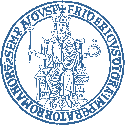
**Event**

**Manager**

**2017**

Mobile App Documentation

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Università degli Studi di Napoli Federico II – Corso di laurea di Scienze Informatiche

Software Engineering Project

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Group INGSW2017\_24**

Esposito Alessandro - N86001998

Florino Andrea - N86002104

**Index**

1. **Requirements Analysis** 
   1. Requirements Elicitation
   2. Requirements Analysis
      1. Use Case Diagram
      2. Use Case Description (Cockburn’s template)
      3. Class Diagram (Analysis)
      4. Sequence Diagram
      5. Statechart Diagram
      6. Activity Diagram
2. **Software Design**

2.1. System Architecture

2.2. Design Class Diagram

2.3. Design Sequence Diagram

1. **Testing plan**

3.1. System testing

3.2. JUnit testing

# **Requirements Analysis Document**

1.1. Requirements Elicitation

* **INTERVIEW n.1**

STAKEHOLDER: Prof. Sergio Di Martino

Control officers have mobile devices for accesses validation. The system provides to give and manage an account for each of them.

Each device, connected to a central server, validates a customer’s ticket by scanning its QR code.

Whenever the QR scanning succeeds, the application must show the linked event’s details

* **INTERVIEW n.2**

STAKEHOLDER: Prof. Sergio Di Martino

It is important for the service to be reliable and fast.

1.2 Requirement Analysis

1.2.1 Use case diagram

Android application



Controller

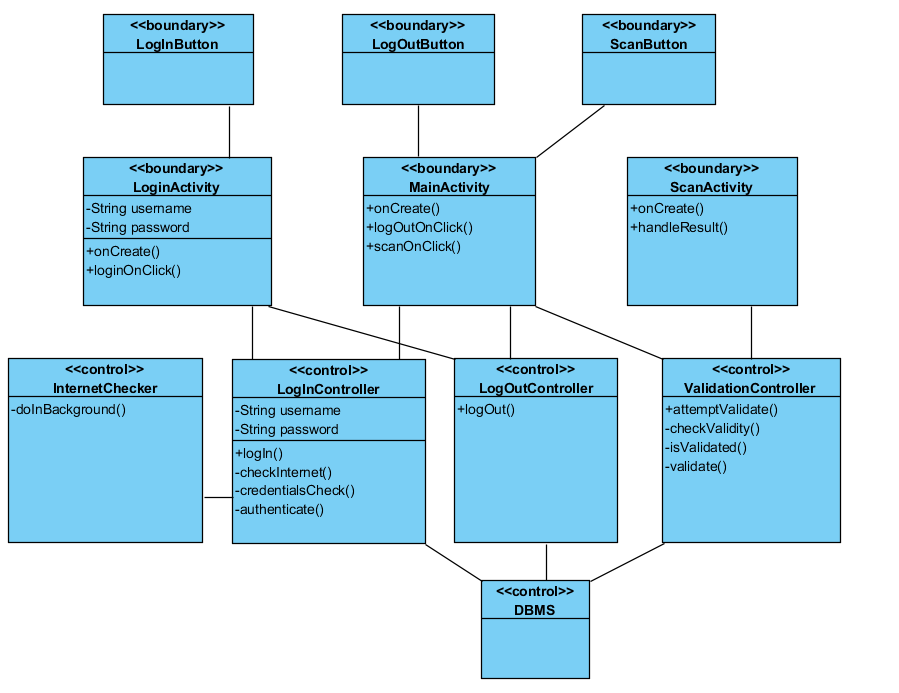
1.2.2 Use Case Description (Cockburn’s template)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| #1 | Log In |  |  |  |
| *Goal in context* | The controller accesses his account on the application | | | |
| *Success End Condition* | The controller succeeds to log in | | | |
| *Fail End Condition* | The controller fails to log in | | | |
| *Primary actor* | Controller | | | |
| *Trigger* | The controller presses the “Log In” button on the app | | | |
| DESCRIPTION | **STEP N°** | **CONTROLLER** | **SYSTEM** | |
|  | 1 | Writes username and password and presses Log In [LOGIN] |  | |
| 2 |  | Shows scan page [HOME] | |
| SUBVARIATION #1 | **STEP N°** | **CONTROLLER** | **SYSTEM** | |
|  | 1 | Fails to match user and password and presses Log In [LOGIN] |  | |
| 2 |  | Shows the same page but displays a “mismatch user-password” message [LOGIN] | |
| SUBVARIATION #2 | **STEP N°** | **CONTROLLER** | **SYSTEM** | |
|  | 1 | Presses Log In while there is no internet connection [LOGIN] |  | |
|  | 2 |  | Displays error message and does not attempt login [LOGIN] | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| #2 | Scan and validate | | |  | |
| *Goal in context* | The controller scans and validates a ticket | | | |
| *Preconditions* | The controller must be logged in | | | |
| *Success End Condition* | The ticket gets validated | | | |
| *Fail End Condition* | The ticket cannot be validated | | | |
| *Primary actor* | Controller | | | |
| *Trigger* | The controller presses “SCAN” button | | | |
| DESCRIPTION | **STEP N°** | **CONTROLLER** | **SYSTEM** | |
|  | 1 | Presses “SCAN” button [HOME] |  | |
| 2 |  | Shows scanning page [SCAN] | |
| 3 | Frames the QR-Code [SCAN] |  | |
| 4 |  | Validates the QR | |
| 5 |  | Shows a feedback (“QR OK”) and the event’s data [SCAN]] | |
| SUBVARIATION #1 | **STEP N°** | **CONTROLLER** | **SYSTEM** | |
|  | 4 |  | Fails to validate the QR | |
| 5 |  | Displays an error message [SCAN FAILURE] | |
| EXTENSION #1 | **STEP N°** | **CONTROLLER** | **SYSTEM** | |
|  | 3 | Presses “Back” button [SCAN] |  | |
| 4 |  | Shows home page [HOME] | |

