LAM Project 2024

Federico Montori, Nicolas Lazzari

April 2024

Rules

In this document we describe two possible projects for the exam of "Laboratorio di applicazioni mobili" course. The project can be implemented by **groups of maximum 2 people** (individual projects are obviously allowed as well), provided they implement the project integration. Each student/group can choose to develop one of the projects proposed here (valid until February 2025) or suggest something else based on his/her personal interests. In the latter case, project proposals **must be individual only** and should be submitted via e-mail to Dr. Nicolas Lazzari `nicolas.lazzari3@unibo.it`, with a brief description of the application goals, contents, and a list of **all** the features that you propose to implement. In extremely exceptional cases we can allow group projects outside the one proposed here, but they will need to be discussed with us in advance. The use of Git (or any other versioning system) is strongly encouraged. The following project description contains a minimum set of requirements for the application. Students are strongly encouraged to expand the track, by adding new features to the application, and/or further customizing the contents. Any questions regarding the projects are to be asked to Dr. Nicolas Lazzari via e-mail (`nicolas.lazzari3@unibo.it`). Regardless of the choice, every student/group is required to produce:

1. *One or more mobile applications*, there is no constraint on the language and the framework used: it can be native or hybrid, but it cannot be a Web Application (*i.e.* it must run natively on the device, and there must be a reason to do so).

2. *A project report*, which is a document that describes the application produced, focusing on the workflow and the design choices. In particular, the Report should be named `SURNAME1_SURNAME2.pdf` and contain:

    - The name, surname, email, and matriculation number of each component of the group.
    - Overview of the application with screenshots.
    - Implementation details on how you chose to implement the functionalities.

A good report is probably between 10 and 15 pages. Less than 5 pages are probably bad, and more than 15 is probably too many. The quality of the report WILL be part of the evaluation. **A good report also contains about 70% of implementation details and choices and only the remaining 30% of screenshots, overview. . .**

3. *A presentation*, which consists of a set of slides that will help you in the project discussion. They should contain a brief recap of the report alongside screenshots of the application. We suggest producing around 10 - 15 slides since the presentation time is approximately 10 minutes (a group discussion may last longer). Furthermore, **each** component of the group must know the details of the entire implementation and will be possibly asked about it during the project discussion.

The **CODE** of the mobile application and the **REPORT** have to be uploaded exclusively on the Virtuale platform in the dedicated section[1] (there are 6 deadlines throughout the year). They must be enclosed in a single .zip file named `SURNAME1_SURNAME2.zip`. In the case of a group, a single component is in charge of completing the upload. If the archive is too big for Virtuale, you can remove the directory "build" from your code (Android only). If the archive is still too big, then the student(s) must share the zipped code via Drive/OneDrive, etc. In this case, the students must still upload on Virtuale a zip file containing the Report, the link to the shared code, and its **hash**, so that we can prove that it was delivered on time. Projects delivered via e-mail will **NOT** be taken into account. The SLIDES, instead, must be brought along on the day of the oral examination.

Do not forget that the oral examination consists also in a theoretical part, which is **individual**. Therefore, knowledge of the topics discussed in class (both iOS and Android) is required. The exam consists of the project discussion (which is per group) and the oral examination (which is per student) and **must** be booked via Almaesami.

The Project: "Personal Physical Tracker"

Overview

In the following project, the student is required to implement an interactive application that monitors the physical activity of the user depending on the mean of transportation. A user can be walking, staying still, or, for example, driving a car (there are many other activities that can be added). The goal of the project is to display the user with an interface where he/she can start/stop one of these activities and these will be recorded in a local database. The user can get access to all the activities done by seeing them in a list (or better, a calendar) and receive the report of the monthly activities through plots (e.g. pie charts). Furthermore, when the user is walking/running, we also want to

[1] `https://virtuale.unibo.it/course/view.php?id=28374`

record his/her number of steps taken per activity chunk, so that we know how many steps one person has made daily. The app also provides the user with a background functionality, either to record the amount of time the user has been in a certain area, or performing the activity recognition in the background. Disclaimer: the term "activity" in this section means a physical activity, not an Android Activity.

Record the activities

The app should be able to display an interface with which the user can start/stop an activity. The app must record the time spent within this activity and, once the activity is done, the app must be able to save the information in a local database. An activity is of type: walking, driving a car, or sitting. The types of activities can be increased, but these are mandatory. Activity types can have special features attached to them. For the activity type "walking" it is mandatory to record the number of steps as a special feature. This can be done in Android using: `https://developer.android.com/health-and-fitness/guides/basic-fitness-app/read-step-count-data`. The user must be able to see the history of past activities either in a list or in a calendar (or any other means of visualization) and there must be at least one filter in the visualization. Logically, it is nearly impossible to cover all 24 hours every day with activities, even though this would be the ultimate goal of the application. The student must then find a way to handle the absence of activities on certain time frames over the day (they can be simply additional activities of type "unknown", for example).

Display Charts

The app must have a section that shows at least two different charts about the activities. For instance, there can be a pie chart showing the activity types over the past month and how much they have been performed. Another idea can be a line plot showing the daily number of steps taken in the past month.

Perform background jobs

The application must be able first of all to send periodical notifications to the user, reminding him/her that it is time to record activities, or that it is time to do more steps because today the user is looking kinda static... There needs to be at least one periodic notification. Furthermore, the application also must do **one of** these background operations:

- The app must understand in the background what kind of activity the user is performing and then record it autonomously without the user having to insert it manually. One way to do this in Android could be using the Activity Recognition API "`https://developer.android.com/develop/sensors-and-location/location/transitions`". Maybe it would be nice to ask the user if the guess is correct from time to time.

- Alternatively, the app must record the user's presence within a certain area of interest that must be registered in advance. This can be done by setting Geofences `https://developer.android.com/develop/sensors-and-location/location/geofencing`. Note that these additional recordings are not activity types, they are separate concepts that have to be handled differently (the student must choose how).

Project Integration for 2 people

In the case of a project for 2 people, students must implement also the following additional feature: the application should be able to display data coming from other users. In particular, a user can contact other users with the same app and "follow" them. More in detail, the application has to store, process, and visualize the other followed users' data, while keeping each data source separate in memory. For the first two tasks, we suggest using different databases/tables, while for the last one, it is possible to show the data of a different user on a different calendar/list... Data must be saved locally as it is imported, however, an additional good practice could be also "monitoring" the followed user over time if there is a shared repository somewhere or if the data from such a user gets imported again. Logically, we cannot add activities to other users' records, we can only visualize them. Users can share their data in different ways:

- *Remote cloud sync.* Users can use a cloud-shared database to synchronize their data, like Firebase. The sync action can be triggered manually or automatically following custom strategies, and the data schema used must be customized by the students. This does **not** replace the local database for his/her own data.

- *Import/Export of a database dump, using a custom data format.* Users can share the file containing their data through an external app and import it into the application to process the additional data. The schema used for exporting the data must be defined by the students.

- *Proximity share.* Users can share their data when approaching a proximity area with another user. The proximity area can be created using, for instance, BLE beacons. If two users become visible in the same area, they are notified that someone else is available and they can use an M2M communication (WIFI-direct, Bluetooth) to share their data. The data schema used must be customized by the students and the range of the area can be customized through the application settings.

All the previous strategies are valid for the exam evaluation, but (from the top to the bottom) the difficulty increases and so does their weight for the final evaluation.