Android Application Development Documentation

Application: Auto Check Attendance

1. **Specification of the project:**

Our group aims to create an application that combines Android application and Web application to simplify the checking attendance process of students with the help of beacon technology.

**2. Initial planning:**

At first, we started planning to set up a server to return information to the Android app by a list of APIs, in the same time we planned to use web application as a simplified tool if there is any part in project that is generally more convenient to use a web application to process data. Secondly, we planned to code the Android application and design the UI to get the information stored in the beacon, in the same time interacting with the built server.

**3. Implementation:**

The project requires understanding of the beacon capabilities, specifically what information it can contain and pass the information to the web application to process through the Android application. The web application contains more information about classroom numbers etc., we found that it is generally more convenient for a user to use web interface to input long and complicated data into our application for processing such as parameters for the beacon about classroom info etc.

The initial project phase started with meeting and sharing of the projection documentation on 24 March 2016. We communicate daily and meet on weekly basis to discuss further and in more details technical challenges of the project and how to solve those challenges.

**4. User Story:**

The user story helps us to understand how the application should function initially and later on we build the Android application and the web application according.

* A student went to a class with his phone
* The student open the app, login
* The app auto detect the beacon in the class (and maybe some other beacons nearby)
* List of beacon(s) is displayed with class name(s) and some other properties
* Student will pick a class as checking attendance
* During the class time, the app records how much time the student has spent.
* After class finish, the student leaves the room, the app auto detect this event.
* All data will be sent to the server and saved in the database.
* A teacher goes to the admin panel, and can list all the class which he has.
* The teacher selects one class, and starts seeing who has been in the class by date.
* The teacher can also see how much time a student spent in the class
* The teacher can also so do some filter (date, name...) to summary
* An admin goes to panel, and can be able to set beacon data of the class (done in the Android app).

**5. Main APi:**

The app uses token for authentication. After logging in, the token will be saved and attached to the header of request for every api.

Link to API: http://5.101.107.114:8080/api/

List of APIs:

* /register  
  Params: firstName, lastName, username (email), password, type (student, teacher)  
  Return: userId, firstName, lastName, username, type, token
* /login  
  Params: username, password  
  Return: userId, firstName, lastName, username, type, token
* /attendClass  
  Params: classId  
  Return: success
* /leaveClass  
  Params: classId  
  Return: success
* /addCourse (only using for teacher, add new course)  
  Params: name, code (course code), description, credit (int), startDate, endDate (format: 2016-04-20 or 2016-4-20)  
  Return: courseId, name, code, description, credit, startDate, endDate
* /getCourseInfo  
  Params: courseId  
  Return: course Info
* /addStudentToCourse (only using for teacher, and dummy data purpose, app doesn’t need to implement this)  
  Params: courseId, studentId (userId of the student)  
  Return: success
* /RemoveStudentFromCourse (implement for teacher)  
  Params: courseId, studentId  
  Return: success
* /addRoom (no need implement)  
  Params: name, code (eg: Game Studio B303, 12345 is beacon code)  
  Return: room info
* /deleteRoom (implement for teacher)  
  Params: roomId
* /getListRoom (get all available room, implement for teacher)  
  Params: no params
* /addClass (implement for teacher)  
  Params: courseId, roomId, startTime, endTime (format 2016-04-20 10:00:00)  
  Return: class info
* /deleteClass (implement for teacher)  
  Params: classId
* /editClassInfo (implement for teacher)  
  Params: classId, courseId, roomId, startTime, endTime (format 2016-04-20 10:00:00)  
  Return: class info
* /getListClass (implement for teacher, get list class, all class available)  
  Params: page (for pagination, only return 12 items per page, default is 1)  
  Return: array of class info
* /getMyClass (get list class which user registered to the courses)  
  Params: page (for pagination, only return 12 items per page, default is 1)  
  Return: array of class info
* /getClassByBeacon (send beacon code as array to get the list of class available nearby which user registered)  
  Params: beacons (code1, code2...)  
  Return: list of class which user register to the courses based on beacon
* /getMyCoures  
  Params: none  
  Return: list of courses which user registered
* /getMyAttendances  
  Params: none  
  Return list of class belong to user with a properties attended = 1 or 0

**6. Sample detail data:**

One detail technical use case could be a student opens the application to see some beacons around, the application shows beacon codes which are corresponding with room numbers. The student checks time to know what courses are happening, the application shows data of room name, course name and time to the student/user.

