University of Worms, Applied Science in Informatics

Inline Skate – Mobile App

Project Proposal for SWA (Course 506)

Nhat-Lam Luong (675523, inf3381@hs-worms.de)
Winter term 2022
with Prof. Dr. Volker Schwarzer

Table of Contents

Table of Figures

Chapter 1 – Introduction and goals

1.1 Design Purpose

Mid-sized inline skating company Rollerblade DE is set to release a mobile application for Android smartphones and -watches with Wear OS (D1) very soon encouraging people to inline skate more often and compete against each other for fun. For an early rollout (D2), the app is expected to offer basic functionality only, such as creating an account and the ability to log their inline skating sessions in order to publish them in leaderboards. Moreover, the idea is quite original and targets a thriving community, so growth in terms of functionality is expected post-launch (D3). Thus, the objective of this project proposal is to construct an android mobile app architecture (D1) that is extensible in its scale of functionality (D3), interoperable between platforms (D4), while also keeping the initial costs of its time for development reasonable (D2).

1.2 Primary Functionality

The following list is sorted by priority in descending order.

ID	Function/Priority	Description
#F1	**Login manager** --- Being logged in into an account grants the user access to the functionalities of the application.	• **Login Check (#F1.1):** It checks upon application start, if the user is already logged in or not. If yes, then the home screen opens. Else, then the login screen opens. • **Login Screen (#F1.2):** After the user has installed the app or is logged out, the UI will visualize the login screen, so the user has access to Login and Create Account. • **Login (#F1.3):** The login is located in the login screen, where it is in the form of text fields, so the user can login into an account by entering the username and password. • **Authenticate login (#F1.4):** This functionality is used by the login function. It checks if the login is legitimate by using an external authentication server (#CT1) and database (#CT2). If so, then the user will be logged in and is redirected to the home screen. Else, the user gets notified about the error and returns to the login screen. • **Recover Account (#F1.5):** Located in the login screen, the user can use this functionality to recover their account by entering their email address and get an email with their username and password. After that redirect to login screen. • **Create Account (#F1.6):** The create account section is located in the login screen, where it is in the form of text fields. There, the user can set up an account by entering their email, username and password. After creating their account, the set-up profile opens. • **Set-up profile (#F1.7):** This functionality activates itself after the user has created their account and the UI will visualize that with a new screen. In there, the user has to set-up their profile with a nickname of their choice made out of letters and numbers with a length of 10 characters and an avatar that will resize itself to 200x200 pixels. After that the user will be redirected to the home screen.
#F2	**Home screen** --- Here the user lands	After the user has been successfully logged in into an account, the UI will visualize the home screen. There, they have access

	after logging in and gets access to further functions.	to Show Profile, Start Inline Skate Session, Show Leaderboard and Show Friends List.
#F3	**Profile manager** --- Each account has a profile and both must be manageable for the user.	• **Show Profile (#F3.1):** This functionality is accessible for the user in the home screen. When they access it, the UI will visualize the user's profile on the screen, where a profile would contain information of the user's nickname, avatar, friends and their session history. For the session history, the function session history is called to display it. On exit, they will be redirected back to the home screen. Also it grants access to edit profile and edit account. • **Edit Profile (#F3.2):** This function becomes accessible for the user, when they access show profile. In there, the user can access edit profile, where they can edit their nickname made out of letters and numbers up to a length of 10 characters and avatar with a max pixel size of 200x200, while the alterations made to their profile will be saved. On exit, they will be redirected back to show profile. • **Edit Account (#F3.3):** This functionality becomes accessible for the user, when they access show profile. In there, the user can use this function, where they can edit (email address or password) or delete their whole account. If they decide to delete their account, then the user will be logged out and redirected to the Login screen and the account is deleted from the backend. • **Session History (#F3.4):** This function is called, when the user accesses show profile. Then, it will show the user's history of inline skate sessions each with its date, skated time and distance, while initially the history is empty.
#F4	**Session manager** --- It comes after F-3 because the CEO wants to resolve the privacy issues first. Also, this function will provide session data for the "Session History" function of F-3, so the session history will take precedence.	• **Start Inline Skate Session (#F4.1):** This function is accessible for the user in the home screen. When accessed, the UI will render a new screen to visualize the function. After that, their inline skate session will start. The UI will then show a timer that shows the passed time since the user has started their session and it will also show the distance they have skated using calculate and show distance function. Also grants access to end inline skate session. • **Calculate and show distance (#F4.2):** When the user accesses start inline skate session, the UI will show the distance they have skated. In order to calculate that, the function will use the GPS of the smartphone. • **End Inline Skate Session (#F4.3):** This functionality is accessible for the user, when they access start inline skate session and in there, they can find this function. When the user decides to use it, the UI will render a new screen, where the user can see the total amount of time passed and distance skated of their current session. After that, the save stats function saves the statistics to their profile and then they will be redirected back to the home screen. • **Save Stats (#F4.4):** This function automatically activates, when end inline skate session is called. There, the data (date, passed time and distance skated) of their ended session will be saved and added to their session history.
#F5	**Sort list** Leaderboard and friends list use it.	• This function gets called by the show friends list or show leaderboard. It will accept a list of users and sort them by skated distance in descending order and returns it back.

#F6	**Leaderboard manager** --- For the Leaderboard Manager to work, F-1 to F-5 have to exist first.	• **Show Leaderboard (#F6.1):** The user can access this functionality from the home screen. When accessed, the UI will visualize the function in a new screen, where it will show the leaderboard that ranks the users based on their skated distance, for that it will call sort list. Also, the leaderboard would be split into pages with 50 entries each. On exit, they will be redirected back to the home screen. • **Show own ranking (#F6.2):** This functionality becomes accessible when the user uses show leaderboard and in there, they can find this function. When used, it will find the currently logged in user in the leaderboard and open the page where the user can be found in. • **Search user (#F6.3):** This functionality becomes accessible when the user uses show leaderboard and in there, they can find this function in the form of a search bar, where they can enter the exact nickname and then the function will then find out the current standing of that nickname in the leaderboard and open the page where that user can be found in. If the nickname is not found then the function will notify the user about that through the UI.
#F7	**Friends Manager** --- According to our usability expert, this feature will be utilized the least.	• **Show Friends List (#F7.1):** The user can access this through the home screen. When accessed, the UI will visualize that in a new screen, where it will show their list of friends and in descending order based on their skated distance, for that it will call sort list. Upon exit, they will be redirected back to home screen. • **Add friends (#F7.2):** This functionality becomes accessible when the user uses show friends list and in there, they can find this function in the form of a search bar where they enter the exact nickname and then the function will search for that nickname. When the nickname has been found, then the user can decide to send a friend request or not. On exit, they will be redirected back to show friend list. • **Show friend requests (#F7.3):** This functionality becomes accessible when the user uses show friends list and in there, they can find this function. When accessed, the UI will visualize the function and they can see their incoming friend requests, where they can decide if they want to accept or decline. On exit, they will be redirected back to show friend list. • **Show friend profile (#F7.4):** When the user uses show friends list, where it shows their friends in a list, there will be show profile of friend button next to each of their friends in the list. When a button of a friend is accessed, then it will show the profile of that friend and let the UI visualize that. On exit, they will be redirected back to show friend list. • **Chat (#F7.5):** This functionality becomes accessible when the user uses show friends list and in there, they can find this function. Using this, they will be able to chat with their friends or look at their profile using show friend profile.

