UNIVERSITY OF ZAGREB
FACULTY OF ORGANISATION AND INFORMATICS
V A R A Ž D I N

Viktor Lazar, vlazar@foi.hr
Martina Šestak, msestak2@foi.hr
Goran Vodomin, gvodomin@foi.hr
Matej Vuković, mvukovic2@foi.hr

BLUETOOTH LE SHOWCASE

TECHNICAL DOCUMENTATION FOR SOFTWARE ANALYSIS AND DEVELOPMENT PROJECT

Varaždin, 2015.

UNIVERSITY OF ZAGREB
FACULTY OF ORGANISATION AND INFORMATICS
V A R A Ž D I N

Team number: T01

Team name: Heisenbug

Team members:

Viktor Lazar, 0108063551
Martina Šestak, 0016091250
Goran Vodomin, 0016092445
Matej Vuković, 0016094754

BLUETOOTH LE SHOWCASE

TECHNICAL DOCUMENTATION FOR SOFTWARE ANALYSIS AND DEVELOPMENT PROJECT

Mentors:
Dr.sc. Zlatko Stapić
Ivan Švogor, mag.inf.
Evolaris mentors:
Christian Adelsberger
Hermann Moser
Martin Schumann

Varaždin, January 2015.

Table of Contents

1. Introduction

1.1 Purpose of this document

The purpose of this document is to give technical information about SeierFriendApp design and implementation.

1.2 Intended Audience

The intended audiences are:

- Course mentors, to analyze the design and implementation of SeierFriendApp.
- Evolaris mentors, to analyze the design and implementation of SeierFriendApp.
- Team Heisenbug members.
- Future developers to extend or use some ideas of this project.

1.3 Scope

This document will describe the design and some technical issues of SeierFriendApp project.

1.4 Definitions and acronyms

1.4.1 Definitions

Keyword	Definitions
Beacon device	Hardware device that can be detected by mobile devices.

1.4.2 Acronyms and abbreviations

Acronym or abbreviation	Definitions
API	Application Programming Interface
BLE	Bluetooth Low Energy

2. General overview

2.1 Technologies used

SeierFriendApp is a mobile application receiving its data from web services.
While working on this application we used various technologies and tools for different activities (development, design, versioning etc.).

Modeling tools (conceptual model, activity and class diagram, system architecture)

- POP (http://popapp.in/) for conceptual modeling
- Creately to model system architecture
- Visual Paradigm CE to create activity diagrams

Version control system

- Github repository with all project materials and application code is available at https://github.com/MartinaSestak/AiR_BLE

Web service and database

- Postman, SQLite local database

Application development tools

- Eclipse Luna IDE, Intellij IDEA 14.0.1 Ultimate Edition (minimal Android version 4.3 (API 18), since earlier versions don't support BLE technology), Genymotion

2.2 General functioning

SeierFriendApp mobile application has the following functionalities:

- User is notified on his mobile device when in range of beacons
- If the user clicks on the displayed notification, SeierFriendApp starts with a login screen. User can’t sign in unless he enters both his username and password.
- If he doesn’t have an account, he can register on the web application by clicking the Register button.
- On his first sign in, user enters Tag ID from his NFC card.
- User receives new points upon first daily check in.
- User can see how many points he has and his Friend status.
- User can see his personal information (firstname, lastname, email, last checkin…).
- User can see a list of all vouchers available in the system.

2.3 Error handling

As the architecture is built upon different layers, exception handling was done on each layer separately. The presentation layer displays a proper message to SeierFriendApp users.

3. User requirements specification

3.1. Introduction

3.1.1. Objectives

This is the User Requirements Specification for Bluetooth LE Showcase project, for use by Evolaris GmbH, team members and project mentors. In this section we will be determining the project’s scope, user requirements that need to be satisfied in our mobile application and describing our task that was assigned to us by Evolaris’ mentors.

3.1.2. History

Evolaris GmbH already developed the entire system and installed terminals inside The Shopping City Seiersberg. New users register online or at a terminal and receive a card with NFC tag. When they arrive at the shopping center, they need to sign in at a terminal in order to gain shopping benefits. After that they can see their “loyality” points and vouchers status and print vouchers. Our assignment is to try to implement the same functionality using Bluetooth LE technology by building a mobile application.