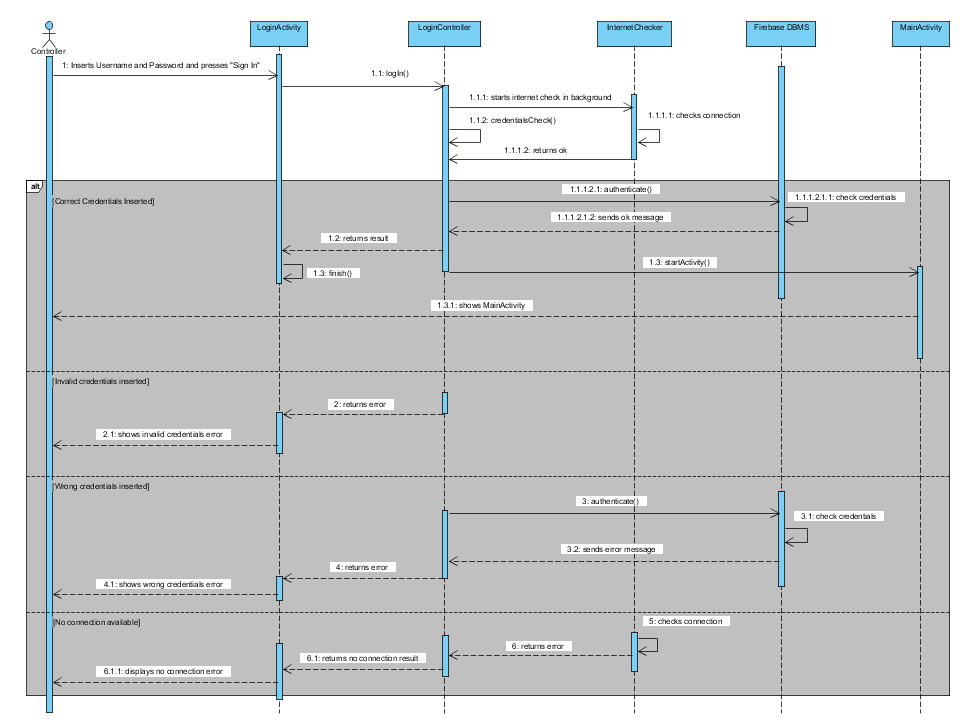
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| #3 | Log Out | | |  | |
| *Goal in context* | The controller logs out | | | |
| *Preconditions* | The controller must be logged in | | | |
| *Success End Condition* | The controller succeeds to log out | | | |
| *Primary actor* | Controller | | | |
| *Trigger* | The controller presses “Log Out” button | | | |
| DESCRIPTION | **STEP N°** | **CONTROLLER** | **SYSTEM** | |
|  | 1 | Presses “Log Out” button [HOME] |  | |
| 2 |  | Shows Log In page [LOGIN] | |