Figure 1 - Primary functions table

1.3 Use Case View

Use case view for end users.

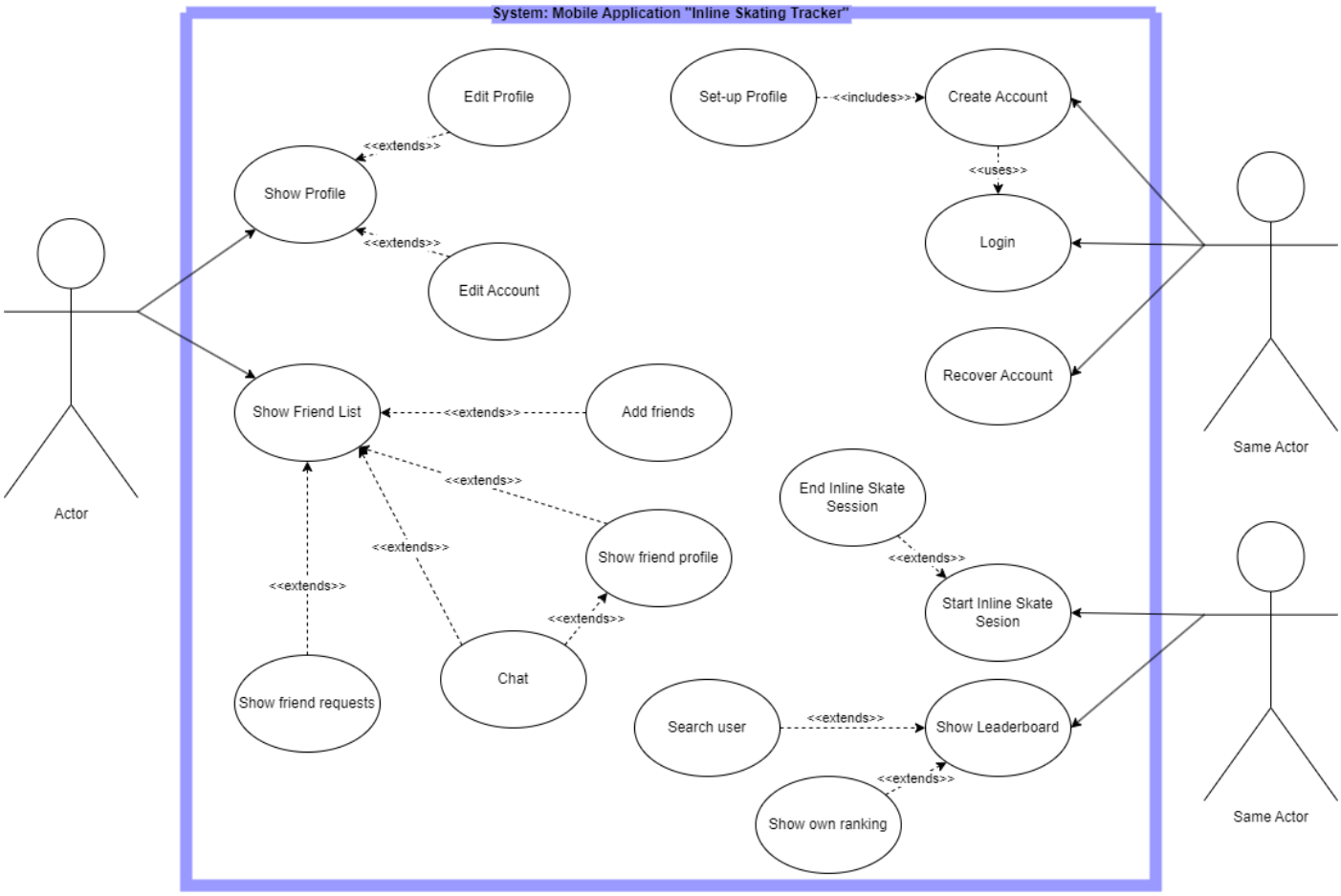

Figure 2 - Use case diagram

1.4 Quality Requirements

The Main-QAs are sorted by priority in descending order.

ID	Main-QA / Prio	Sub-QA (Prio): Motivation	Scenario
#Q1	**Usability** ---------- High: 2x	1. **Learnability (H):** the project team mainly consists of bad developers (#CB3). So, the architectural pattern has to be as simple as possible.	Developer navigates through the code of the system and is able to understand its overall structure swiftly. The time for understanding the overall structure of the code should be under 15 minutes.
		2. **Learnability (H):** the main end user base consists of non-technical users, so the ease of use of the product has to be good.	User navigates through the app on their device, and the UI displays all the information in such a way that the user can quickly learn or understand it. So, there should be no signs of confusion showing on their faces while using the app.
#Q2	**Modifiability** ---------- High: 1x Medium: 1x	3. **Portability (H):** the app has to be made useable for android smartphones and -watches.	End user installs the mobile app on the smartphone and smartwatch and opens it on both devices. The app should then be open and usable on both devices.
		4. **Scalability (M):** the product is new, so more resources need to be added in the future.	Developers add a new function to the app, ideally without breaking a sweat. Thus, the developer's team generated amount of sweat should be less than a bucket (<10 litre).
#Q3	**Interoperability** ---------- High: 1x	5. **Communication between systems (H):** the app should be able to communicate between smartphone, smartwatch and external systems.	The end user calls up the End Inline Skate Session on their smartwatch while it is connected to their smartphone via Bluetooth so that the smartphone can connect to the external database to store the statistics. The database must successfully receive the statistics.
#Q4	**Availability** ---------- Medium: 1x	6. **Fault recovery (M):** implement spare and rollback to recover profile or session data.	Thor (the god of thunder) accidentally strikes the database with a mighty thunderclap while it is working peacefully (rip). A replacement database will then restore the lost data from at least the previous day.
#Q5	**Security** ---------- Medium: 1x	**7. Confidentiality and Authenticity (M):** Personal data should be protected and handled with discretion.	If hackers try to crack the authentication system, the security measures must be able to successfully repel all attacks and prevent any unauthorised access.
#Q6	**Performance** ---------- Low: 1x	8. **Latency (L):** the system needs to be responsive enough for use on the go.	If the end user accesses any function from the home screen, like Show Profile, the execution time must be less than 2 seconds.

Figure 3 - Quality attributes table

Chapter 2 – Architecture Constraints

2.1 Technical Constraints

ID	Constrains	Description
#CT1	Authentication System	The project team doesn't have enough personal for the security of the authentication system.
#CT2	Database	The project team does not have the resources for a database.

Figure 4 - Technical constraints table

2.2 Business Constraints

ID	Constrains	Description
#CB1	Deadline	The CEO prefers a quick product launch, so the initial cost of development time must be reasonable.
#CB2	Software Architect	Unfortunately, the software architect of this project sucks and has very limited knowledge only.
#CB3	Developer Team	Unfortunately, the development staff consists mainly of poor students and their knowledge is very marginal.

Figure 5 - Business constraints table

Chapter 3 – Context and Scope

3.1 Business Context

User: Inline skaters in particular use the app.
Database: An external provider supplies the database and communicates with external provider for authentication system (#CT2 #R6).
Authentication System: An external provider supplies the authentication system and communicates with external provider for database (#F1.4 #CT1 #R5).