3.1.3. Scope

While doing this project we are expected to build a mobile application that will start when the terminal recognizes the user with Bluetooth LE turned on in its range. The user gets a notification signaling his device is recognized and receiving beacons and after that he can start his application by signing in. A signed in user can see how many points and what kind of vouchers he has and receive additional points or vouchers each time he is inside the shopping center.

3.2. Organisational / Functional Areas Affected

3.2.1. Assumptions

The application that we build will periodically need to be tested in real environment (The Shopping City Seiersberg) and we will need access to customers' database and server instance using the appropriate API.

3.3. Requirements

All requirements are defined in point form and are rated either Mandatory (M) or Highly Desirable (HD) or Desirable (D), dependent on business need and University Policy.

3.3.1. Functional Requirements

3.3.1.1. Common Features

Requirement	Preference
1.1.1.1 User sign in	M
1.1.1.2 User can see his points status and receive new ones	M
1.1.1.3 User can see his points history	D
1.1.1.4 User can see his personal information	M
1.1.1.5 User can see available vouchers	D

3.3.1.2. Reporting

Requirement	Preference
1.1.1.6 Project documentation	M
1.1.1.7 Notes from SCRUM meetings	M

3.3.2. Production Requirements

3.3.2.1. Hardware

Requirement	Preference
1.1.1.8 Mobile device with BLE support	M
1.1.1.9 Beacon device	HD

3.3.2.2. Software

Requirement	Preference
1.1.1.10 Min. Android API 18	M

3.3.3. Development Requirements

3.3.3.1. Hardware

Requirement	Preference
1.1.1.11 Mobile device with BLE support	M
1.1.1.12 Beacon device	HD

3.3.3.2. Software

Requirement	Preference
1.1.1.13 Intellij IDEA 14.0.1 Ultimate Edition	M
1.1.1.14 Holo Circular ProgressBar Android library	M
1.1.1.15 Evolaris API	M
1.1.1.16 ActiveAndroid library	M

3. Technical requirements

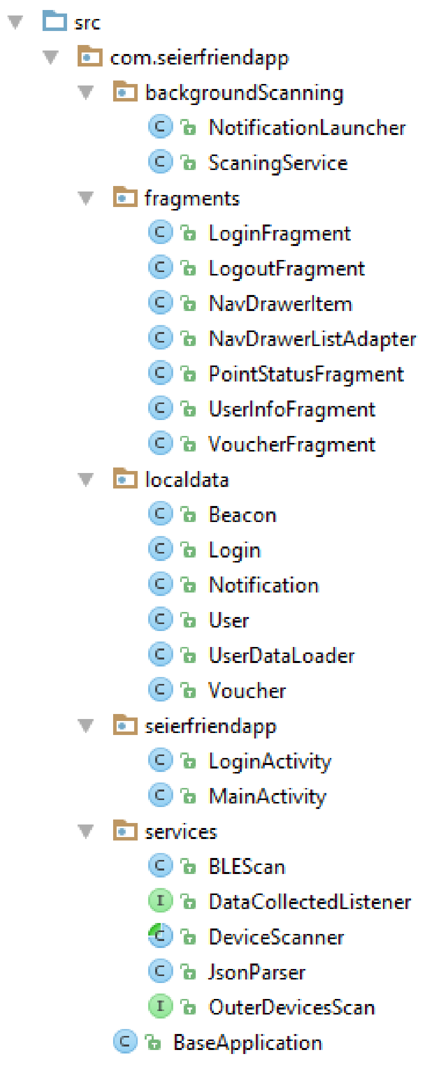
3.1 Evolaris API

Evolaris provided us with its own API and its methods for executing activities such as user registration, check in, fetching voucher and user information etc.

4. SeierFriendApp Source structure

There are several directories in *hr.foi.seierfriendapp* package, each one of them containing layer components; application data layer is implemented in classes and interfaces located in *backgroundScanning*, *localdata* and *services* directories, business layer is implemented in classes located in *fragments* directory, while application presentation layer is implemented in activity classes located in *seierfriendapp* directory.

Application source code specification is automatically generated using Javadoc and is available in our project Github repository.



```
src
  com.seierfriendapp
    backgroundScanning
      NotificationLauncher
      ScaningService
    fragments
      LoginFragment
      LogoutFragment
      NavDrawerItem
      NavDrawerListAdapter
      PointStatusFragment
      UserInfoFragment
      VoucherFragment
    localdata
      Beacon
      Login
      Notification
      User
      UserDataLoader
      Voucher
    seierfriendapp
      LoginActivity
      MainActivity
    services
      BLEScan
      DataCollectedListener
      DeviceScanner
      JsonParser
      OuterDevicesScan
    BaseApplication
```

Figure 1. Project source tree

5. Architecture

Architectural design consists of conceptual application model, system structure, application architecture including class diagrams.