1.2.3 Class Diagram (Analysis)

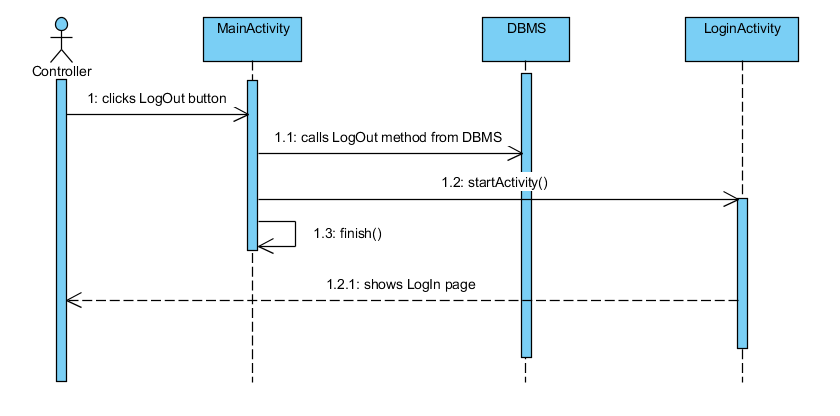


1.2.4 Sequence Diagram

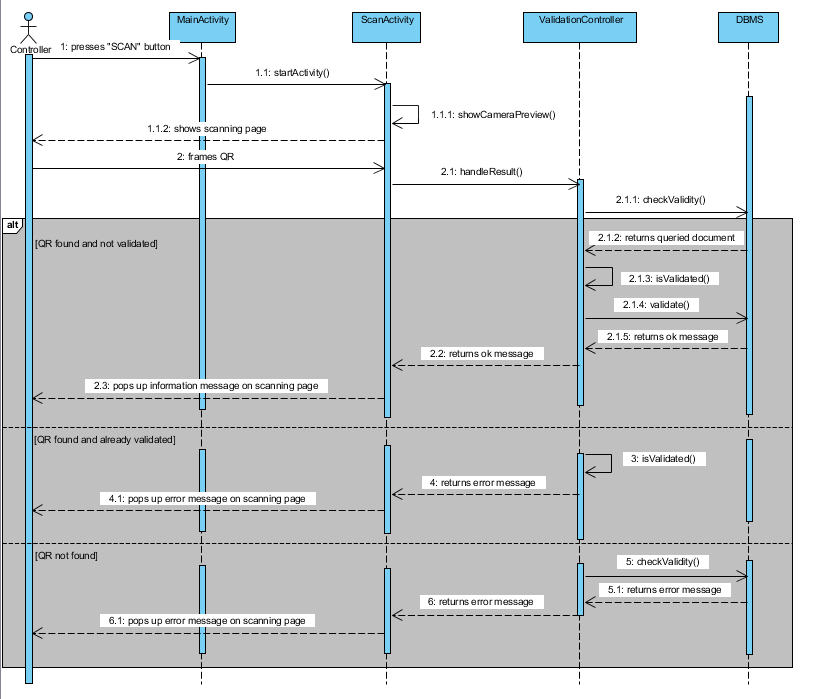
Use Case #1: Log In



Use Case #2: Log Out

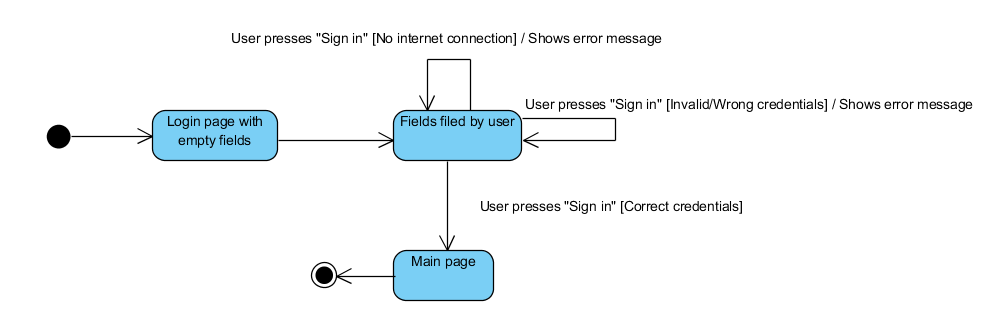


Use Case #3: Scan and Validate

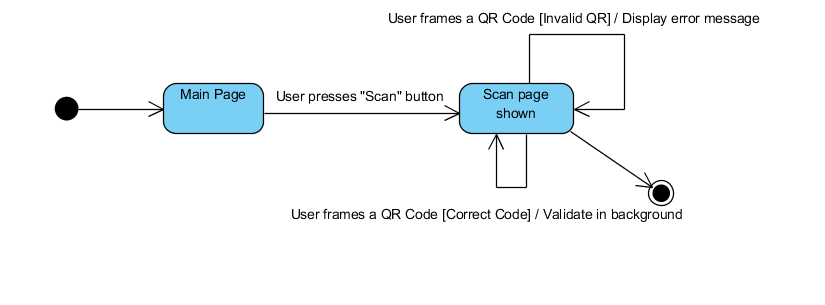


1.2.5 Statechart Diagram

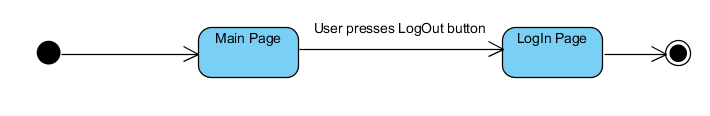
Statechart #1: Log In



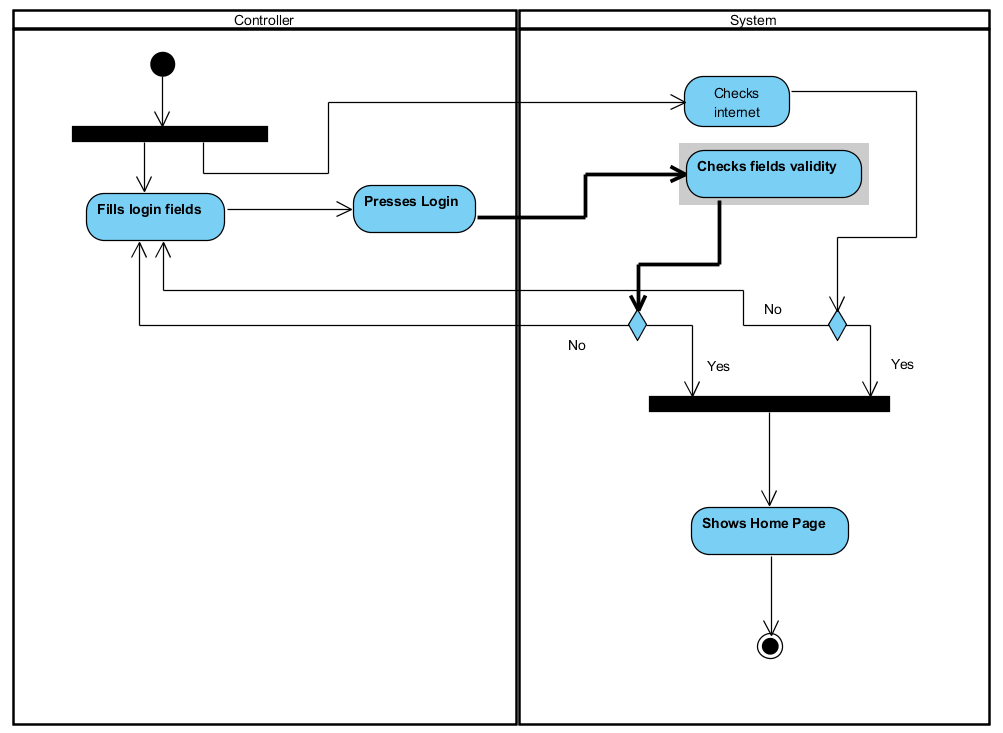
Statechart #2: Scan and Validate



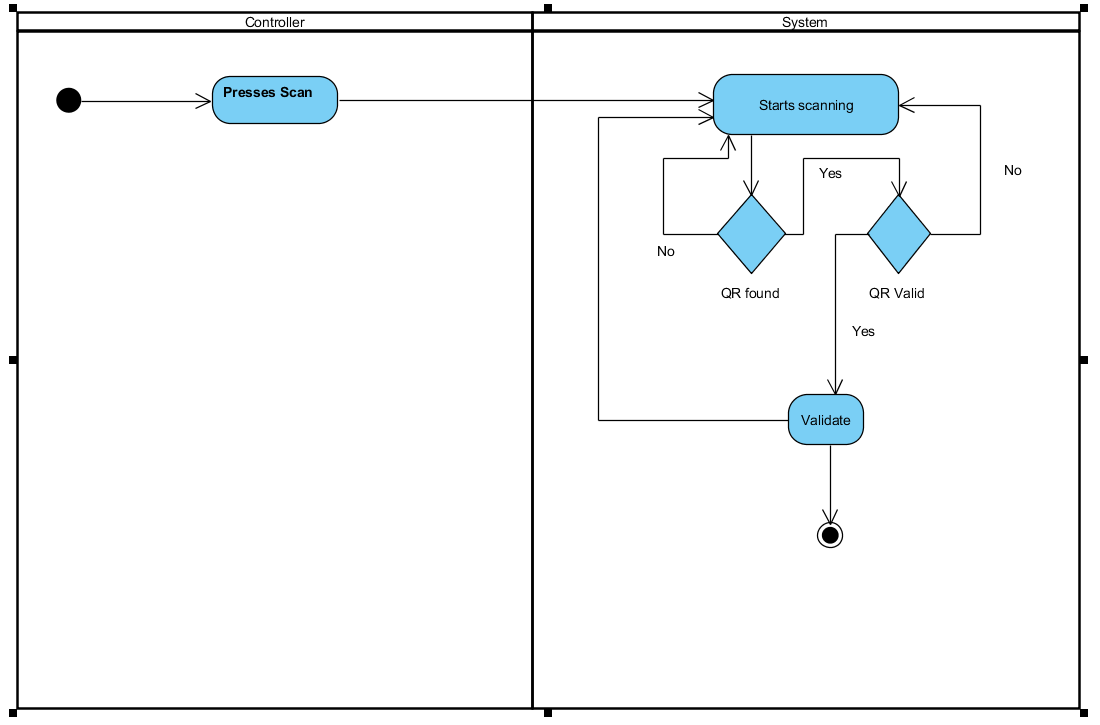
Statechart #3: Log Out



1.2.6 Activity Diagram

Activity Diagram #1: Log In

Activity Diagram #2: Scan and validate



# **Software Design**

* 1. System Architecture

This system uses a Client-Server model architecture. This application is meant to be interfaced with the Event Manager’s database, but for a demonstrative-only purpose the system is going to simulate it using Google’s online database Firebase.

This application also implements the open-source ZXing’s QR Code Scanner library.

The application is structured with the Model-View-Controller architectural pattern, where the control classes stands between the UI elements and the external database. The UI elements are provided by XML external files.

The controller classes do not directly access the database, such duty is assigned to an inner lever classes so that the software and the database are completely unlinked.

Each level only communicates with the below level, to keep architectural closure.