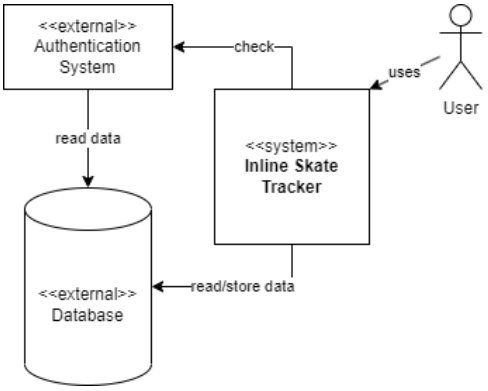

Figure 6 - Business context view

3.2 Technical Context

User: Interacts with the mobile application via the device's graphical user interface, which could be an Android smartphone or smartwatch with Wear OS (#D1).
Graphical User Interface: Displays information and visualises functions.
API: Enables communication and exchange between ISTA and external systems, with two types of API: smartphone and -watch (#CC4).
Authentication System (external): Used by the smartphone or -watch API to verify and authenticate user credentials using the external database (#F1.4 #CT1 #R5).
Database (external): Stores inline skate sessions, profiles and accounts while providing access for the external authentication system (#CT2 #R6).

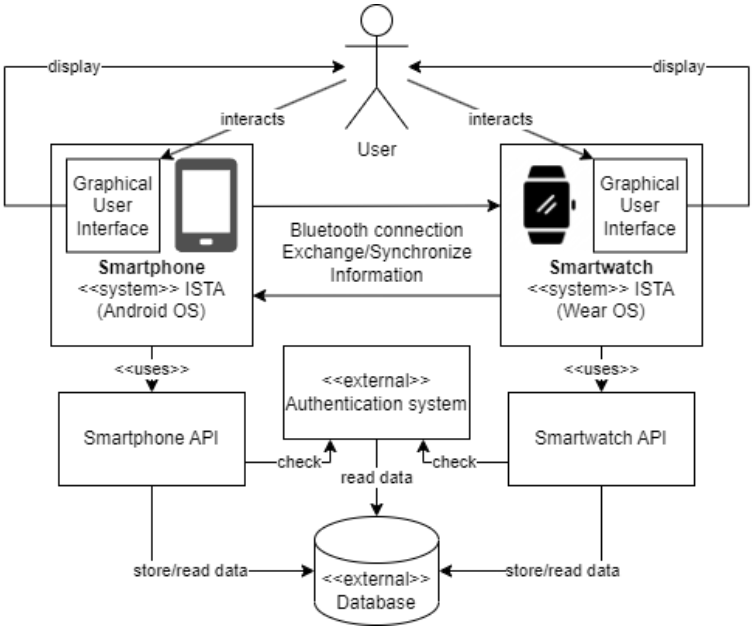

Figure 7 - Technical context view (Figure 26 altered)

Chapter 4 – Solution Strategy

4.1 Organizational Decisions

Software Architecture Life Cycle

The following figure 8 shows the life cycle of the architecture for this project so that the architecture can be continuously evaluated and revised. The architect can participate in the daily scrum meetings to gain knowledge and insight into the project for the evaluation or implementation of the architecture (#CB2).

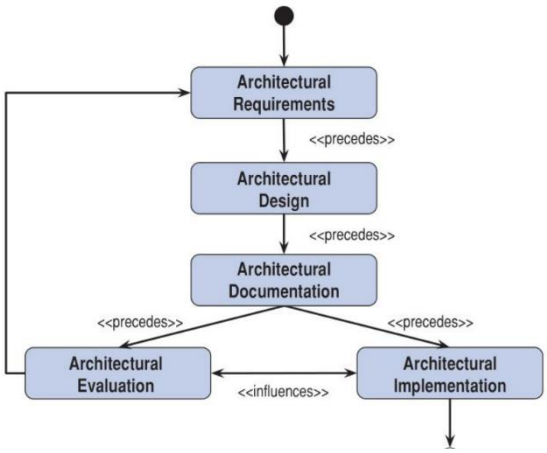

Figure 8 - Software Architecture Life Cycle

Agile development methodology

The software development team works according to scrum with 1–2-week sprints where the software architect can participate in the daily meetings (#CB1 #D2).

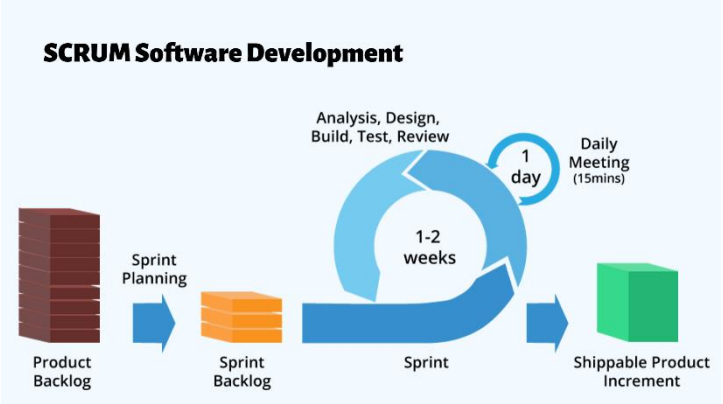

Figure 9 - Scrum Cycle

Chapter 5 – Building Block View

The building block view shows the static decomposition of the <<system>> ISTA into building blocks as well as their dependencies and for a quick overview, the figure 10 depicts again the general scope as seen in chapter 3.1. For the following considerations in this chapter, solely the <<system>> ISTA is considered, as both the authentication system and the database are <<external>>.

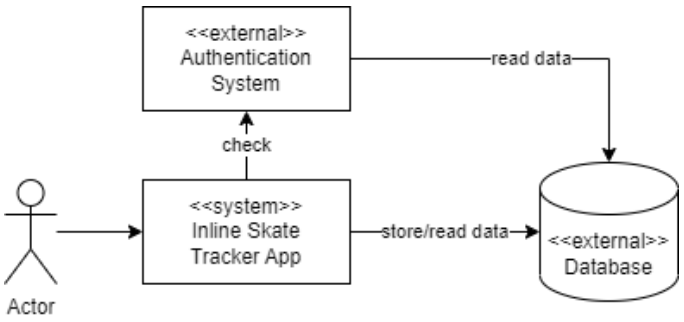

Figure 10 - Fast quick scope

5.1 Level 1 (Implementation View)

The architectural pattern of the system ISTA must be able to handle its most important QAs well, so the priority is on usability (#Q1.1 #Q1.2 #CB3), modifiability (#Q2.3 #Q2.4 #D2 #D3 #CB1) and interoperability (#Q3.5 #D3), while the other QAs, such as availability, security and performance, are not as important. The most suitable architectural pattern is therefore the **microkernel** pattern, as it has been evaluated by software architecture experts as being very compatible with the system's main QAs and also with performance QA (#Q6.8).

