Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Computer Science
Field of Study: Software Engineering and Technology

Mobile application for multiprojection system control

BACHELOR'S THESIS

Author:	Victoria Savvateeva
Supervisor:	Ing. Ivo Malý, Ph.D.
Date:	May 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Savvateeva** Jméno: **Victoria** Osobní číslo: **453254**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Mobilní aplikace pro ovládání multiprojekčního systému

Název bakalářské práce anglicky:

Mobile application for multiprojection system control

Pokyny pro vypracování:

Get acquainted with the project 5D projection system created for The City of Prague Museum (MMP) and with the first prototype of its control system [1]. Focus especially on the mobile application controlling the playback of presentation on the 5D projection system. Analyze issues of the user interface of original controlling application [1]. Develop updated set of requirements and design new version of application controlling the playback. During design follow User-Centered Design methodology and use prototyping tool to evaluate the design. Focus on devices with larger display (tablets). Finally, implement the application using React Native framework and other suitable libraries. Test the application of at least 4 presentations with different structure and configuration.

Seznam doporučené literatury:

1. O. Trojan, Řídící systém pro multiprojekci, Diplomová práce, České vysoké učení technické v Praze, 2020.
2. B. Fling, Mobile Design and Development, O&#39;Reilly Media, 2009
3. G. K. Mostefaoui, F. Tariq (ed.), Mobile Apps Engineering: Design, Development, Security, and Testing, CRC Press, 2018.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Ivo Malý, Ph.D., katedra počítačové grafiky a interakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **15.02.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **19.02.2023**

Ing. Ivo Malý, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.

Datum převzetí zadání	Podpis studentky

Declaration

I declare that I have elaborated the submitted Thesis myself and only used the resources quoted in the attached Bibliography list.

In Prague on ....
Victoria Savvateeva

Acknowledgements

I thank my supervisor, Ing. Ivo Malý, Ph. D., for giving me the opportunity to take part in the mentioned project, for his guidance and valuable advice whenever I needed. I also express my deepest gratitude to my family for supporting me throughout my studies.

Victoria Savvateeva

Abstract

A multiprojection control system for managing digital presentations in the City of Prague Museum is being developed at the Department of Computer Science. The system includes three components: a web administration tool for CRUD operations and data handling, a media player for performing the playback on several client devices and a mobile application for remote control and daily usage. While the web part is managed by the administrator, a mobile application is supposed to be used by the presenters (museum employees). Although the first version of the mobile application has been prototyped and implemented, it became clear that it needs to be redesigned and upgraded. This Thesis aims at creating a completely new version, both in terms of functionality and user interface design. The outcome is a tested and fully functional application, preceded by a high-fidelity prototype, also tested for its usability.

Keywords: mobile development, frontend development, multiprojection system, digital exhibitions, client application

Abstrakt

Na katedře počítačů se vyvíjí multiprojekční systém sloužící ke správě digitalizovaných výstav v Muzeu hlavního města Prahy, jako například Langweilův model Prahy. Systém se skládá ze tří komponent - webové aplikace pro řízení CRUD operací a ukládání dat, přehrávače pro reprodukci media souborů na několika klientských zařízeních a mobilní aplikace pro vzdálené ovládání a každodenní použití. Zatímco webová aplikace je řízená administrátorem, mobilní aplikace je určená pro předváděče prezentací (zaměstnance). Přestože první verze mobilní aplikace již byla implementována, při její vyhodnocení byla odhalená řada nedostatků. Tato Teze má za cíl vytvoření kompletně nové verze, v rámci jak funkcionality, tak i návrhu uživatelského rozhraní. Výsledkem je testovaná a plně funkční aplikace, které předchází high-fidelity prototyp, také testovaný na použitelnost.

Klíčová slova: mobilní vývoj, frontend vývoj, multiprojekční systém, digitalní výstavy, klientská aplikace

Table of Contents

List of Tables

List of Figures

List of Listings

1. Introduction

This Bachelor Thesis follows the 5D Projection System project carried out at the Department of Computer Science at CTU FEE. In terms of the project, a multiprojection system allowing both autonomous and guided presentation of Langweil model of Prague on several client devices with 1-4 connected displays was developed, including a web-based console for maintenance of the exhibition's playlists and the first version of a mobile application for remote control. The system is designed as a lightweight and portable solution for the City of Prague Museum and its main purpose is to establish an innovative way of creating and implementing educational programs for children and students. I focused on design, implementation, and testing of a brand-new version of a mobile application with some significant user interface and functionality improvements. The developed product is scaled for larger devices such as tablets and is supposed to be used by the City of Prague Museum employees, responsible for managing, controlling, and presenting the digital exhibitions to the visitors, the Langweil Model of Prague in particular. The objective of the Thesis to report the results achieved within the development process including all its phases covered step by step. In this chapter, I define the basic terms and methodology concepts used. The Motivation Section describes the main ideas behind the Thesis realisation and states its goals. In the Analysis, I provide a detailed overview of the existing alternatives available on the market, compare its core concepts with the motivation behind our project and, as a result, define the newly shaped requirements and use cases. The Design Chapter represents my solution of the user interface and the consequently originated high-fidelity prototype. The Implementation chapter covers technical aspects, while the following, Evaluation and Testing, provides the usability outcome. Finally, I make a conclusion and address the issues that could be processed in the future.

1. 1 Terms and Methodology

1. 1. 1 User Centered Design

The term "User-Centered Design" was coined by Rob Kling in 1977 [1] and later popularised in the Donald A. Norman's book User-Centered System Design: New Perspectives on Human-Computer Interaction in 1986 [2].

The User-centered design (UCD) process outlines the phases throughout a design and development life cycle all while focusing on gaining a deep understanding of who will be using the product. The international standard 13407 is the basis for many UCD methodologies [3].

The process may be split into the following phases (Figure 1 [4]):

- **Specify the context of use**: Identify the people who will use the product, what they will use it for, and under what conditions they will use it.

- **Specify requirements**: Identify any business requirements or user goals that must be met for the product to be successful.

- **Create design solutions**: This part of the process may be done in stages, building from a rough concept to a complete design.

- **Evaluate designs**: Evaluation - ideally through usability testing with actual users - is as integral as quality testing is to good software development.

