Informatique et systèmes de communication ISC
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

208.1 – Mobile development summer school v1.0

25th of August – 5th of September 2025

Goals of the project of the Summer School

The aim of this project is to enable you to:

- Define and develop a cross-platform mobile application in the domain of data engineering and analytics, integrating different data sources and exploiting them.
- Work and collaborate with other students in a team.
- Apply Software Engineering good practices.
- Implement the SCRUM methodology for the realization of this application.
- Communicate with the different stakeholders in English.

Please find below a description of what is expected of the application, the technologies to use, how to organize your groups in SCRUM, and the various deadlines.

We want you to achieve a high professional level in the management and development of this application.

Technologies

In this project, we will be using the following technologies and frameworks:

- Flutter (framework for multi-platform mobile development)
- Firebase framework (used for the data storage of your mobile applications)
- DeepFace python framework to recognize faces
- Machine Learning libraries to train regression models

Asynchronous Preliminary Training Recommended

We recommend that you follow several personal trainings in advance :

- Flutter basic tutorials such as:
  - https://docs.flutter.dev/get-started/
  - https://docs.flutter.dev/ui
  - https://docs.flutter.dev/ui/layout/tutorial
  - https://docs.flutter.dev/ui/interactivity
  - https://docs.flutter.dev/cookbook/navigation/navigation-basics
  - https://docs.flutter.dev/data-and-backend/state-mgmt/intro
- Firebase as cloud backend for Flutter:
  - https://docs.flutter.dev/data-and-backend/firebase

Group composition and mandatory presence at meetings!

The group compositions are available on ISC Learn. The size of each group is ideally of 3 students, but possibly of 4. Each group has been assigned a prof coach who takes the role of a SCRUM Product Owner.

Note that the presence of EACH student at ALL Scrum meetings (daily standup, sprint review and planning) is mandatory. Please, respect the other students in your group by participating in all those meetings, as this shows the level of your professional maturity. You will be graded on this.

Each group will have the following meetings with the respective prof coach:

- Kickoff Meeting (Sprint 0 planning)
- Sprint 0 review / Sprint 1 planning
- Sprint 1 review / Sprint 2 planning
- Sprint 2 review / Sprint 3 planning
- Sprint 3 review

One of those meetings may be online.

In each sprint, each group is expected to have at least 3 daily standup meetings (each half day), documented in the SCRUM file. The prof coach will most likely participate in one daily standup meeting per sprint.

Professional Communication in English

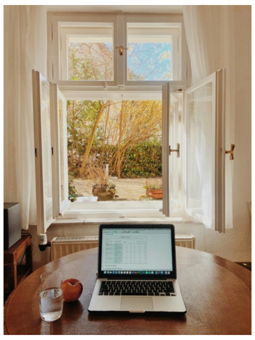
We want you to use a high standard of professional communication in English.

The communication in English within the group will concern:

- Email sent before each sprint/review to the prof coach to announce the result of the sprint (main goal, achievements, link to deployed application, impediments) and the agenda of the meeting.
- Sprint review/planning meetings
- Communication on the group channel
- User guide
- Technical guide (can be done directly in GitHub)
- Comments in the code
- Final video and final presentation

Application Requirements: ROOM RENTING

Vision



Are you a student planning to spend some time at another HES-SO institution, whether for a month, a semester, or even longer? Wondering how to secure accommodation during your stay? Or perhaps consider renting out your current room or flat since you will not need it temporarily. Do you have a spare room or a flat available for rent? Are you seeking a suitable tenant to occupy it?

The purpose of this platform is to facilitate connections between individuals in those situations, making it easy to find either a room or a tenant.

The different student groups must develop the same application vision; however, they will differentiate by the specific additional information to be added.

Requirements

Overall

- Multi-platform mobile application
- Easy-to-use UI/UX
- Security by Design: data should be accessed only by authorized users.