ER diagram will not be included in this documentation because of the risk of sensitive data exposure.

5.1 Conceptual application model

The conceptual model for our application was first described and created with Pop tool and is available online. This model defines 5 possible activities (user stories) which a user can access. They are connected in the following way:

1. A user starts the application, enters his credentials in a login form and presses the Sign in button. If this is his first sign in, he needs to enter Tag ID from his NFC card in order to receive an authorization token that he will later use to receive points, else the main activity is displayed.
2. If he doesn't have an account, he can press the Register online link that will redirect him to the web application where he can register.
3. If he signs in successfully the next activity enables him to see how many points he has and what is his "Friend of Seiersberg" status.
4. By pressing the Menu icon in the top left corner, a side menu appears. It contains his personal information, point status and available vouchers list.
5. If the user selects *Personal Info* option, he can his personal information (firstname, lastname, email, number of points etc.).
6. If the user selects the *Point status* option, he is sent back to the Point status fragment described in step 3.
7. If the user selects the *Vouchers* option, a new fragment appears, showing him a list of all vouchers that are available to him. Each voucher is shown as an image with a name.

5.2 System architecture

The entire system architecture behind the SeierFriendApp application includes several participants. Their communication (exchanged data) is described in Figure 4. The entire architecture is divided into three areas displayed inside dotted shapes in different colors.

Elements inside the green shape are property of Evolaris GmbH (terminal, web services, database and web application), elements inside the red shape are property of Team Heisenbug, while the Bluetooth LE device, i.e. beacon device, is a property of both Evolaris GmbH and Team Heisenbug.

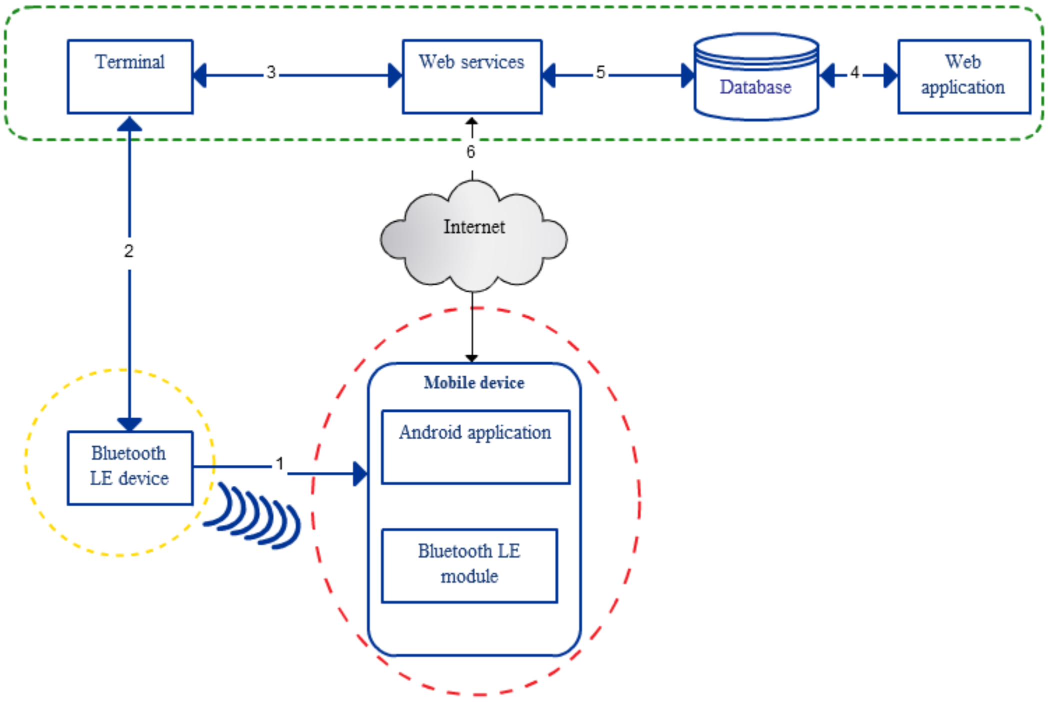


Figure 2: System architecture

Table 1. Communication between architectural components

Communication between architectural components	
1	Bluetooth LE device sends beacons to mobile device. A beacon contains device's MAC address.
3	Terminal and web services exchange user data, data about new vouchers etc.
4	Web application gets data from database, and changes it depending on administrative needs.
5	On request web services communicate with database to fetch or change data, this could be data about vouchers, user data, business logic data, etc.
6	Mobile device on user request contacts web services for data. This can be fetching data (voucher details) that user needs to see, or sending user data for processing (getting visit points).