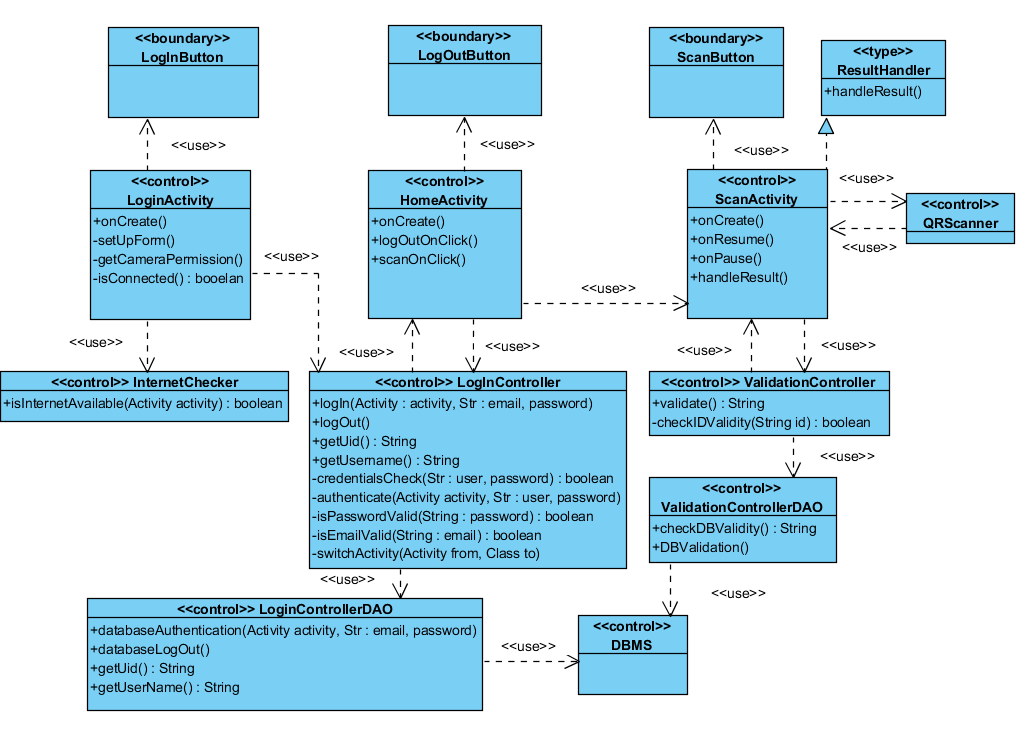
Control Layer

DBMS

Database Interfaces

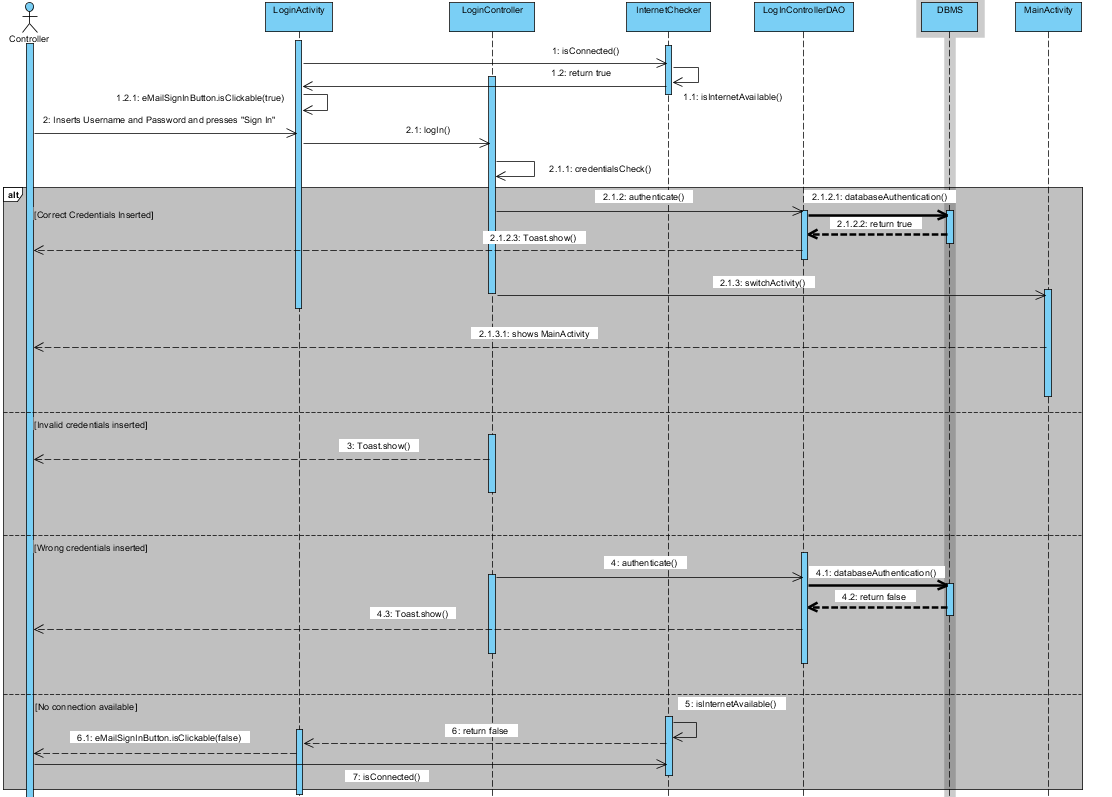
Interface Layer

* 1. Design Class Diagram

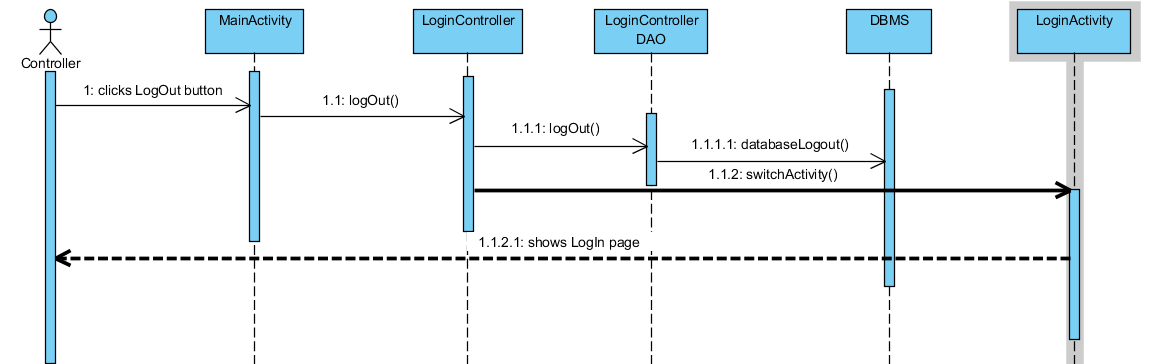


* 1. Design Sequence Diagram

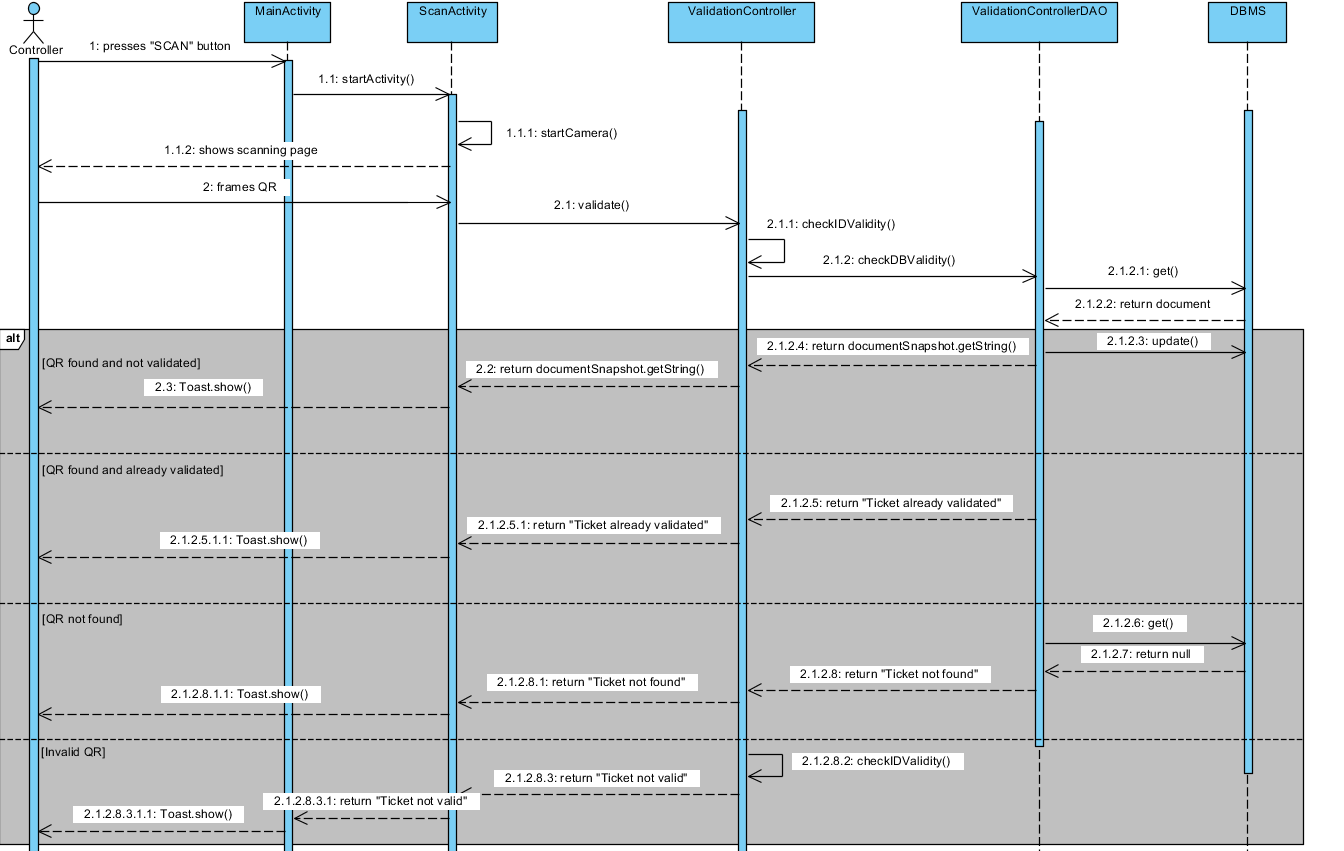
Use Case #1: Log In



Use Case #2: Log Out

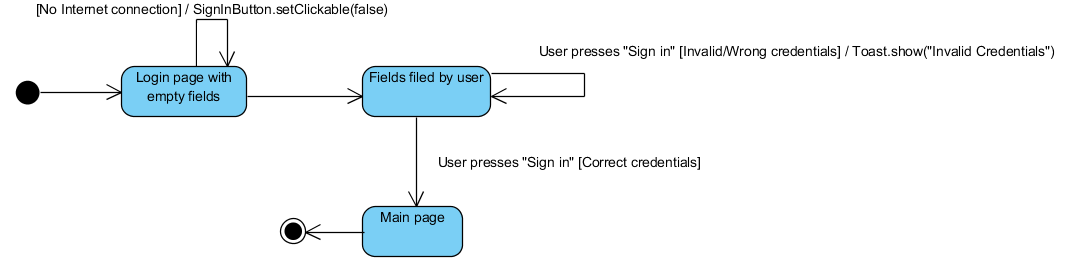


Use Case #3: Scan and Validate

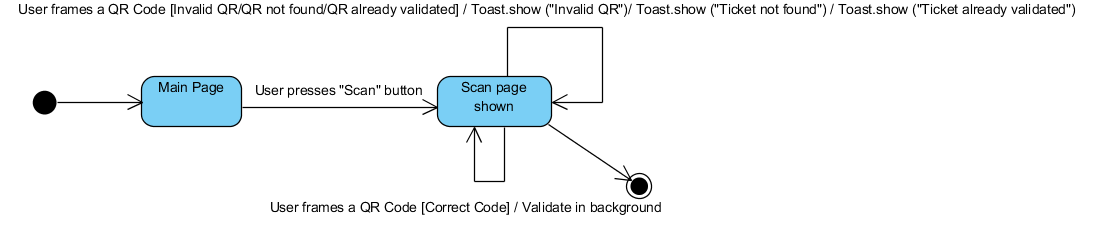


* 1. Design Statechart Diagram

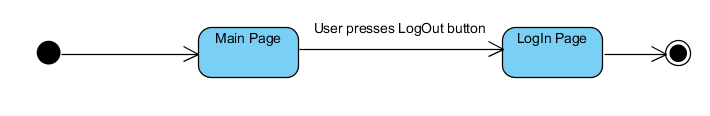
Statechart #1: Log In



Statechart #2: Scan and Validate



Statechart #3: Log Out



# **Testing Document**

* 1. System Testing

The following testing plan covers all user’s possible inputs in the application. The test will stress all the use cases (that can be found on the top of the document) with a full branch cover to make sure the application works properly with every input.

|  |  |  |
| --- | --- | --- |
| Test id | 1 | |
| Test name | Log in test | |
| Test description | Verify that the log in function works properly | |
| Inputs | **Result wanted** | **Result achieved** |