Minimal Functional Requirements:

- The mobile application should be responsive and work on various devices such as smartphones and tablets.
- User registration and authentification.
- User profile information storage, including the profile picture, home address and HES/University address.
- Recognition of a person on a taken picture.
- If only one person is on the picture, binding of the picture to a specific registered user and ask the password.
- About information (developers, etc.).
- Property listing creation, including photos, descriptions, location, price (for students).
- Search functionality for students to find rooms based on various filters such as location, price range, property type (whole property/ single room), and other amenities (Internet, …).
- Management of the availability of the rooms (for homeowners).
- Calendar integration to track rental periods and set custom availability schedules (for homeowners).
- Booking system to manage rental agreements and reservations (for students).
- Review and rating system for both students and homeowners after the completion of the stay.
- By integrating a public transportation API (such as https://transport.opendata.ch/), proposition of several possible public transportation paths for the next two hours to go the student HES/University (for students).
- Based on an existing dataset of existing properties, creation of a prediction model (linear regression, etc.) to predict the price that this property should have.
- Comparison of the proposed property price and the predicted price (for students and howeowners).
- Proposition of additional useful information on the destination (such as weather forecast, …, see below) by integrating external API data (for students).
- Admin view on registered users.

Commenté [AV1]: We could clarify, for every bullet point, whether we consider for guests or for hosts (property listing is for guests, manage availability of the rooms is set by the host, seen by the guests, etc)

Commenté [AV2]: There might be no university / HES in the city. I think it might be best for each user to enter the university he/she attends, and compute the paths to this university.

Nice to Have Functional Requirements

- Live update of availability based on reservations (for howeowners).
- Profile management for users including personal information, preferences, and history.
- Profile data recovery by email or SMS.
- A dashboard for users to easily access their listings, bookings, messages, and reviews.
- Messaging system for communication between students and homeowners.
- Onboarding tutorial for new users to decrease the initial learning curve.
- For homeowners: Insights on property performance, including occupancy rates, average stay length.
- Notifications for users about new listings that meet their saved preferences (for students).
- Allow users to save listings to a favorite list for easy access (for students).
- Help comparing different options before deciding (for students).
- Integrate interactive maps.
- Admin dashboard on user statistics (number of recognized faces, etc.).
- Admin user management (delete user, password reinitialization).
- Further third integration of external API data.

Technical requirements:

- The mobile app must be a multiplatform and developed in Flutter.
- Its data must be stored within Firebase.
- You must use the deepface framework for image recognition.
- You must create a dataset of the faces to recognize.
- You must train a face recognition model, and integrate it in the App.
- You must train a price prediction model, and integrate it in the App.

Commenté [MS3]: confirm @Andrearczyk Vincent

Commenté [AV4]: I modified it

- You must integrate current transport data from the Transport API.
- You must integrate a second API, such as touristic information of the city of the property, available shared mobility means (Publibike, etc.), etc. In discussion with its Product Owner, each group can choose one API to integrate and show its information within the mobile application.

User stories

You must create the Product Backlog based on the example given to you. From the application description, transpose its requirements towards the product backlog. Define all acceptance criteria and evaluate the story points for all user stories. Those story points must then be re-evaluated during each sprint planning.

Steps and calendar

KICKOFF morning:

During the first meeting, this project will be explained to you.

Then, the main goals are the following:

- Meet all members in your group.
- Decide who the SCRUM Master is and communicate it on your group Teams channel.
- Copy the SCRUM excel document to the Files section of your channel and copy within it all user stories from the application description.

- Plan sprint 0. Within this sprint, you should achieve the following:
  - Setup the full SCRUM Product Backlog (including user stories, acceptance criteria, story points evaluations and proposition of priorities from 1000 for the most important to 0 for the less important, done definition, etc...).
  - Set up all the tools for each student.
  - Set up and show GitHub repository for each student.
  - Decide about the GitHub policy within your group.
  - Prepare mockups for all mandatory user stories (Minimal Functional Requirements).
  - Implement a first very simple functional user story that can be used as reference for a user story of 1 story point. This should be a very simple functionality going from the frontend to the backend.
    - Deploy your first application.
    - Be aware that your application must be deployed and testable by your prof coach before each Scrum meeting.

- Meet with your prof coach.
- Fix ALL meetings with your prof coach.