We provide a sample of data returned in JSON format from our server when a user is using application and is requesting information from the server, the Android application will handle the JSON data to get relevant information out.

{

"success":true,

"data":[

{

"classId":2,

"startTime":"2016-04-26T07:00:00.000Z",

"endTime":"2016-04-26T09:00:00.000Z",

"roomId":1,

"roomCode":"b303",

"roomName":"Game",

"courseId":1,

"courseName":"Android Application Development",

"courseCode":"TI00AB76-3012",

"description":"Contents - Google Android application architecture - Android application framework and development tools - Application resources - User interface - 2D and 3D graphics - Services and multiprocessing - Data storage and SQLite - Content providers and broadcast receivers - Google Maps in Android - Android Web applications - Application publishing - A project ",

"credit":5,

"startDate":"2016-01-10T22:00:00.000Z",

"endDate":"2016-05-14T21:00:00.000Z"

}

]

}

**7. Screenshots:**

In this section, we provide some screenshots of our application and some detail explanation about the process of using the application.

Link to source code:

<https://github.com/BisquitBehemoth/BeaconApp> student view

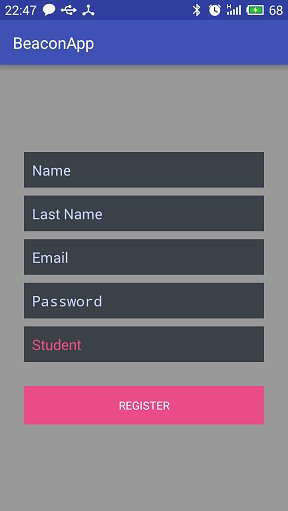
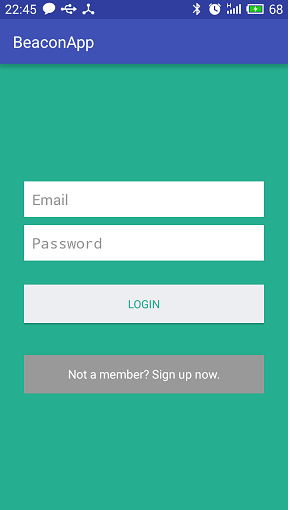


Figure 1. Figure 2.

Figure 1 shows the logging in UI for a user, figure 2 shows registering interface for a user (can be a student/teacher).

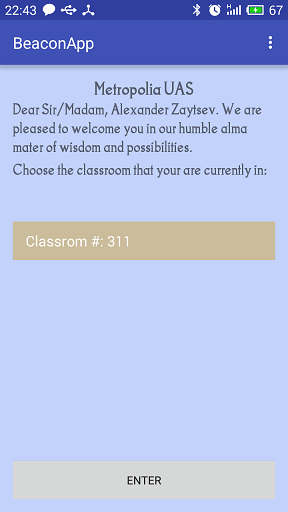
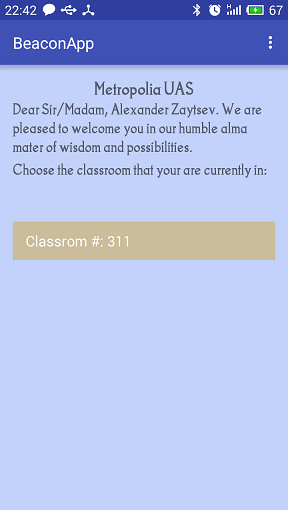


Figure 3. Figure 4.