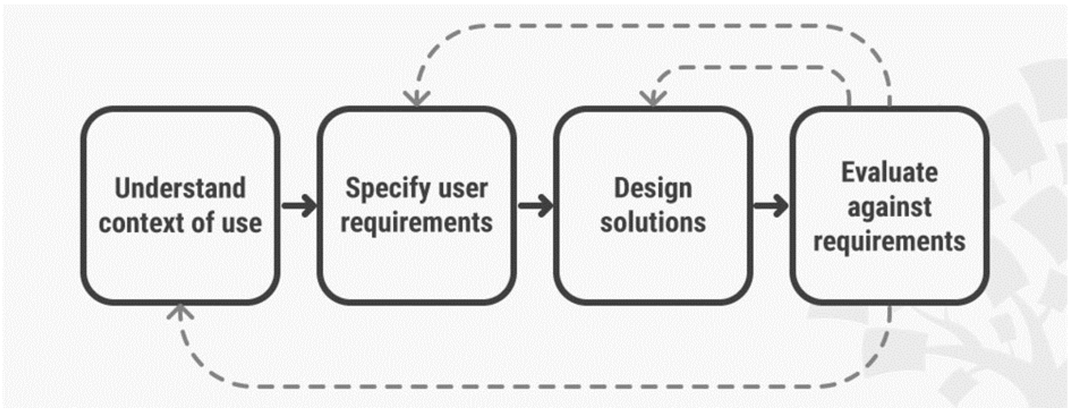

Understand context of use
Specify user requirements
Design solutions
Evaluate against requirements

Figure 1*. Introduction - User-Centered Design phases [4]*

The mentioned techniques are considered and applied throughout my work.

1. 1. 2 Reactive Programming

Reactive Programming is a paradigm in which declarative code is issued to construct asynchronous processing pipelines. It is programming with asynchronous data streams that sends data to a consumer as it becomes available, which enables developers to write code that can react to these state changes quickly and asynchronously.

A stream is a sequence of ongoing events (state changes) ordered in time (Figure 2). Streams can emit three different things: a value (of some type), an error, or a “completed” signal. The events are captured asynchronously, by defining a function that will execute when a value is emitted, another function when an error is emitted, and another function when ‘completed’ is emitted. “Listening” to the stream is called subscribing. The functions we are defining are observers. The stream is the subject (or “observable”) being observed.

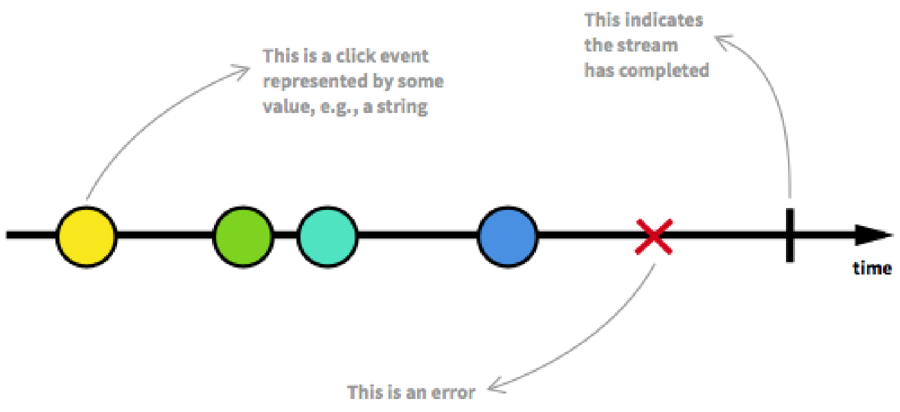

This is a click event represented by some value, e.g., a string
This indicates the stream has completed
time
This is an error

Figure 2. Introduction - Stream as a sqeuence of asynchronous events

Reactive programming can be a useful implementation technique for managing internal logic and data flow transformation locally within components through this asynchronous and non-blocking execution [5].

1. 1. 2. 1 Reactive Extensions

Reactive programming deals with data flow and automatically propagates changes via the data flow. This paradigm is implemented by Reactive Extensions.

Reactive extensions enable imperative programming languages to compose asynchronous and event-based programs by using observable sequences. In other words, it enables your code to create and subscribe to data streams named *observables*. Reactive extensions combine the observer and iterator patterns and functional idioms to give you a sort of toolbox, enabling your application to create, combine, merge, filter, and transform data streams [5].

One of the examples is RxJS, which I used in my implementation (see "Implementation" section).

2. Motivation

Within a multiprojection system used for presentation purposes, given our project as an example, a remote control device is necessary. It is important that the developed application is well-designed, robust, and highly reliable, as the presenter always has limited time to perform and there is no place for a technical problem. He cannot postpone a presentation due to an application flaw and there is no time for bug fixing.

My application is scaled for tablets due to certain reasons. One major advantage is resolution and screen size, which makes it suitable for targeting any audiences from the young to the oldest. Middle-aged or elederly people may have trouble reading small text during their presentations, so the larger screen size makes them feel way more comfortable while working. Another great option is the battery life, which is significantly extended in comparison with regular smartphones. It is crucial that the device does not run out of battery during presentation and is able to stay awake during the entire working day – there could be many visitor groups coming to the Museum.

2. 1 Issues of the existing version

The primary source of motivation is the issues behind the user interface of the proposed version. After conducting an analysis, the following problems were determined.

Firstly, the design lacks contrast. Any application should come with a clear and fresh contrast, so the information is more readable and understandable, and a user is confident to proceed with operationa. Poor colour choices result in difficulties reading and makes the application aesthetically unpleasant, which can further deter a user form using it.

Secondly, the overall style is inconsistent. The Homepage, including display sets, lists its items in a grid, while the following similar section, Display Tracks, is designed as a list, which results in a visual conflict and confusion. Although it is arguable whether the main body of both sections correlate directly, in my opinion, they should be designed in the same way.

The next problem comes from the Heuristic analysis first set of rules - which is visibility of the system status – and represent wrong heading description – Homepage in particular. It does not say anything about the page contents. It should be named as Display Sets.

2. 2 Goals

The goal of the Thesis is to design and build a product, which will bring a value to the whole system and to the City of Prague Museum. This can be split into the following phases:

Goal 1. Design an upgraded version of a mobile application scaled for tablets, eliminate all the problems mentioned above and represent it with a high-fidelity clickable prototype.

Goal 2. Implement the designed solution through creating a multi-platform application mainly targeted at Android devices.

