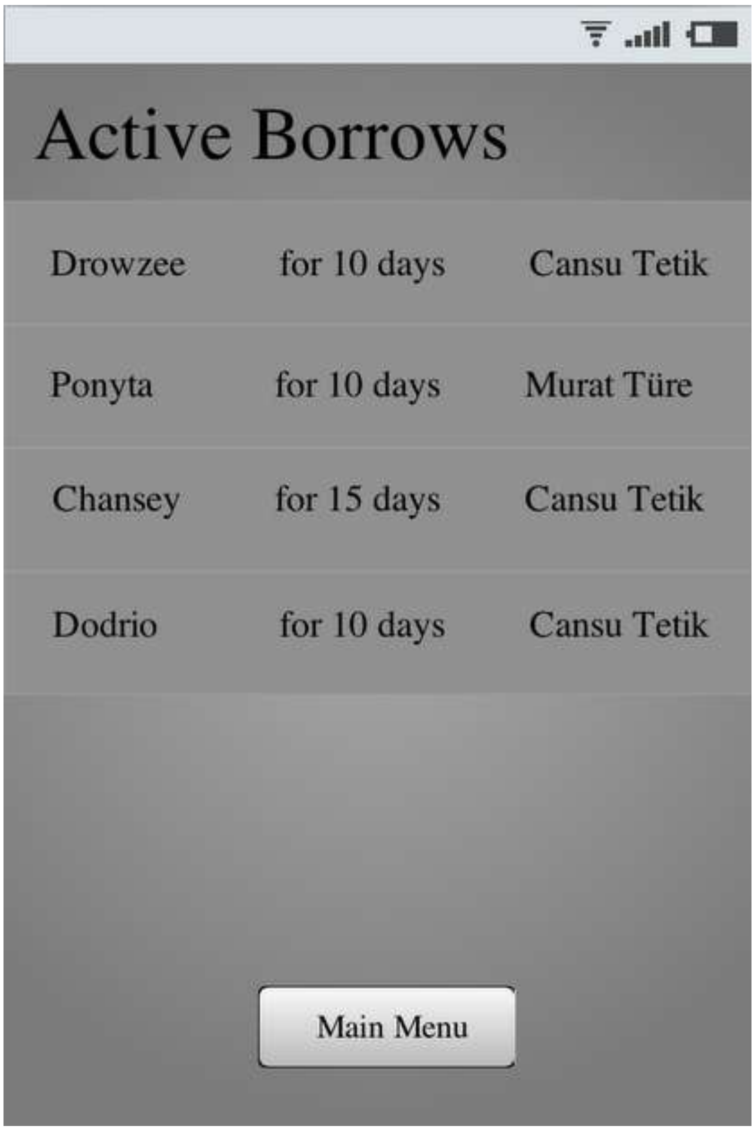
Active Lends

Drowzee	for 10 days	Cansu Tetik
Ponyta	for 10 days	Murat Türe
Chansey	for 15 days	Cansu Tetik
Dodrio	for 10 days	Cansu Tetik

Figure 21 – Active Lends Page

In Active Lends Page, the user is able to see which cards are currently lent to whom along with the remaining days.

6.2.11. Active Borrows Page



Drowzee	for 10 days	Cansu Tetik
Ponyta	for 10 days	Murat Türe
Chansey	for 15 days	Cansu Tetik
Dodrio	for 10 days	Cansu Tetik

Figure 22 – Active Borrows Page

In Active Borrows Page, the user is able to see which cards are currently borrowed from which collector along with the remaining days.

6.3.Screen Objects and Actions

Page	***Button***	***Action***
Login / Register Page	Login	Directs to Main Page when valid user name and password are entered
	Register	Directs to Register Page
Register Page	Register	Redirects to Login / Register Page when proper user name, password and e-mail are entered
Main Page	Search	Searches for the string provided by the user in the database and shows the results. If selected result is a ‘collector’ then directs to that collector’s Collection Page, otherwise directs to the chosen item’s Item Page
	Scan Page	Directs to Scan Page
	Accept	Accepts the borrow request, if there exists any
	Reject	Rejects the borrow request, if there exists any
	Collection Page	Directs to the user’s Collection Page
	Logout	Logs out from the application
Scan Page	Scan Picture	Scans the image and searches for it in the database, if there’s a match directs to the Item Page.

Collection Page	Main Page	Redirects to Main Page
	Tap an Item	Directs to that item’s Item Page
Item Page	Add to Collection	Adds the item to user’s collection
	Main Menu	Redirects to Main Page
	Item Set	Directs to Item Set Page of related item
	Search for Owners	Searches for the owners of that item in the database, directs to Owners Page
Owners Page	Borrow Request	Sends ‘Borrow Request’ to the selected user for that item
Item Set Page	Tap an Item	Directs to that item’s Item Page

7. Requirements Matrix

	Components	5.1*	5.2*	5.3*	5.4*	5.5*	5.6.1*	5.6.2*	5.6.3*	5.6.4*
Requirements										
4.2.1*		X	X	X			X			
4.2.2*		X	X	X				X		
4.2.3*		X	X	X				X		
4.2.4*		X		X						
4.2.5*		X	X	X	X					
4.2.6*		X	X	X		X			X	
4.2.7*		X	X	X						X
4.2.8*		X	X	X						X
4.2.9*		X	X	X	X					
4.2.10*		X	X	X						X
4.2.11*		X	X	X						
4.2.12*		X	X	X	X					
4.2.13*		X		X						
4.2.14*		X	X	X						
4.2.15*		X	X	X						X
4.2.16*		X	X	X						
4.2.17*		X	X	X						
4.2.18*		X	X	X						X
4.2.19*		X	X	X						X
4.2.20*		X	X	X						
4.2.21*		X	X	X						
4.2.22*		X	X	X					X	