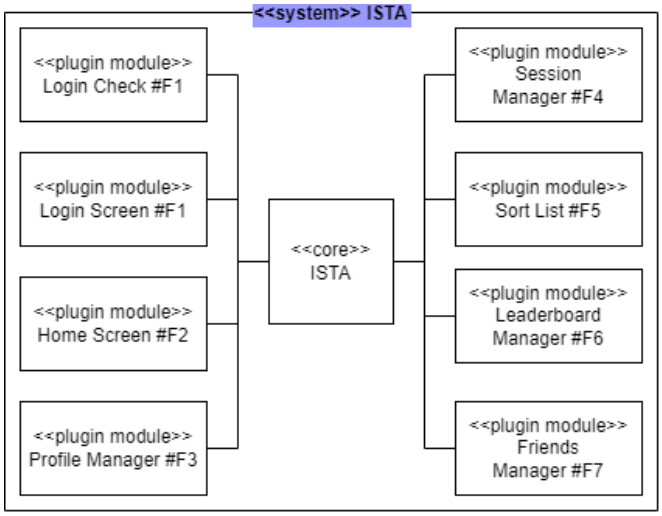

Figure 11 - Building Block Level 1

5.2 Level 2 (Structural View)

At this level, the class diagram in Figure 12 illustrates the plug-in modules and their functionalities. Further descriptions can be found in the references #F1 to #F7, which are also given next to the class names in the diagram.

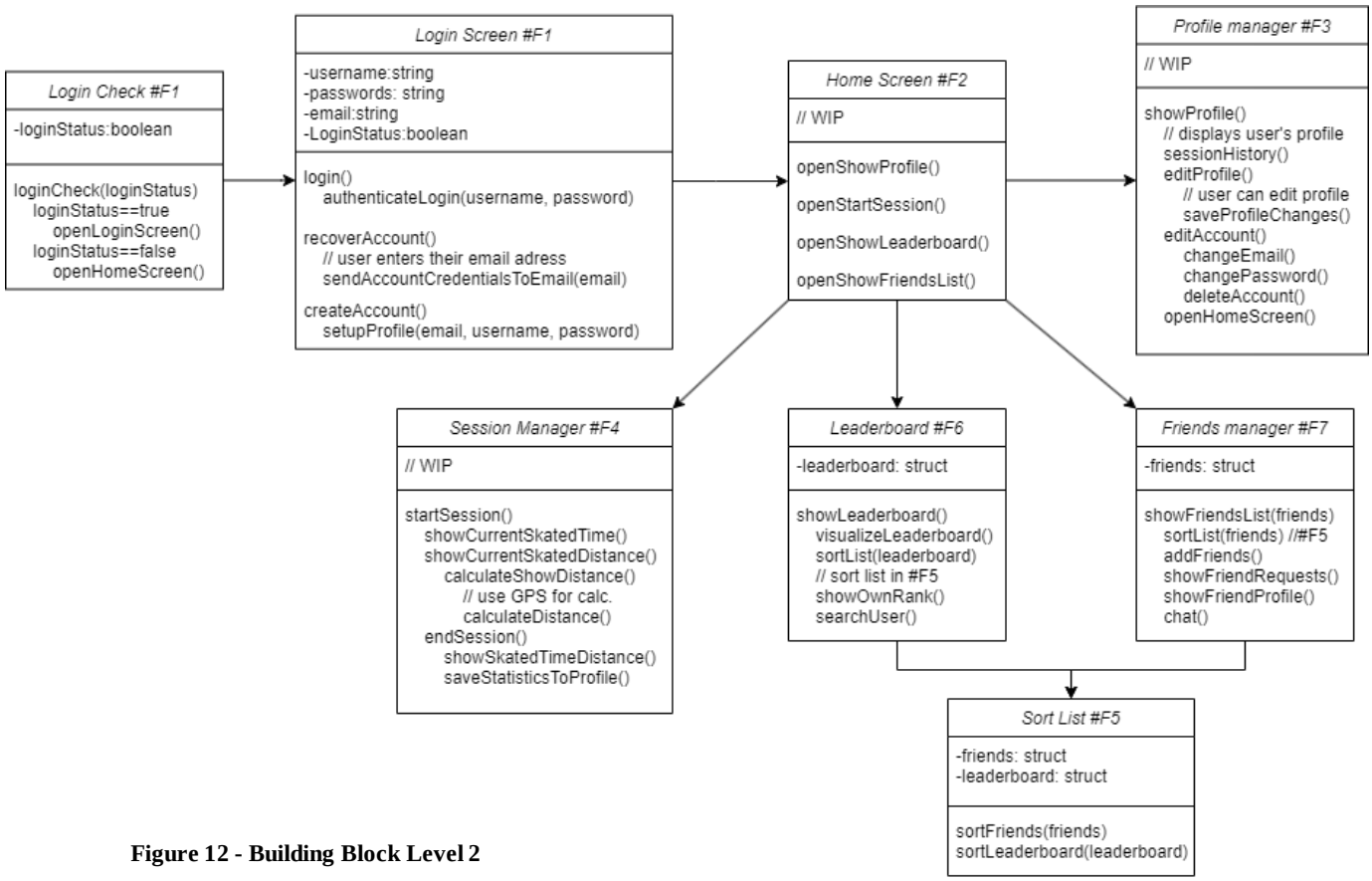

Figure 12 - Building Block Level 2

Chapter 6 – Runtime View

6.1 Overall View

This subchapter is intended to show the overview of the runtime. However, in contrast to chapter 5.2, whose overview is manageable, this overview is too extensive for this documentation. Since it is too extensive, it will be divided into several smaller runtime views in the following subchapters. Further descriptions can be found in references #F1 to #F7, which are also indicated accordingly in the diagrams.

6.2 Login Manager

6.2.1 Login Check

After the user has started the ISTA application, in LoginCheck the login status is evaluated. If the user is not logged in there yet, the login screen opens in chapter 6.2.2.

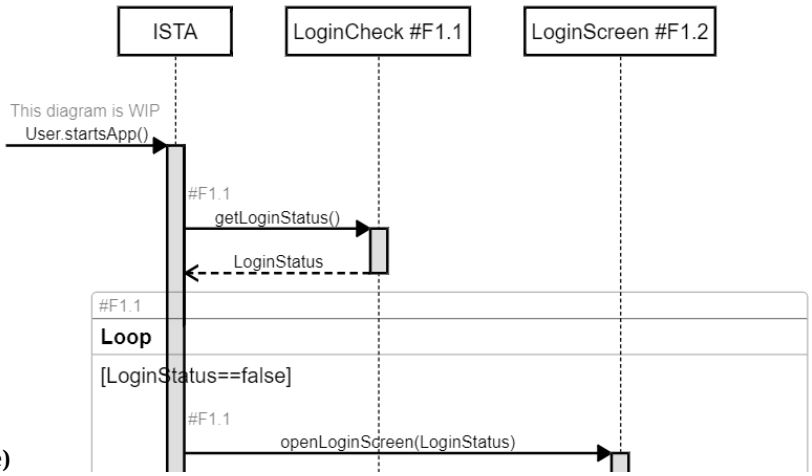

Figure 13 - Login Check (false)

On the other hand, in case the user is already logged in, then the home screen opens in chapter 6.3.