Goal 3. Test the application on four different presentations.

3. Analysis

The concept of online viewing first started to blossom in the late 2010s as artists like David Zwirner and Pace Gallery began to offer access to their pricey Koons and Hockney works in their online viewing rooms, which used a log-in system. Now more and more galleries and museums around the globe are turning to digital exhibitions and offering virtual tours [6]. Representing art works digitally makes a huge contribution to broadening and enhancing a museum's work outreach to the public with its easy access via Internet or mobile applications and creates a great additional base for education.

Multi-projection systems are systems with multiple projectors and multiple projection surfaces. The term is strongly coupled with video mapping, which is often combined with an audio track in order to create a more interactive user experience [7].

3. 1 Existing Alternatives

I conducted an analysis on existing mobile applications dedicated to digital presentations. Although no exact matches, including the same target audience and overall purspose were found, three similarly structured Google Play Store apps, all of which have 4+ stars ranking were analyzed. All of them share the same objective, UI and do not imply any remote video playback control as they use audio tracks instead.

3. 1. 1 Schweizerisches Nationalmuseum app – Swiss National Museum

Developer and Publisher: Swiss National Museum (Schweizerisches Nationalmuseum)

Platform: Android

Link:
https://play.google.com/store/apps/details?id=com.landesmuseum.snmch&hl=en_US&gl=US

The application is in several languages. Besides the exhibitions themselves, includes a brief text preview and detailed 3D navigation from the very entrance to the exhibits. Each item is provided with an audio recording telling the history of the given showpiece. A user can pause the recording, play back and forward.

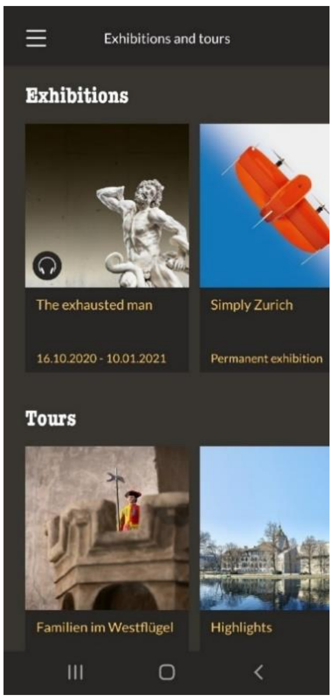
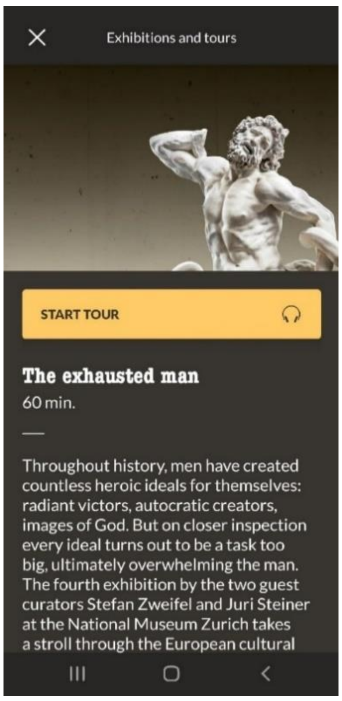


***Figure 3**. Main page*



The exhausted man
60 min.

Throughout history, men have created countless heroic ideals for themselves: radiant victors, autocratic creators, images of God. But on closer inspection every ideal turns out to be a task too big, ultimately overwhelming the man. The fourth exhibition by the two guest curators Stefan Zweifel and Juri Steiner at the National Museum Zurich takes a stroll through the European cultural

***Figure 4**. Exhibition preview*

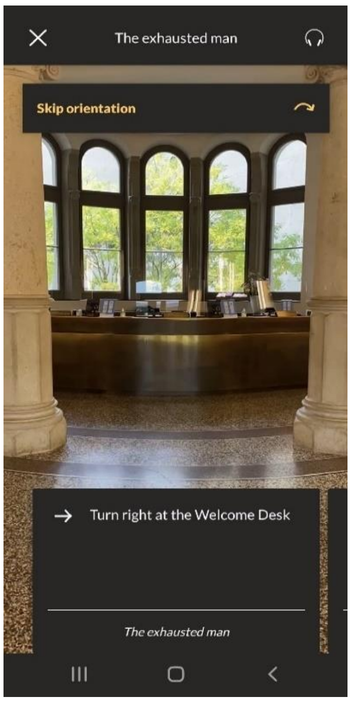


***Figure 5**. Tour started*

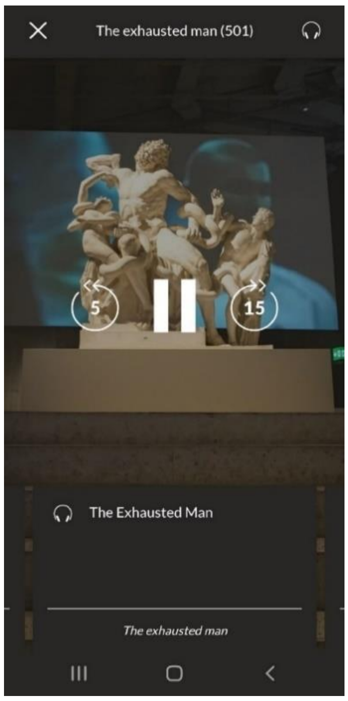


***Figure 6.** Playing audio track*

Concept similarities:

- same objective – digital museum app
- a sliding menu on the left
- same component structure – a media set (playlist) consisting of tracks
- same media set preview principle

Concept differences:

- no user authentication needed
- several languages support
- no videos – 3D navigation and audio instead
- no autonomous mode
- no color switching
- playback and playforward functions
- no shuffle and loop

3. 1. 2 Museum Barberini

Developer and Publisher: Museum Barberini gGmbH
Platform: Android
Link:
https://play.google.com/store/apps/details?id=com.barberini.museum.barberinidigital&hl=en&gl=US

The application has a completely different and simplified minimalistic-styled UI and encapsulates rich exhibition database targeted at both adults and children. Only audio tours are supported. Two of the outlined features, such as information about the exhibitions or a biographies collection, can be useful and are later to be considired for adding as nice-to-have features in our application.