Calendar

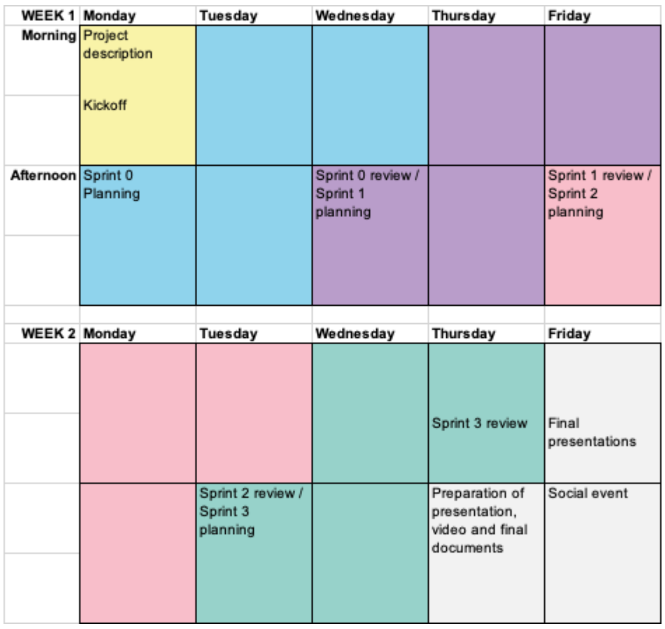
- Decide with your prof coach about meeting dates.

Next sprint reviews and planning:

- After the Sprint 0 planning, each group is expected to organize 4 sprint review/plannings meetings.

Before each sprint review meeting, you must send an email to your prof coach announcing the agenda of the meeting. Summarize what was done in the last sprint and give the link to the deployed application and your source code. Communicate in a professional way.

The next steps of the project will be the following, according to the following calendar:



WEEK 1	Monday	Tuesday	Wednesday	Thursday	Friday
Morning	Project description Kickoff				
Afternoon	Sprint 0 Planning		Sprint 0 review / Sprint 1 planning		Sprint 1 review / Sprint 2 planning
WEEK 2	Monday	Tuesday	Wednesday	Thursday	Friday
				Sprint 3 review	Final presentations
		Sprint 2 review / Sprint 3 planning		Preparation of presentation, video and final documents	Social event

Final Deliverables & Presentation

The following elements must be delivered the day before the presentations:

- Code committed to GitHub.
- User Manual document.
- Technical Guide document (can be markdown documentation in github)
- Scrum documentation ("Excel" document).
- 5 minute - video in English describing the usage of the application OR live demo, specified at the beginning of the project.

You must also make a final presentation (5-10 minutes) as part of the public event of all Summer Schools. There, the focus should be on the final application and the used technologies.

Evaluation criteria

The evaluation will be split as follows:

- 10% User manual and technical guide
- 10% Final presentation & video
- 20% SCRUM management and documentation
- 25% Final application: mobile app and APIs integration
- 15% Final application: data analytics
- 20% Final application: functionalities of the Product Backlog

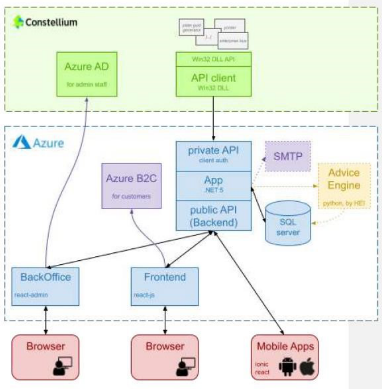
User manual and technical guide

The user manual:

- Will include the full description of the application from a user perspective, including good-quality screenshots.
- A table of content should be present with an order considering the most important functionalities first.
- The writing should be neutral, without humor or personal thoughts. It should be as concise as possible and include only information helping the user to use that specific functionality.
- Use markers on screenshots, when necessary, to help the user focus quickly on the important parts.

The technical guide criteria:

- Architecture schema. Example below:



- Use the appropriate diagrams which help best in explaining your choices in data classes and which functionality is related to which file.
    - Database structure (schema or other if not using tables), sequence diagram, data flow analysis
    - Appropriate UML schema
    - Explanation of Technologies and versions of additional libraries used.
    - Word document that will evolve throughout your project or directly work with markdown in Git.

SCRUM management and documentation

Concerning the SCRUM management and documentation, **you will be evaluated at each sprint meeting**. Below you can find the evaluation criteria.