5.3 SeierFriendApp application architecture

If we look more closely at the Android application's infrastructure, it can be described as follows:

- Presentation layer consists of UI components (view controls used in forms) and Presentation logic components (event listeners that execute part of a code when triggered).
- Presentation layer communicates with Business layer, which has some Business logic components used to implement business logic (for instance, to check whether the user signed in more than once in one day, which vouchers are disabled to him etc.)
- Business layer communicates with Data layer, which contains components for working with the local database, accessing physical device (BLE device, NFC card or something else) and exchanging data with the corresponding module (in this case BLE module) and web services access components for exchanging data with services.
- BLE beacon device and web services represent a supporting infrastructure for the application.

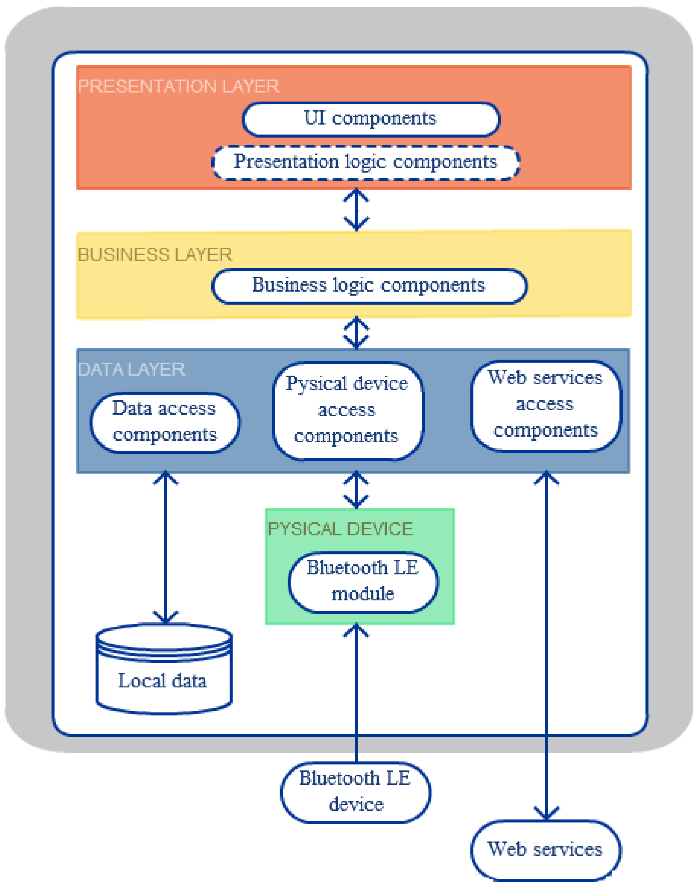


Figure 3. Mobile application architecture

Table 2. shows the format of the communication between layer shown in Figure 3.

Table 2. Communication form between application layers

Bluetooth LE module	
Input data	**Output data**
MAC addresses (BLE device)	Broadcast intent
Data layer	
Input data	**Output data**
Entries from tables (local database)	New/updated database entries (local database)
MAC addresses and Broadcast intent (BLE module)	Context for launching notification (Business layer)
Objects (web service)	GET/POST requests (web service)

Business layer	
Input data	**Output data**
Objects (Web service access components)	Parsed JSON objects saved as database table entries (local database)
Presentation layer	
Input data	**Output data**
Database entries (Data layer)	Values for layout views

5.3.1.1 BLE scanning module logic

On application startup, application checks if Bluetooth is enabled. If that's not the case, application enables it and gets MAC addresses of beacon devices from web service and stores them in local database. After that application launches background scanning service. Background service gets previously stored MAC addresses via interface, launches BLE scanning module providing it with MAC addresses.