Features:

- Audio tours for children and adults
- Information about current and upcoming exhibitions
- 360° panoramas with multimedia content
- Tours archive
- Biographies
- Integrated ticketing
- Videos about artists and curators
- Opening times, offers, prices, location
- Information about the history of the building, the art and the founder
- Newsletter subscription

3. 1. 3 Rijksmuseum

Developer and Publisher: Rijksmuseum
Platform: Android
Link:
https://play.google.com/store/apps/details?id=nl.rijksmuseum.mmt&hl=en&gl=US

Offers paintings audio tours and a very comfortable dark-colored UI creating an immersive digital museum antiquity environment.

Features:

- Follow a route or search for the numbers accompanying the works of art
- Iinteractive floor plans and directions
- 3D audio clips and animations
- Feedback or questions

3. 1. 4 Summary

Android market has a lot to offer for museum visitors of any age. A similar concept is widely used among developers and is well-known among users. All of the mentioned products are unified by the idea of creating a vivid, enjoyable user experience within a digitalized museum environment and may serve as a source of inspiration as well as provide useful tips for further functionality. However, all of the mentioned above products are targeted at museum visitors rather than internal employees and do not implement any remote control device functionality. Our application will primarily be used by the museum workers to remotely launch and handle presentations.

3. 2 Requirement Analysis and Specifications

Here, I apply the first two User-Centered Design principles, which state:

- **Specify the context of use**: Identify the people who will use the product, what they will use it for, and under what conditions they will use it.
- **Specify requirements**: Identify any business requirements or user goals that must be met for the product to be successful.

As it was previously mentioned, the people who will use the product are the Museum employees, or those who are supposed to perform presentations. They will use the application daily for controlling the exhibitions playback performed on four client devices (screens) remotely, as well as managing video chapters. Requirement analysis is critical to the success or failure of a systems or software project [8]. It is the process of determining user expectations for a new or modified product, involving defining needs and objectives in the context of planned customer use, environments, and identified system characteristics to determine requirements for system functions.

3. 2. 1 Requirements

One of the requirement analysis purposes is to define functional and performance requirements based on customer provided measures of effectiveness. By community agreement, Requirements Analysis should result in a clear understanding of:

- functions: what the system has to do,
- performance: how well the functions have to be performed,
- and interfaces: Environment in which the system will perform [9].

Requirements below are categorized according to the method described in the “Systems Engineering Fundamentals” book by Defense Acquisition University Press published in 2001 [9]. Based on the given business goals and demands, a set of Functional (Table 1), Performance (Table 2), Design (Table 3) and Derived (Table 4) requirements was defined.

3. 2. 1. 1 Functional

F1	System must provide connection to the exhibition using IP address and HTTP requests for connection establishment.
F2	System must provide user authentication and authorization.
F3	System must provide requesting/querying available display sets.
F4	System must provide Controlling the video players through HTTP requests to the server.
F5	System must notify a user of errors and system state.
F6	System must support different light conditions.
F7	System must provide viewing all the tracks within a given display set.
F8	System must provide that the control panel of the exhibition contains Shuffle, Previous track, Pause/Play, Next track, Loop current track functions within a given display set.
F9	System must provide Display track chapters creation.
F10	System must Playback from the start of a chosen chapter (e.g. chapter 0:40 - 1:10).
F11	System must provide viewing video files list within a given display track.
F12	System must provide Playback status querying and displaying periodically.
F13	System must have an Autonomous mode.

***Table 1**. Functional requirements*

3. 2. 2. 2 Performance

P1	System shall have high performance.
P2	System shall have low latency and response time.

Table 2. *Performance requirements*

3. 2. 2. 3 Design

D1	System shall have a user-friendly and intuitive UI following UCD principles.
D2	System shall run on Android 5.0 and higher.
D3	System shall be designed following the SoC principle.
D4	System shall provide responsive design.
D5	System shall provide that display sets and tracks have both list and grid view.

Table 3. *Design requirements*

3. 2. 2. 4 Derived

De1	Whatever user input must be handled and validated.
De2	Data querying is done through a separated provider responsible only for server HTTP requests and serialization of the received data.

Table 4. *Derived requirements*

3. 2. 2 Technical Specifications

The application will be developed for the Android platform and will be based on Separation of Concerns (using the BLoC pattern) and Material Design principles. The application will be scaled for tablets.

Robustness and reliability are put as main priorities while developing the further application. Due to this, React Native was chosen as a framework because of its performance and maturity level.

BLoC pattern handles the business logic of the application, separating the responsibilities of individual components. Its principles, initially developed for Flutter, will be applied to React Native environment.

Platform: Android

Programming Language: TypeScript

Framework: React Native with usage of ReactiveX and RxJS for reactive programming and asynchronous programming with observable streams.

3. 3 Use Cases

The Use Case analysis is a bridge between the users (Actors) and the modelers to determine the needs that the system must satisfy. It captures the functionality and needs that the users of the new system want to have the system do. It also is done in a way that the both the team and interested users can review and criticize [10].

The structure of each Use Case (Tables 5-19) comprises:

- an identifier consisting of an application type (MOB as mobile) and a number
- fulfilled requirements
- a goal to be achieved
- an Actor (User)
- the state of the system and conditions for the Use Case to be performed
- the state of the system after successful completion of the Use Case
- steps to be covered (Main/Alternate flow)
- exceptions that may occur and their handling.

ID:	**MOB-1**
Requirement:	**F1, F2, F3, De1**
Objective:	Signing in
Actor:	E*
Precondition:	-
Postcondition:	E is authenticated and authorized. U is redirected to the homepage with available display sets
Main flow:	1) Open the app. 2) See the login page with login and password required. 3) Enter the login and password and click on the “Sign In” button.
Exceptions:	1.a) Connection is not established or a server-side error occurred. Related notification is shown. Application exits. The use case ends.
Alternate flow:	3.a) Wrong input. Related notification is shown. Repeat 2) and 3). 3.b) Wrong data. Related notification is shown. Repeat 2) and 3). 3.b) No sets are available. Reload the page or exit.