| Insert an e-mail and a password which have been registered in the database: [alex19@live.it](mailto:alex19@live.it) and alex123456789 | The user logs correctly in and reach the main page, which shows current user’s info | Correct |
| Insert an e-mail that has been registered in the database but with a wrong password: [alex19@live.it](mailto:alex19@live.it) and a12345 | The application does not log in and shows an error message “Invalid Credentials” | Correct |
| Insert an e-mail that has been registered in the database : [alex19@live.it](mailto:alex19@live.it) but leave the password field blank | The application does not log in and shows an error message on the password field “Invalid password” | Correct |
| Lave both field blank | The application does not log in and shows an error on both fields, showing also a message on email field: “this field is required” | Correct |
| Insert an e-mail which has not been registered in the database with a password that is: [alex1@live.it](mailto:alex1@live.it) , 123456789 | The application does not log in and shows an error message: “Invalid credentials” | Correct |
| Insert a non-valid format email: alex19@live-it | The application does not log in and shows an error message on the e-mail field: “Invalid e-mail format” | Correct |
| Open the app without internet connection | The Log In button cannot be pressed | Correct |
| NOTES | The user [alex19@live.it](mailto:alex19@live.it) with the password alex123456789 must be registered in the database | |
| Test id | 2 | |
| Test name | Scan and validate | |
| Test description | Verify that the scan and validate function works properly | |
| Inputs | **Result wanted** | **Result achieved** |
| Scan a valid QR code which represents the ID of a non-validated ticket in the database:  @em1 | The application validates the ticket and in the database the flag “isValidated” become true | Correct |
| Scan a valid QR code which does not correspond to any ticket in the database:  @em1234 | The application does not change the Boolean flag “isValidated” in the database and returns an error, which is shown with a Toast message “Ticket already validated” | Correct |
| Scan a non- valid QR code:  @1234, “alessandro”, “@eCiao” | The application shows an error message with a Toast: “invalid QR” without querying the database | Correct |
| NOTES | The ticket with id @em1 must be in the database | |

3.2 JUnit Testing

The following JUnit tests are focused on stressing the email and password checking during the log in phase.