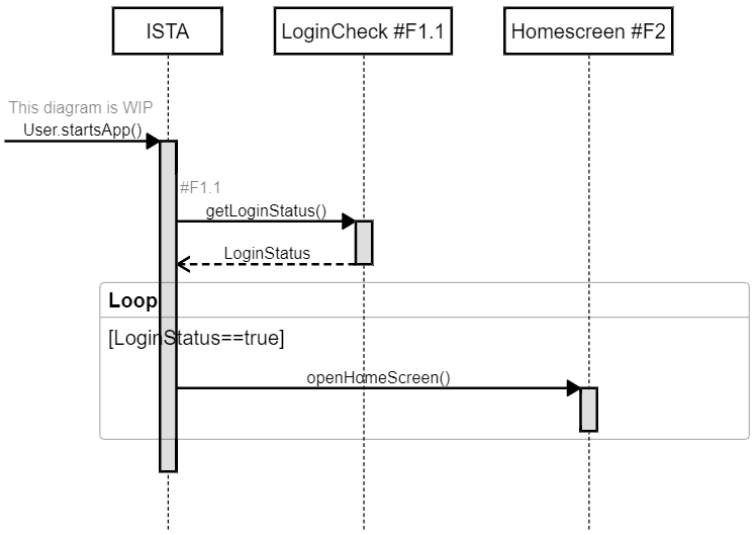

Figure 14 - Login Check (true)

6.2.2 LoginScreen

6.2.2.1 Login

In the LoginScreen, several functions are available to the user. This diagram illustrates the case where the user has an account and wants to login with it, while this process also calls authenticateLogin().

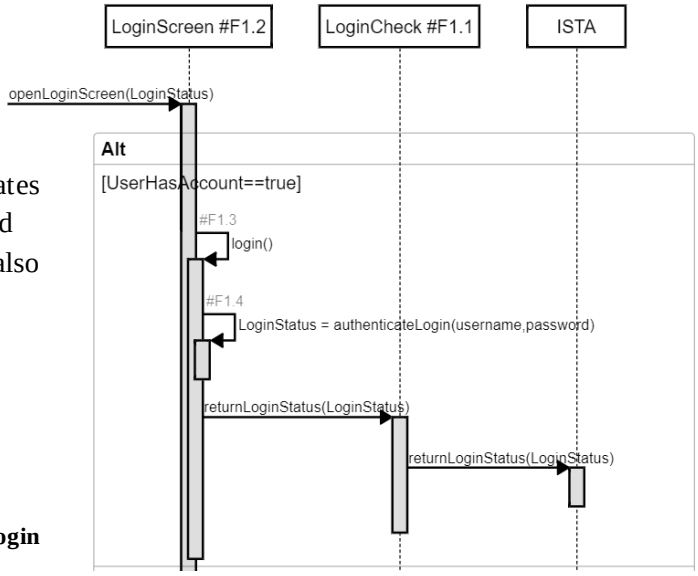

Figure 15 - Login

6.2.2.2 Recover Account

If the user has forgotten their account, they can try to get it back with recoverAccount() where the user enters their email address and then sendAccountCredentialsToEmail().

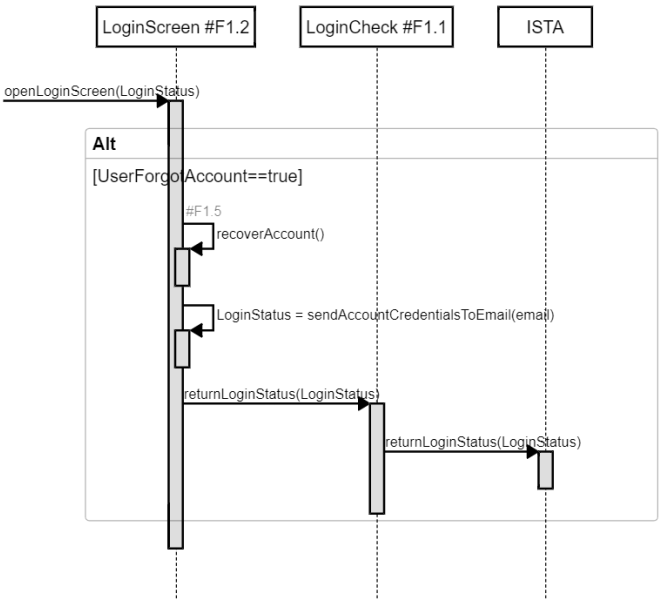

Figure 16 - Recover Account

6.2.2.3 Create Account

When the user does not have an account and decides to create one with createAccount(), then this process will also call setupProfile().

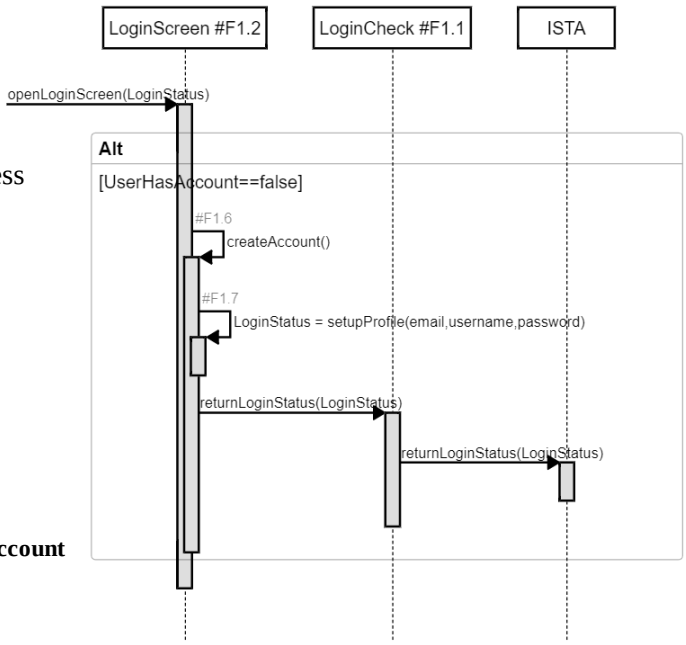

Figure 17 - Create Account

6.3 Home Screen

After the user has passed the login check from Chapter 6.2.1, the home screen opens. From the home screen, they can access the Profile, Session Manager, Leaderboard and Friends Manager classes via the correspondingly named methods. Within these classes, they can also return to the home screen by exiting them.