***Table 5**. Use Cases - Signing in*

**E = Employee*

ID:	**MOB-2**
Requirement:	**F3, F7**
Objective:	Open a display set
Actor:	E
Precondition:	E is logged in, display set list is present
Postcondition:	Display set page is opened, all display tracks are listed
Main flow:	Click on a display set.
Exceptions:	The display set cannot be opened. Related notification is shown. The use case ends.
Alternate flow:	

***Table 6**. Use Cases - Opening Set*

ID:	**MOB-3**
Requirement:	**F4, F5, F8, F12**
Objective:	Play a display set
Actor:	E
Precondition:	E is logged in, display set is opened
Postcondition:	Display set is being played from the first track, playback status bar is showing valid playback status
Main flow:	Click on the “Start” button
Exceptions:	The display set cannot be played. Related notification is shown. The use case ends.
Alternate flow:	

Table 7. *Use Cases - Playing Set*

ID:	**MOB-4**
Requirement:	**F4, F5, F8, F12**
Objective:	Pause a display set
Actor:	E
Precondition:	E is logged in, display set is opened, display set is being played
Postcondition:	Display set is paused
Main flow:	Click on the “Pause” button
Exceptions:	The display set cannot be paused. Related notification is shown. The use case ends.
Alternate flow:	

Table 8. *Use Cases - Pausing Set*

ID:	**MOB-5**
Requirement:	**F4, F5, F8**
Objective:	Play the next track
Actor:	E
Precondition:	E is logged in, display set is opened, display set is being played/paused
Postcondition:	The next track is being played
Main flow:	Click on the “Next track” button
Exceptions:	The next track cannot be played. Related notification is shown. The use case ends.
Alternate flow:	

Table 9. *Use Cases - Playing Next Track*

ID:	**MOB-6**
Requirement:	**F4, F5, F8**
Objective:	Play the previous track
Actor:	E
Precondition:	E is logged in, display set is opened, display set is being played/paused
Postcondition:	The previous track is being played
Main flow:	Click on the “Previous track” button
Exceptions:	The previous track cannot be played. Related notification is shown. The use case ends.
Alternate flow:	

Table 10. *Use Cases - Playing Previous Track*

ID:	**MOB-7**
Requirement:	**F4, F5, F8**
Objective:	Shuffle tracks within a given display set
Actor:	E
Precondition:	E is logged in, display set is opened
Postcondition:	The tracks are ordered randomly within a given display set
Main flow:	Click on the "Shuffle" button
Exceptions:	The display set cannot be shuffled. Related notification is shown. The use case ends.
Alternate flow:	

***Table 11**. Use Cases - Shuffling Tracks*

ID:	**MOB-8**
Requirement:	**F4, F5, F8**
Objective:	Loop currently played track
Actor:	E
Precondition:	E is logged in, display set is opened, display set is being played/paused
Postcondition:	The current track is being replayed when finished
Main flow:	Click on the "Loop current track" button
Exceptions:	The currently played track cannot be looped. Related notification is shown. The use case ends.
Alternate flow:	

***Table 12**. Use Cases - Looping Track*

ID:	**MOB-9**
Requirement:	**F4, F11**
Objective:	View video files list within a given display track
Actor:	E
Precondition:	E is logged in, display set is opened
Postcondition:	The video files belonging to particular screens are shown
Main flow:	Click on the "Open track" button
Exceptions:	The currently played video files cannot be shown. Related notification is shown. The use case ends.
Alternate flow:	

***Table 13**. Use Cases - Viewing Video files*

ID:	**MOB-10**
Requirement:	**F4, F9**
Objective:	Create a video chapter within a given display track
Actor:	E
Precondition:	E is logged in, display set is opened
Postcondition:	A video chapter is created within a given display track
Main flow:	1) Click on the "Create chapter" button. 2) Choose starting time. 3) Choose ending time. 4) Click "Save".
Exceptions:	The chapter cannot be saved. Related notification is shown. The use case ends.
Alternate flow:	

***Table 14**. Use Cases - Creating chapter*

ID:	**MOB-11**
Requirement:	**F4, F10**
Objective:	Play the current track from the stated time
Actor:	E
Precondition:	E is logged in, display set is opened, a video chapter is created within a given display track
Postcondition:	Current track is being played from the stated time
Main flow:	Click on a chosen a chapter
Exceptions:	The chapter cannot be played. Related notification is shown. The use case ends.
Alternate flow:	

Table 15*. Use Cases - Playing chapter*

ID:	**MOB-12**
Requirement:	**D5**
Objective:	View display sets as a grid
Actor:	E
Precondition:	E is logged in
Postcondition:	Display sets are shown as a grid
Main flow:	Click on “Grid view” button
Exceptions:	The display sets cannot be shown as a grid. Related notification is shown. The use case ends.
Alternate flow:	

Table 16*. Use Cases - Switching Sets to grid view*

ID:	**MOB-13**
Requirement:	**F6**
Objective:	Turn on dark mode
Actor:	E
Precondition:	-
Postcondition:	Dark mode is activated
Main flow:	Click on “Dark mode” button
Exceptions:	Dark mode cannot be activated. Related notification is shown. The use case ends.
Alternate flow:	

Table 17*. Use Cases - Switching to Dark mode*

ID:	**MOB-14**
Requirement:	**F13**
Objective:	Run the autonomous mode (loop a given display set)
Actor:	E
Precondition:	E is logged in
Postcondition:	Autonomous mode is run
Main flow:	1) Choose Set to be set as autonomous 2) Click on the “Run Autonomous” button
Exceptions:	Autonomous mode cannot be run. Related notification is shown. The use case ends.
Alternate flow:	

Table 18*. Use Cases - Running autonomous*

ID:	**MOB-15**
Requirement:	**F2**
Objective:	Log out
Actor:	E
Precondition:	E is logged in
Postcondition:	E is logged out and redirected to the Login page. Input data are prefilled.
Main flow:	Click on the "Logout" button on the right top menu.
Exceptions:	Logout cannot be performed. Nothing happens.
Alternate flow:	

***Table 19**. Use Cases - Logging out*

Alternatively, for visualisation purposes, I provide a UML diagram (Figure 7) drawn using the Sparx Enterprise Architect tool (version 15.2).