Besides that, it also implements broadcast receiver which receives broadcast intents from scanning module (class BLEScan). BLE scanning module has few methods, some of which work with the Bluetooth adapter. They are continuously scanning and if there is a MAC address match they make broadcast intent. The scanning service receives that intent and calls a class (class NotificationLauncher) that handles intent. After that NotificationLauncher checks if notification were sent and if there weren't, it creates notification.

If the user quits application ScanningService continues running in the background scanning for beacon devices. When the user clicks the notification NotificationLauncher launches Login Activity with extra data that indicates if Check method in web service needs to be called. On the other hand, if the user cancels notification, if there is a match, notification is launched again.

5.3.1.2 Working with web services and local database

Application communicates with a backend server in order to ensure that all necessary data about the user and his benefits are available. This communication is achieved through API provided by Evolaris. In the domain of our Android application this communication is mainly based on a class called JsonParser which is included in services package. The most important feature of JsonParser class is its nested class called WebServiceDataCollection. Here is the whole logic that stands behind data retrieval from the backend server. Depending on the type of the HTTP request (GET or POST) this class parses forwarded parameters and calls backend server's API. After that data returned in the server's response are validated and converted to JSON object. Interface which indicates if data is collected or not is also ensured.

In the application, data is stored in a local SQLite database. Database is achieved through ActiveAndroid and all work and transactions related to database is made through it. ActiveAndroid is an object relational mapper (ORM) which provides easy and fast way to work with local database. At this Sprint of development local database consists of five tables, i.e. classes. After user login and data synchronization, data is stored in the local database. This action is handled in UserDataLoader class. Other functionalities related to fetching and storing data will be added in the next Sprint according to our development methodology.

The application will be built in a modular way enabling the system to work properly no matter which technology is used in Data layer, whether physical device access components are communicating with a BLE module, WIFI, NFC or any other module.

5.4. SeierFriendApp class diagram

Class diagram representing our application code is generated from Intellij IDE. From this diagram can be seen that there are two main logical parts. First of them refers to interaction with user and consists of two activities, *LoginActivity* and *MainActivity*. Both of them are interacting with ShoppingCity Seiersberg web application. This interaction is provided by *JsonParser* class and its interface *DataCollectedListener*. Collected data is saved into local database with *DbDataSaver* which contains two methods for storing data into *User* and *Voucher* entities. Views for user (i.e. application screens) are implemented with *Fragments*.

Second part contains classes which are responsible for scanning signals transmitted by BLE beacons and informing user about occurred events. *ScanningService* class represent service that runs in background, scan for devices and receives broadcast when device is found. Currently it works with BLE beacons, but *BLEScan* class can be replaced with another class which implements scanning for other device types (e.g. NFC module).

Beacon class is a database entity used to store information about valid BLE beacons, to identify valid BLE beacons and filter them from various other BLE devices. *ScanningService* class uses *NotificationLauncher* class to inform user about events that have occurred because of BLE beacon is scanned. Every notification is stored in *Notification* entity to provide notification management and ensure data for later interpretation of scanning results.