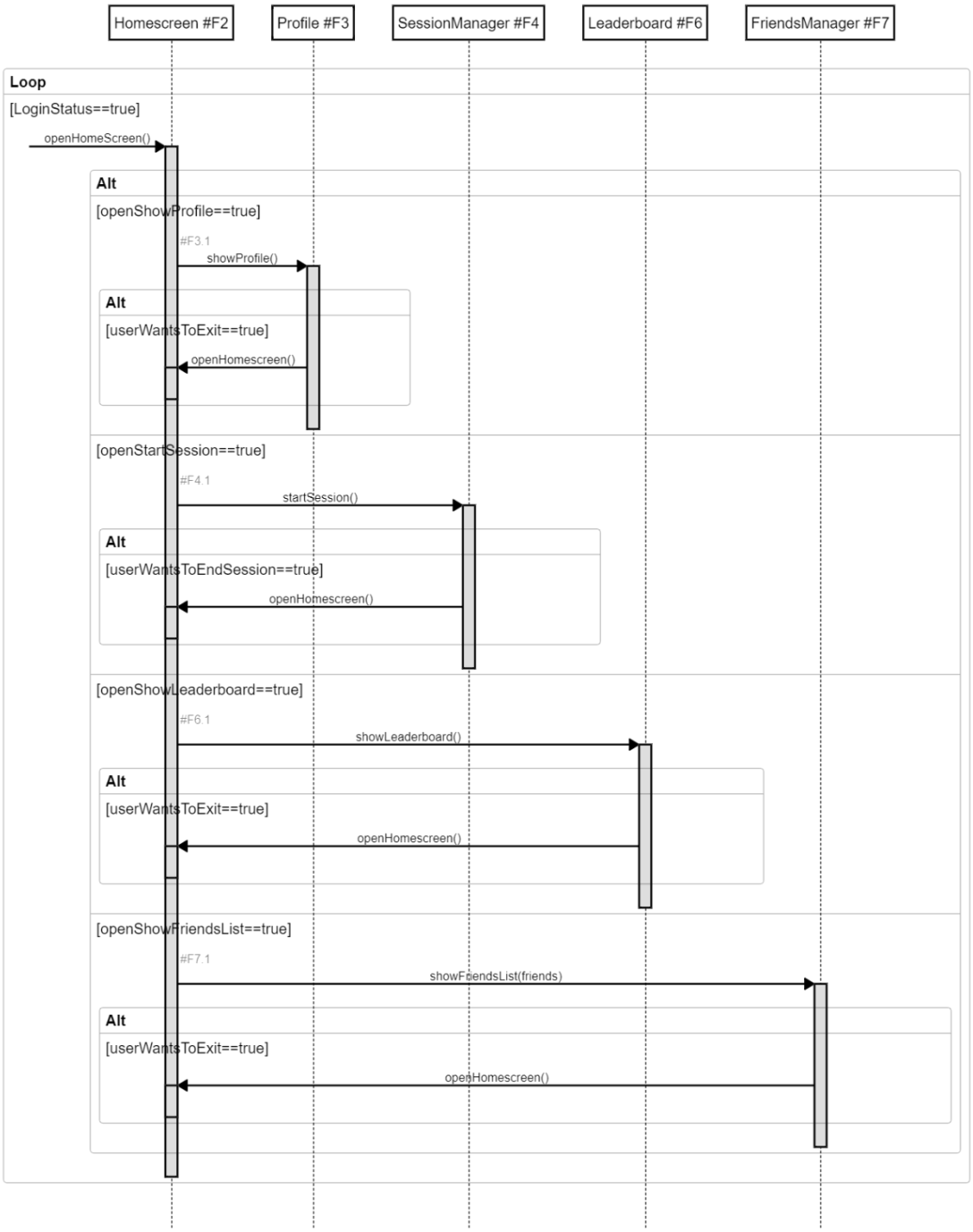

Figure 18 - Home screen

6.4 Profile

When the profile class is called from the home screen, the showProfile() function displays the user's profile with its sessionHistory(). From there, the user has several options:

If they would like to **edit their profile**, they can use editProfile() function to do it, after that the alterations will be saved with saveProfileChanges().

If they would like to **edit their account**, they can use editAccount() function to do it, from there the user can changeEmail() or changePassword() and also decide to deleteAccount().
If the latter happens, the LoginCheck and ISTA classes are notified that the user has been logged out by returning the LoginStatus.

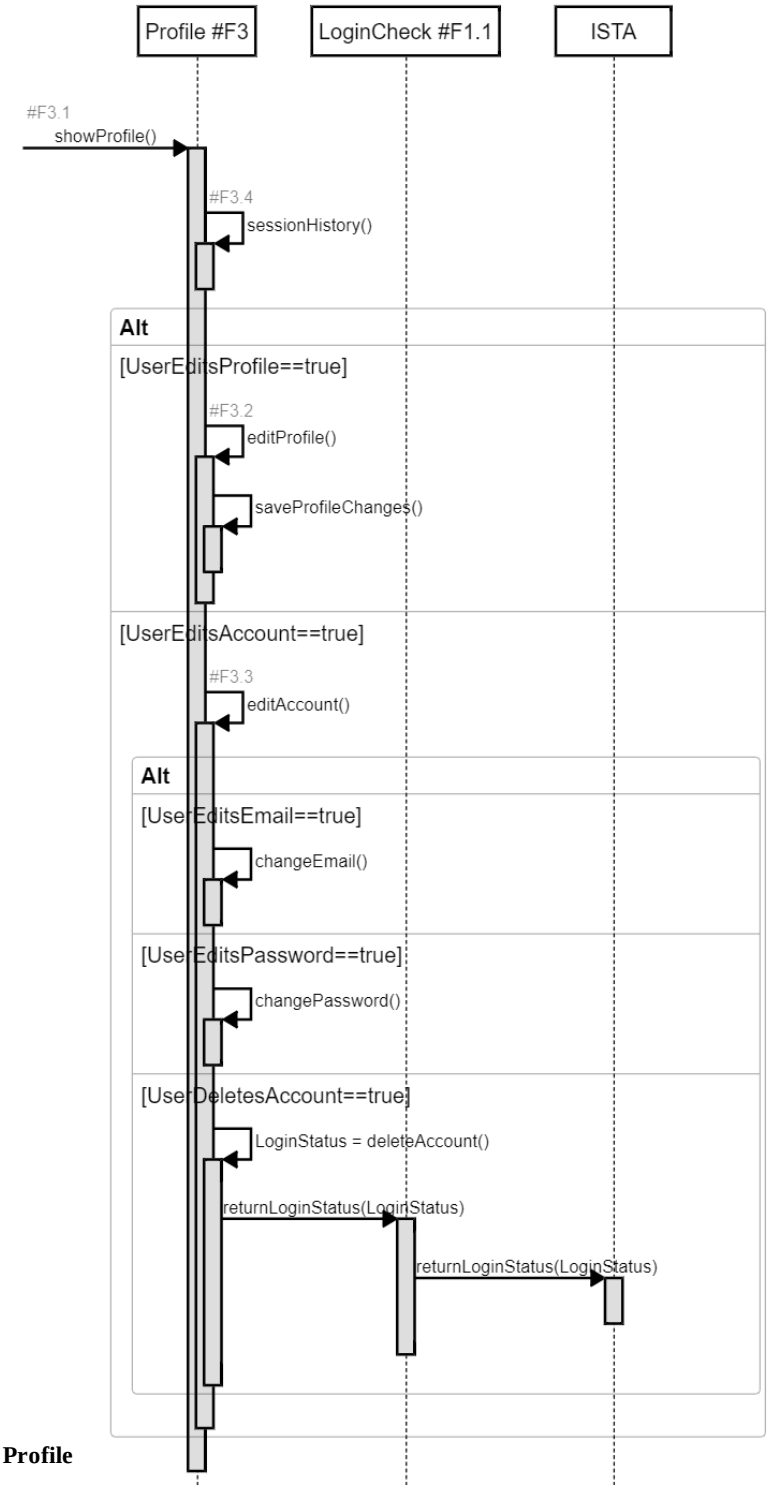

Figure 19 - Profile

6.5 Session

When the session class is called from the home screen, the user has decided that they want to start an inline skate session via startSession(). During the session, the skated time and distance are displayed in real time with showCurrentSkatedTime() and showCurrentSkatedDistance(), the latter also calling calculateDistance().

If the user chooses endSession(), then showSkatedTimeDistance() displays the total time and distance the user has reached in this session. Then these statistics are saved for the user's profile and they return to the home screen.