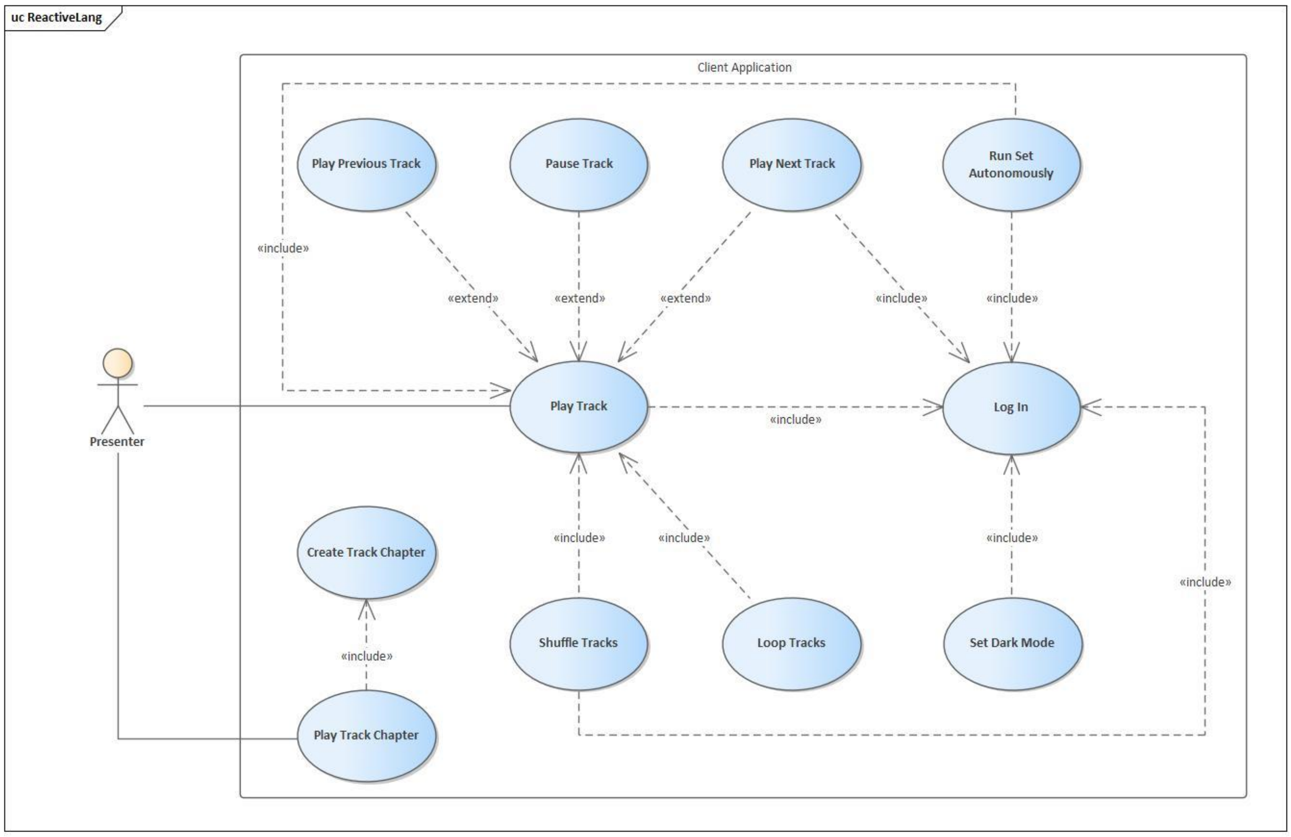


***Figure 7**. Use Cases - UML diagram*

4. Design

The last two User-Centered Design principles are applied in this part. They claim:

- **Create design solutions**: This part of the process may be done in stages, building from a rough concept to a complete design.
- **Evaluate designs**: Evaluation - ideally through usability testing with actual users - is as integral as quality testing is to good software development [3].

4. 1 Prototype Description

A high-fidelity prototype based on the Material Design principles was designed in Figma. Material is an adaptable system of guidelines, components, and tools that support the best practices of user interface design [11].

4. 1. 1 Launch & Authentication

The application starts with a City of Prague Museum logo and establishes server connection (Figure 8). Once connected, it proceeds to a login page (Figure 9).

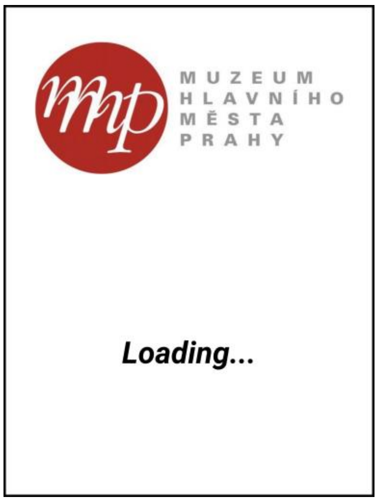


***Figure 8**. Prototype - Intro*

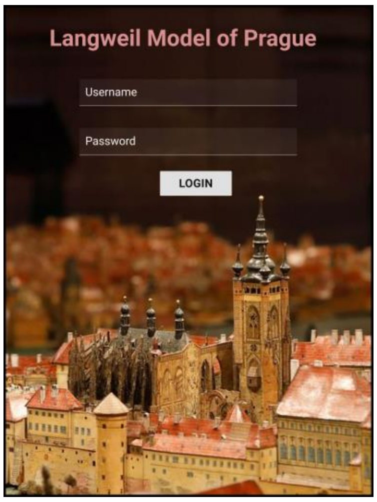


***Figure 9**. Prototype - Authentication*

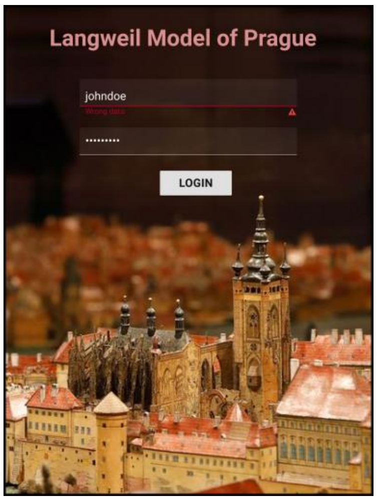


***Figure 10**. Prototype - Wrong user data*

4. 1. 2 Homepage with Display Sets

Display Sets list is shown with a checkbox and a disabled Autonomous mode button (Figure 11). When checkboxed, a number of Display Sets may be run autonomously (playing continuously without any user interaction, Figure 12). Alternatively, a Display Track list opens by clicking on a Display set name (Figure 13).