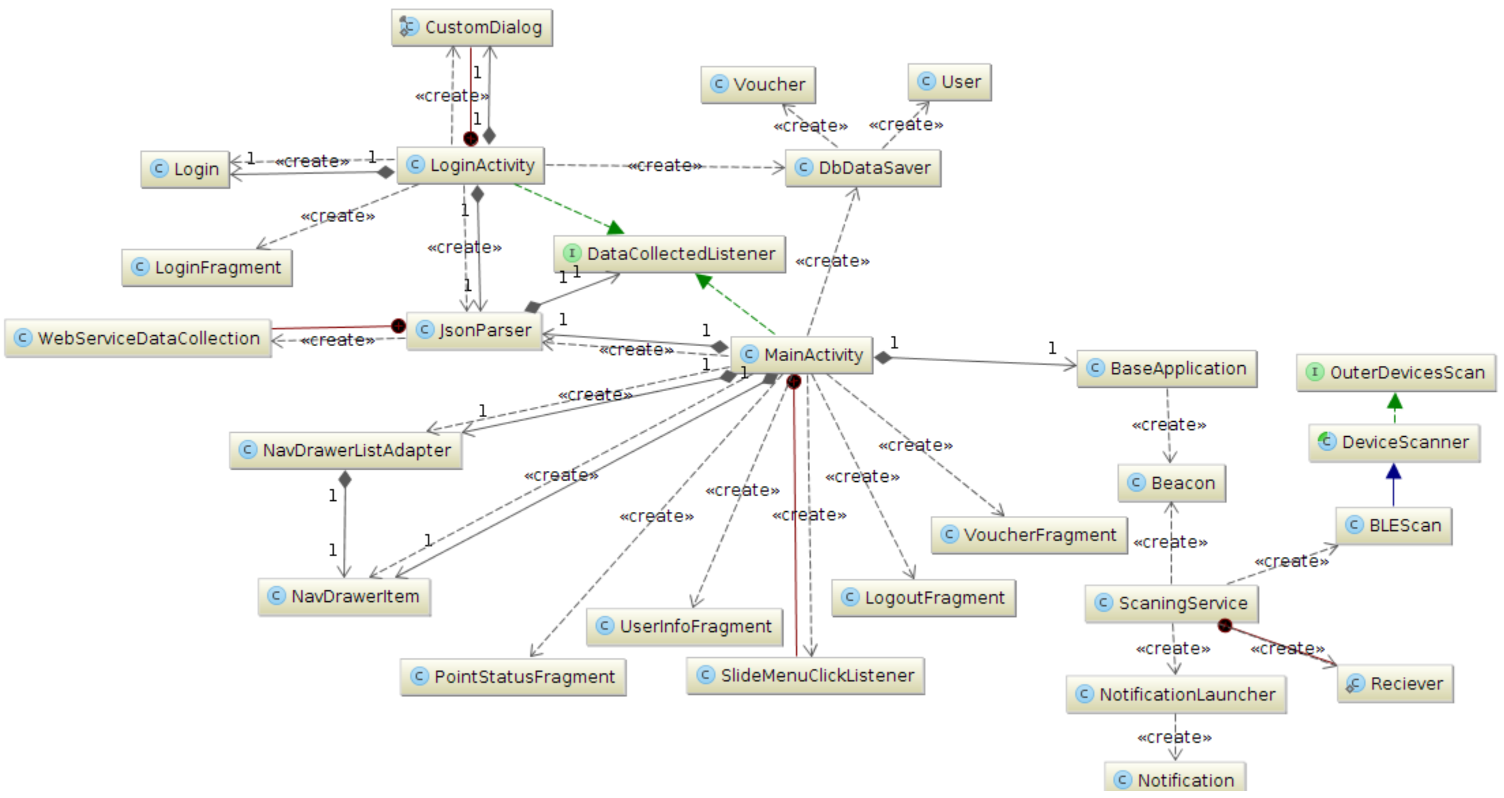


Figure 4. SeierFriendApp class diagram

6 Application testing

6.1 Acceptance test

Application user stories are according to Scrum methodology defined and discussed in meetings with the Product Owner and development team (in our case development team representative). Also, for each user story there must be a list of acceptance criteria which define boundaries of a given user story and are used to confirm if a user story is completed or progressing in the right way. Furthermore, an acceptance test represents a formal description of the behavior of the software product expressed through different user stories. The result of this test can be either a pass (software product meets defined acceptance criteria) or a failure (software product does not satisfy acceptance criteria and cannot be accepted as completed by Product owner).

For our course's requirements, only one acceptance test was created to check user story called *User login* (Table 3).

Table 3. Acceptance test for "User login " user story

Project Identification		
Project Name	**Project Number**	**Date Created**
Bluetooth LE Showcase	01	27.01.2015.
Project Sponsor	**Product Owner**	
Evolaris GmbH	Martin Schumann and Evolaris' mentors	
Project mentors	**Created by**	
Prof.dr.sc. Zlatko Stapić Igor Švogor, mag.inf.	Team Heisenbug	

ACCEPTANCE CRITERIA TEST MATRIX

User story: User login

Description: **As a user I want to be able to login with my current credentials (my email and my password), so I can start using the application.**

		Critical		Test Results		
Number	**Acceptance Criterion Description**	**Yes**	**No**	**Accept**	**Reject**	**Comments**
1	User clicks SeierFriendApp notification which displays application login form.		☐	☐		
2	User enters his correct credentials and logs into the SeierFriendApp application.	☐		☐		
3	User can’t login without entering his username and password.	☐		☐		
4	User can’t login if he doesn’t have a “Friend of Seiersberg” account	☐		☐		
5	User must enter Tag ID from his NFC card on his first application login.	☐		☐		
6	After successful login, user can start using the application.	☐		☐		