For Sprint 0 review / Sprint 1 planning:

- Are the mockups covering the full product backlog?
- Are the User roles defined?
- Product Backlog (PB): are the user stories complete?
- PB: are the acceptance criteria complete?
- PB: is the list of priorities complete and according to the user needs?
- PB: are the MoSCoW complete and according to the user needs?
- Done definition: was it defined, and is it realistic?
- Were all tools setup for everybody?
- Is a git repository existing, and a first commit done?
- Was a first "hello world application" deployed?
- Are the User stories story points estimated?

For each further sprint review/planning:

- Are Daily meetings documented?
- Is the "journal de bord" documented by each team member?
- Was the Product Owner (PO) informed before the meeting of the agenda, sprint summary and link to deployed app?
- Is the app deployed and testable by PO?
- Is the code for every single user story committed on git?
- Sprint tab: are the sprint tasks correctly planned and documented?
- Sprint tab: is the burn down chart correct?
- Retrospective: is the meeting correctly done and documented?
- Are the User stories story points re-evaluated?

Furthermore, for the last sprint review:

- Is the release roadmap correct?

Final application: technical criteria

- Git Usage Commits / Branches, naming
- Code maintainability and scalability
  - Respect and apply "best practices" to promote code maintainability and scalability, separation of concerns (business layers, logic, etc.), architecture model (e.g. MVC), design patterns. Well-structured and commented code.
- Project architecture
  - Separate classes according to layers or architecture model. The project structure/architecture must be clear and facilitate code maintainability and evolution.
- Master widgets (with a particular focus on stateful widget):
  - This is the most important criteria because it's the whole philosophy of Flutter. You need to understand the architecture/functioning of widgets (widget tree structure) and state management, stateful vs stateless widgets. Use stateful widgets only when necessary, not by default.
- Cross-platform compatibility:
  - The app must work on iOS/Android as well as on the computer. This tests whether the app works and is responsive, whether the user experience is the same everywhere, and whether the specific features of each platform are considered.
- Responsive design:
  - The layout must adapt to screen size and orientation (portrait/landscape). Use dynamic rather than fixed values (e.g. 50% of screen width rather than 200px). Often the application displays correctly on the phone on which the students are developing it, but not when the size/device changes.
- User-friendly application
  - UI/UX, create a user-friendly app, customize widgets, font, size, style, themes, icons, colors, animations, etc. Give feedback to the user (during loading, when a button is clicked, etc.).
- State management
  - Understand and correctly manage states to avoid unnecessary re-renders and improve performance. Use one of the most popular approaches (Provider, Bloc, Riverpod, etc).
- Widget composition
  - Understand the system/functioning of widgets and use them correctly. Create custom/enhanced widgets based on those provided by Flutter and re-use them when necessary. Aim for a modular approach that promotes code maintainability and reusability.
- Navigation and hierarchy
  - Correctly manage routing from one screen to another, using best practices, passing elements from one screen to another correctly, from a parent widget to a child widget or vice-versa.
- Firebase integration
  - Manage authentication/connection and data storage with Firebase, avoid unnecessary queries, integrate logic in a specific layer (i.e. in the right place in the project architecture/code), use

appropriate widgets (Flutter provides widgets specifically for this purpose: StreamBuilder, FutureBuilder, etc).
        - Managing users and user rights
        - CRUD operations with Firebase

- Network handling
    - Perform queries correctly, avoid unnecessary queries (or queries that are made at every re-render), cache results, handle errors, manage if the user has no network (which can happen with a telephone). Keep data local and make queries as soon as possible (in the event of an upload, for example), etc.

Final application: data analytics

Commenté [AV5]: Voilà une suggestion

- Data Preprocessing, Analysis, Splitting and Model Training
    - Perform data cleaning and preprocessing, including: Handling missing values, Encoding categorical variables (e.g., one-hot encoding for room type), normalizing or scaling features if necessary
    - Conduct exploratory data analysis to understand feature distributions and identify useful predictors for price
    - Split the dataset into a training and test set
    - Train a room price prediction model
    - Save the trained model for use in the app
- Model Evaluation
    - Evaluate the trained model using the test set using regression performance metric
    - Interpret the model’s performance and briefly discuss limitations
- App Integration
    - The model must be integrated into the app and allow the user to click a button to predict the price of a given room
    - The app should display the predicted price and compare it to the real price of the room
    - The app should handle missing data or prediction errors gracefully