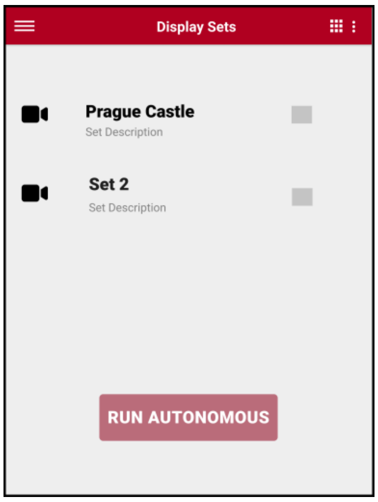


Figure 11. Prototype - Sets list

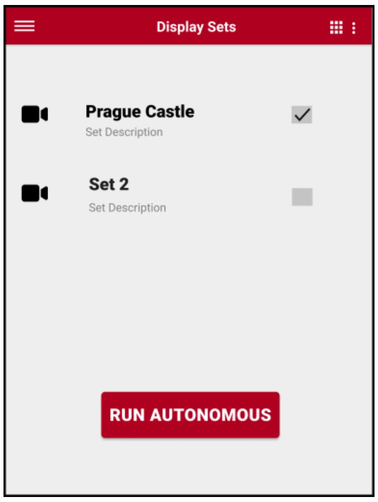
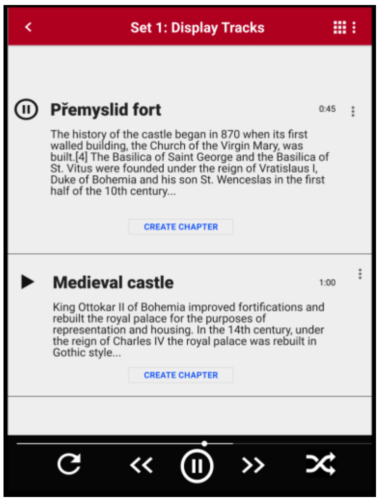


Figure 12. Prototype - Set checkboxed



The history of the castle began in 870 when its first walled building, the Church of the Virgin Mary, was built.[4] The Basilica of Saint George and the Basilica of St. Vitus were founded under the reign of Vratislaus I, Duke of Bohemia and his son St. Wenceslas in the first half of the 10th century...

King Ottokar II of Bohemia improved fortifications and rebuilt the royal palace for the purposes of representation and housing. In the 14th century, under the reign of Charles IV the royal palace was rebuilt in Gothic style...

Figure 13. Prototype - Run Autonomous / Play

Settings (Figure 14), dark mode are available (Figure 15). A Grid view can be switched when viewing the Sets (Figure 16).

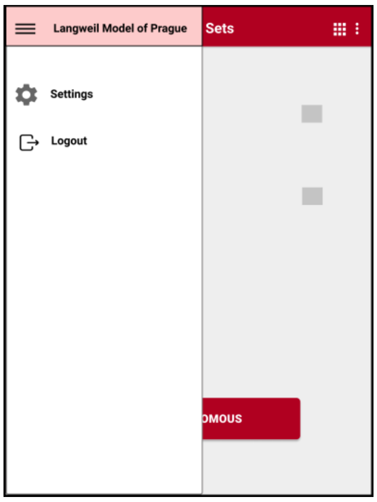


Figure 14. Prototype - Sliding menu

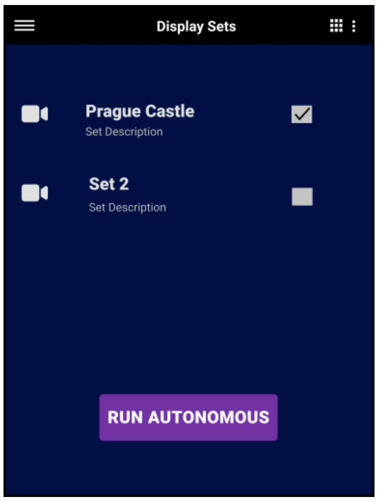


Figure 15. Prototype - Dark mode

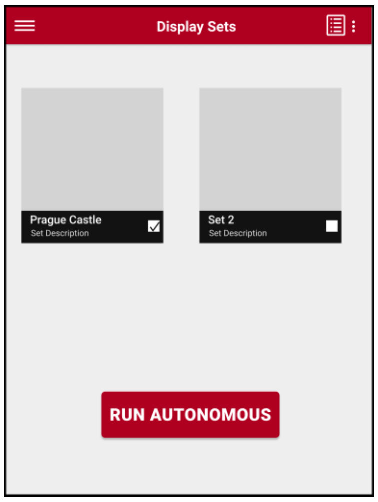


Figure 16. Prototype - Sets grid

4. 1. 3 Playing Display Tracks

While playing Display Tracks, a track with information about its video files being played on certain client devices can be opened and viewed (Figure 17).

Regardless of if the track is being played or not, a chapter can be added by specifying a name, start time and end time (Figures 18 and 19).

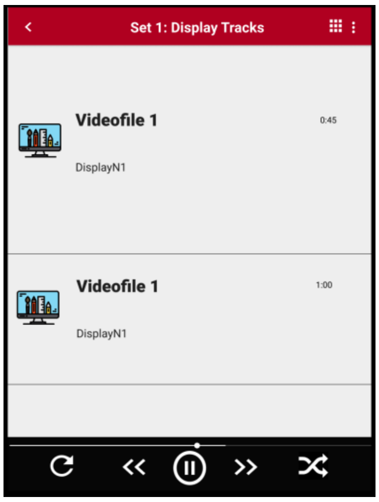
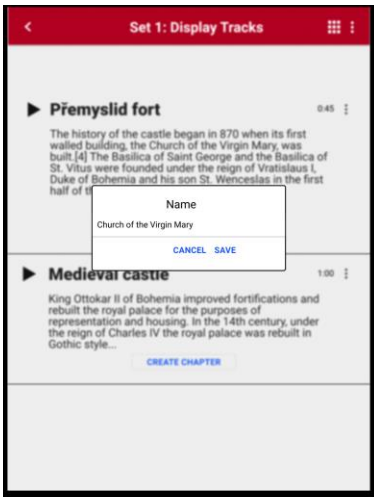


Figure 17. *Prototype - Open Track*

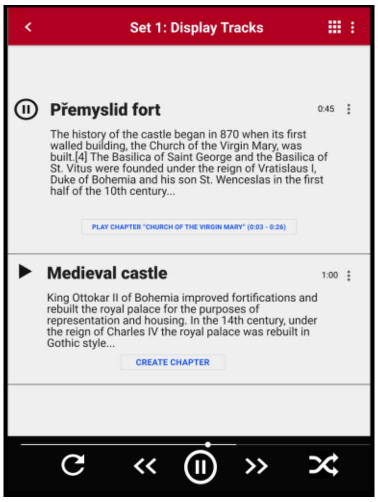


The history of the castle began in 870 when its first walled building, the Church of the Virgin Mary, was built.[4] The Basilica of Saint George and the Basilica of St. Vitus were founded under the reign of Vratislaus I, Duke of Bohemia and his son St. Wenceslas in the first half of th

Church of the Virgin Mary

King Ottokar II of Bohemia improved fortifications and rebuilt the royal palace for the purposes of representation and housing. In the 14th century, under the reign of Charles IV the royal palace was rebuilt in Gothic style...