After logging process is done a user will see the UI as in figure 3. A couple of textViews and a list of beacons that are closeby (the application looks only for specific beacons, with major value=1808). When the user clicks on one of classrooms from the list, a button appears as in figure 4.

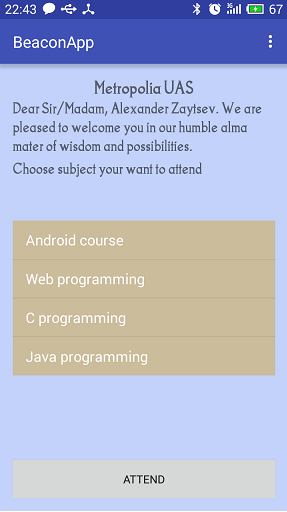


Figure 5.

When a user clicks enter the list of classes associated with the class will appear. Now it is dummy data figure 5( the API /getClassByBeacon was implemented with params: beacons=311, beacons=0, returns =success. Unfortunately we did not have enough time to do the adding courses functionality. Afterward, the user chooses course from the list and clicks attend,

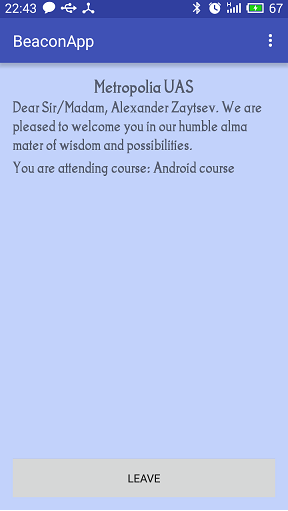


Figure 6.

Finally the user sees an UI similar to figure 6. After that s/he can leave manually by pressing the button leave or just leave the area of the beacon. In any case leaveReuest will be sent to our server



Figure 7: Sample teacher view.

Figure 7 shows checking status of students in the class in real time. The web application automatically updates the status of student who is attending or already left the class and also the duration of student spending in the class. The application only shows the classes which are happening at the current time.

Teacher view can be seen by visiting the URL : <http://5.101.107.114/beacon/app/>

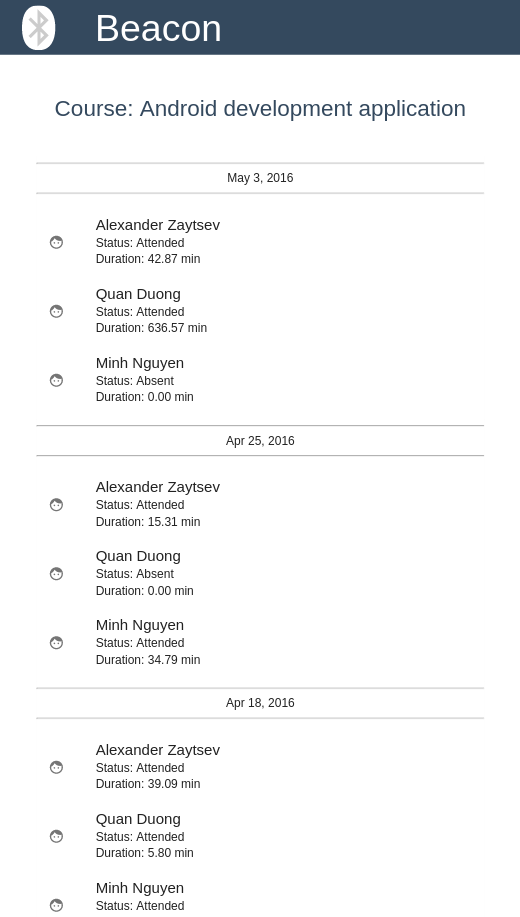


Figure 8.

Figure 8 shows all the classes of the course which available until the current date of accessing and status of student as well as the duration of students in the class.

**8. Conclusion:**

We learned about fundamental components of an Android application and were able to apply the theory we learned to each assignment each week, finally we did a project which is somehow helpful for students and teachers, and the application can be developed further with the existing source code.