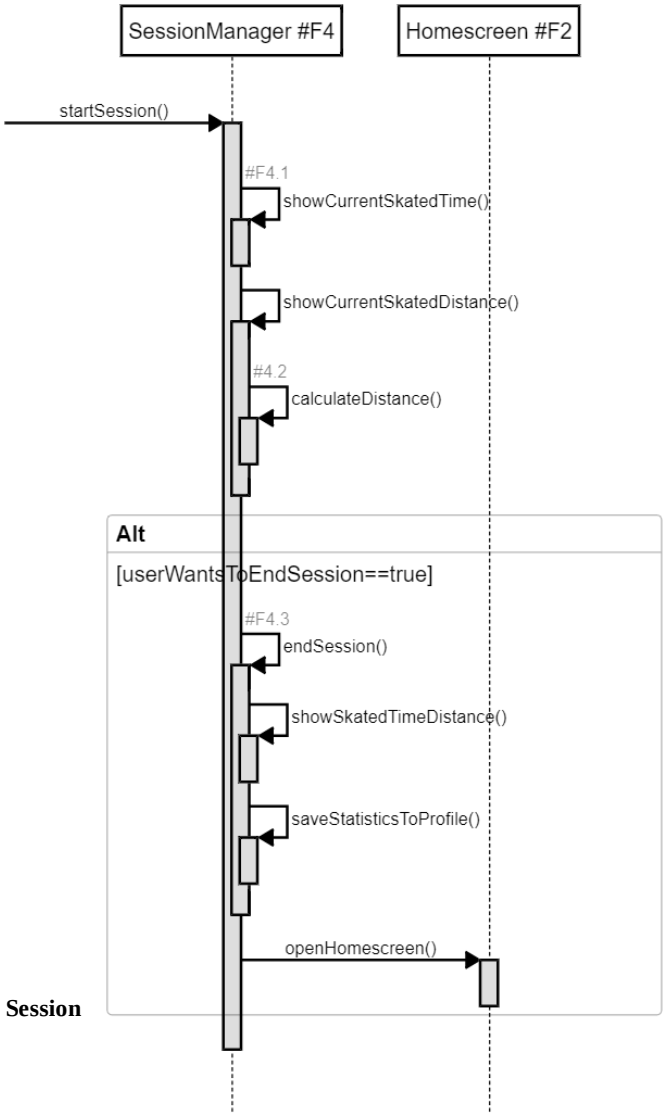

Figure 20 - Session

6.6 Leaderboard (with Sort List)

If showLeaderboard() is called by the user from the homescreen, then the leaderboard is visualised first with the call to sortList() from the SortList class afterwards, where the users in the leaderboard gets sorted #F6.1.

The user then has the choice between showOwnRank() and searchUser(), the former showing the rank of the user in the leaderboard and the latter offering the possibility to search for a specific inline skater in the leaderboard and relinquish its existence.

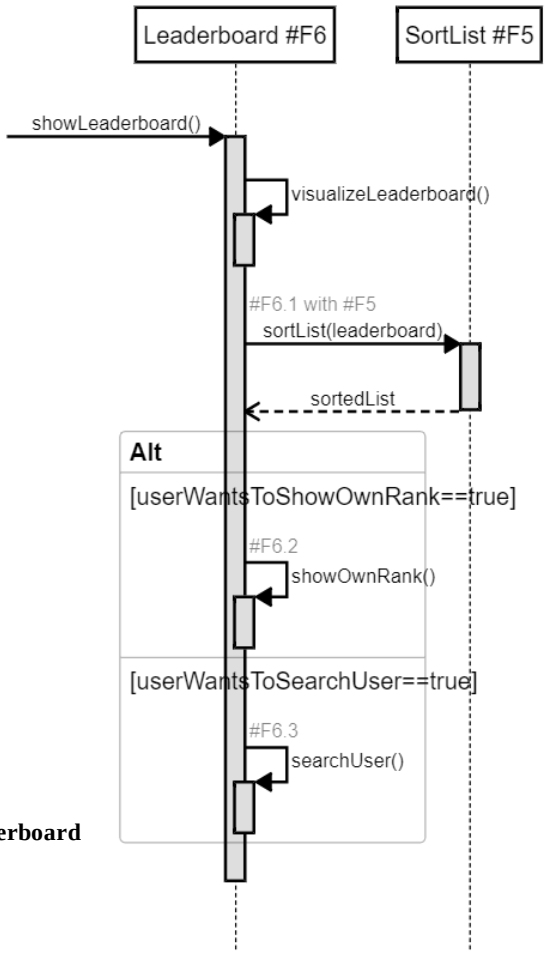

Figure 21 - Leaderboard

6.7 Friends (with Sort List)

The user decides to manage their friends, so he calls the showFriendsList() function from the homescreen, then the sortList(friends) function of the SortList class is called and returns the sorted list.

From there, the user can either call the functions addFriends(), showFriendRequests(), showFriendProfile() or chat().