package com.ingsw.myapplication;  
  
import org.junit.Test;  
import static org.junit.Assert.\*;  
  
  
public class CredentialCheckTest {  
  
  
 @Test  
 public void testIsPasswordValid(){  
  
 *assertTrue*(LogInController.*isPasswordValid*("a9jf932f2h398"));  
 *assertTrue*(LogInController.*isPasswordValid*("a1s9d9484"));  
 *assertTrue*(LogInController.*isPasswordValid*("a1234567"));  
  
 *assertFalse*(LogInController.*isPasswordValid*(""));  
 *assertFalse*(LogInController.*isPasswordValid*("213s"));  
 *assertFalse*(LogInController.*isPasswordValid*("asdsdgdgfgf"));  
 *assertFalse*(LogInController.*isPasswordValid*("aa11"));  
  
 }  
  
 @Test  
 public void testIsEmailValid(){  
  
 *assertTrue*(LogInController.*isEmailValid*("alex19@live.it"));  
 *assertTrue*(LogInController.*isEmailValid*("ciao@dominio.it"));  
 *assertTrue*(LogInController.*isEmailValid*("alex19@live.com"));  
  
 *assertFalse*(LogInController.*isEmailValid*("alex19Alive.it"));  
 *assertFalse*(LogInController.*isEmailValid*("alex19@liveit"));  
 *assertFalse*(LogInController.*isEmailValid*("alex19Aliveit"));  
 *assertFalse*(LogInController.*isEmailValid*(""));

}  
}