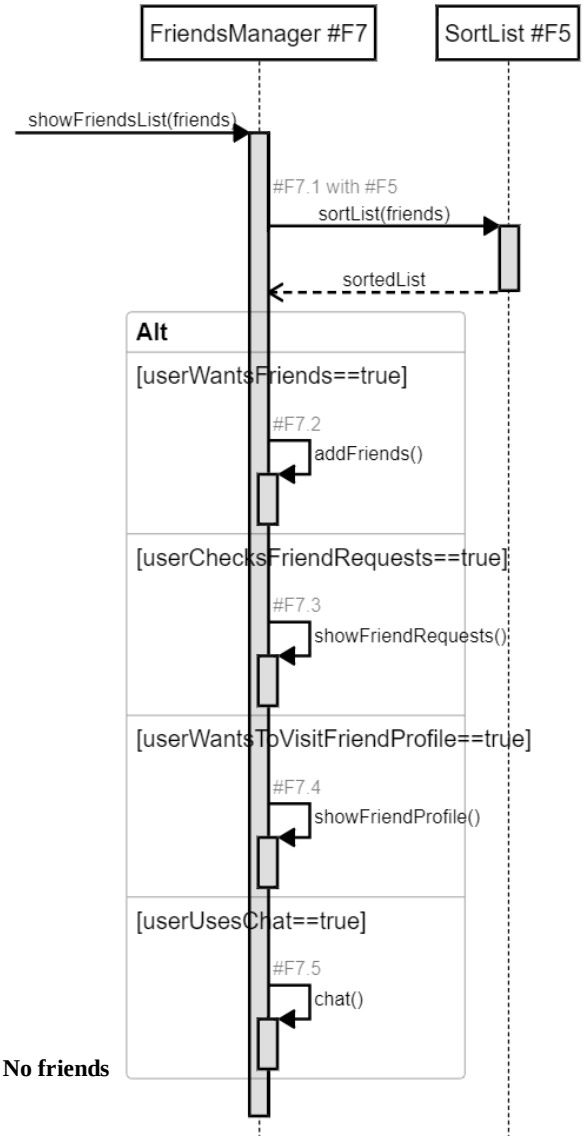

Figure 22 - No friends

Chapter 7 – Physical Deployment View

7.1 Broad infrastructure

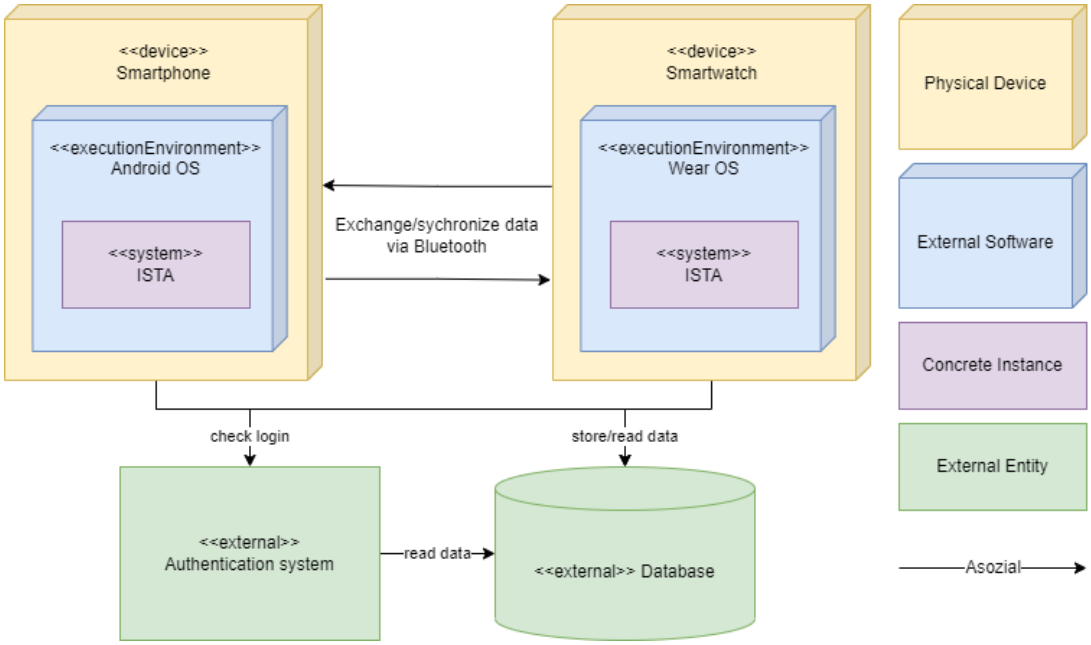

Figure 23 - Deployment View

The **device tier** describes the type of physical device on which the ISTA system can be run, for this the client must use either smartphone or -watch to use the mobile application. Furthermore, the Bluetooth connection of the devices is used for the exchange/synchronisation of data between these two devices.

The **external software tier** describes the environment in which the ISTA is to be executed. For the smartphone this is the Android operating system and for smartwatches the Wear OS.

The **concrete instance** describes the internal software that will be instantiated in the execution environment, that is the ISTA mobile application in this project.

The **external entity** describes the external systems that are outsourced for this plan, while the external database is mainly used by ISTA to store/read data and the external authentication system is used to verify and authenticate ISTA logins, for which the external database is also used as a reference point.

Chapter 8 – Cross-cutting Concepts

This table contains the tactics that can aid in the attainment of given QAs.

ID	QA	Sub QA	Tactics
#CC1	Usability	Learnability for Users or Devs (#Q1.1 #Q1.2 #CB3)	Support System Initiative: Maintain task model (determines context, like auto suggestion), -user model (represents user's knowledge, like language-learning apps) and -system model (determines current system's state and gives feedback, like progress bar). Support User Initiative: Provide the user with options to cancel, undo, pause/resume or aggregate. Google Developers (#D1 #Q1.2): Additional style guidelines to help design user-friendly GUIs.
#CC2	Modifiability	Portability of app (#Q2.3)	Defer binding: Parametrization (raising level of abstractness by introducing parameters to increase flexibility and defer binding).
#CC3	Modifiability	Scalability of app (#Q2.4)	Split Module: Modules with low cohesion and high coupling are split into submodules to achieve high cohesion and low coupling → reduction of cost for future changes.
#CC4	Interoperability	Communication between systems (#Q3.5)	Use an intermediary: Use APIs to establish communications between systems and for breaking dependencies between components.
#CC5	Availability	Fault recovery (#Q4.6)	Shift responsibility to external provider, because of #CT2, more at #R6.
#CC6	Security	Authentication (#Q5.7)	Shift responsibility to external provider, because of #CT1, more at #R5.
#CC7	Performance	System latency (#Q6.8)	Control Resource Demand: Manage work requests (manage event arrival- or sample rate) and prioritize events (rank events by importance).

Figure 24 - Tactics

Chapter 9 – Risks

For the technical risk assessment of the project, a meeting was held with all relevant stakeholders and the following risks were identified using the 3x3 risk matrix:

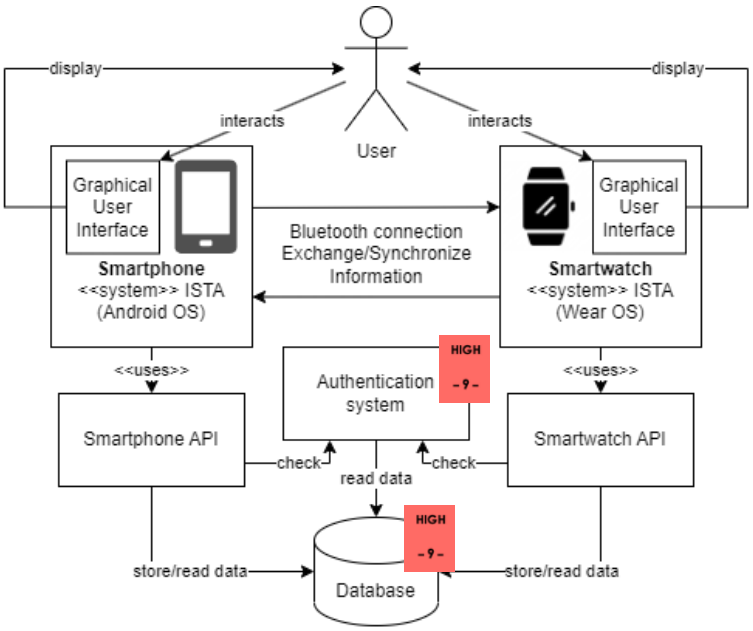

Figure 26 - Risk Voting

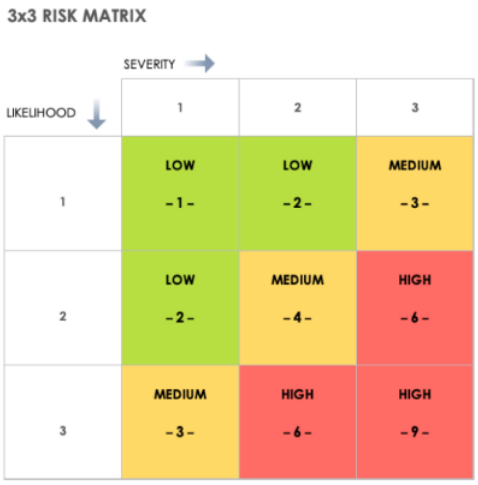

Figure 25 - Risk Matrix

As the following table shows, only #R5 and #R6 were approached, while #R1-4 were not addressed due to insufficient technical knowledge (#CB2 #CB3).

ID	Technical Entity	Risk-Level	Concern	Solution
#R1	ISTA (Android OS)	Unknown	// WIP	// WIP
#R2	ISTA (Wear OS)	Unknown	// WIP	// WIP
#R3	Smartphone API	Unknown	// WIP	// WIP
#R4	Smartwatch API	Unknown	// WIP	// WIP
#R5	Authentication System	High: 9	• Security: Single point of failure • #CT1	Outsource to an external provider to take care of the security concerns.
#R6	Database	High: 9	• Availability: Single point of failure • #CT2	Contract an external provider who ensures an availability of 99.99 %.

Figure 27 - Risk Table

Chapter 10 – Glossary

Term	Definition
UI	User interface
OS	Operating system
QA	Quality attribute
WIP	Work in progress
API	Application programming interface
ISTA	Inline Skate Tracker App (the project's mobile app)

Figure 28 - Abbreviations