Figure 18. *Prototype - Name chapter*



The history of the castle began in 870 when its first walled building, the Church of the Virgin Mary, was built.[4] The Basilica of Saint George and the Basilica of St. Vitus were founded under the reign of Vratislaus I, Duke of Bohemia and his son St. Wenceslas in the first half of the 10th century...

King Ottokar II of Bohemia improved fortifications and rebuilt the royal palace for the purposes of representation and housing. In the 14th century, under the reign of Charles IV the royal palace was rebuilt in Gothic style...

Figure 19. *Prototype - Chapter added*

4. 2 Heuristic Analysis (Usability Testing)

Heuristic Analysis (or Heuristic Evaluation) was chosen as a usability testing method. Heuristic evaluation involves having a small set of evaluators examine the interface and judge its compliance with recognized usability principles (the "heuristics") [12]. It is a helpful technique for identifying and specifying usability problems and as, a result, improving future versions of the product.

Analysis below is based on the "10 Usability Heuristics for User Interface Design" developed by Jakob Nielsen [13]. The template structure and question types reference Xerox Usability Checklist [14].

4. 2. 1 Visibility of system status

The design should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time.

Question	Answer
Does every display begin with a title or header that describes screen contents?	Yes
Is there a consistent icon design scheme and stylistic treatment across the system?	Yes
Is a single, selected icon clearly visible when surrounded by unselected icons?	Yes
Do menu instructions, prompts, and error messages appear in the same place(s) on each menu?	Yes
Is there some form of system feedback for every operator action?	No

Table 20. *Heuristics - Visibility of system status*

4. 2. 2 User control and freedom

Users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted action without having to go through an extended process.

Question	Answer
When a user's task is complete, does the system wait for a signal from the user before processing?	No
Are users prompted to confirm commands that have drastic, destructive consequences?	Not considered
Can users cancel out of operations in progress?	Yes

***Table 21**. Heuristics - User control and freedom*

4. 2. 3 Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform and industry conventions.

Question	Answer
Are icons labeled?	No
Are there no more than twelve to twenty icon types?	Yes
Does each window have a title?	Yes
Does the menu structure match the task structure?	Yes

***Table 22**. Heuristics - Consistency and standards*

4. 2. 4 Error prevention

Good error messages are important, but the best designs carefully prevent problems from occurring in the first place. Either eliminate error-prone conditions, or check for them and present users with a confirmation option before they commit to the action.

Question	Answer
Are instructions given to the user when taking ambiguous actions?	No

***Table 23**. Heuristics - Error prevention*

4. 2. 5 Recognition rather than recall

Minimize the user's memory load by making elements, actions, and options visible.

Question	Answer
Has the same color been used to group related elements?	Yes
Is color coding consistent throughout the system?	Yes

***Table 24**. Heuristics - Recognition rather than recall*

4. 2. 6 Flexibility and efficiency of use

Shortcuts — hidden from novice users — may speed up the interaction for the expert user such that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Question	Answer
Does the system allow novice users to enter the simplest, most common form of each command, and allow advanced users to add parameters?	Yes
Does the system offer "find next" and "find previous" shortcuts for database searches?	Yes

Table 25*. Heuristics - Flexibility and efficiency of use*

4. 2. 7 Aesthetic and minimalist design

Interfaces should not contain information which is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility.

Question	Answer
Does each icon stand out from its background?	Yes
Is the structure of a data entry value consistent from screen to screen?	Yes

Table 26*. Heuristics - Aesthetic and minimalist design*

4. 2. 8 Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no error codes), precisely indicate the problem, and constructively suggest a solution.

Question	Answer
Is sound used to signal an error?	No
Are prompts stated constructively, without overt or implied criticism of the user?	Not considered

Table 27*. Heuristics - Help users recognize, diagnoze, and recover from errors*

4. 2. 9 Help and documentation

It may be necessary to provide documentation to help users understand how to complete their tasks.

Question	Answer
Are data entry screens and dialog boxes supported by navigation and completion instructions?	No

Table 28*. Heuristics - Help and documentation*

4. 3 Summary

High-level prototyping makes for a true native app experience thanks to its user interaction level and degree of detail. Source code for the presented above prototype was generated and exported, usability testing involving multiple users was perfomed. Results, reflecting the prevailed opinions were gathered. The method used for usability testing has showed positive outcome in terms of the UI processing and, on the other side, detected problems within such aspects as error handling and user awareness and guidance.

5. Implementation

I developed the application in Expo and React Native with Typescript codebase. It is essentially multi-platform and can run on Android 5 / iOS 10 and higher. As stated earlier, the core development principles applied were Separation of Concerns (using the BLoC pattern) and the Material Design. Although it was primarily made for tablets, in the end it is fully responsive and can be used on any devices.

The code is written in the newest officially supported ECMAScript 2020 specification with JSX usage. The core libraries include RxJS and React Native Paper.

5. 1 Development Tools

- **Visual Studio Code** as a code editor for building and debugging
- **Expo** for cross-platform deployment using TypeScript codebase
- **React Native** for native UI management
- **Swagger** as a RESTful API documentation tool
- **Android Studio** as an Android emulator
- **ESLint** as a static code analysis tool

5. 2 Libraries

- **RxJS** - Reactive Extensions Library for JavaScript [15]
- **React Native Paper** - a collection of customizable and production-ready components for React Native, following Google's Material Design guidelines [16]

5. 3 Architecture

React Native provides its own UI abstraction layer over iOS and Android platforms. React Native core and native components invoke the native views so that it is possible to write the application UI with JavaScript or TypeScript, instead of Kotlin/Java or Swift/Objective-C [17].

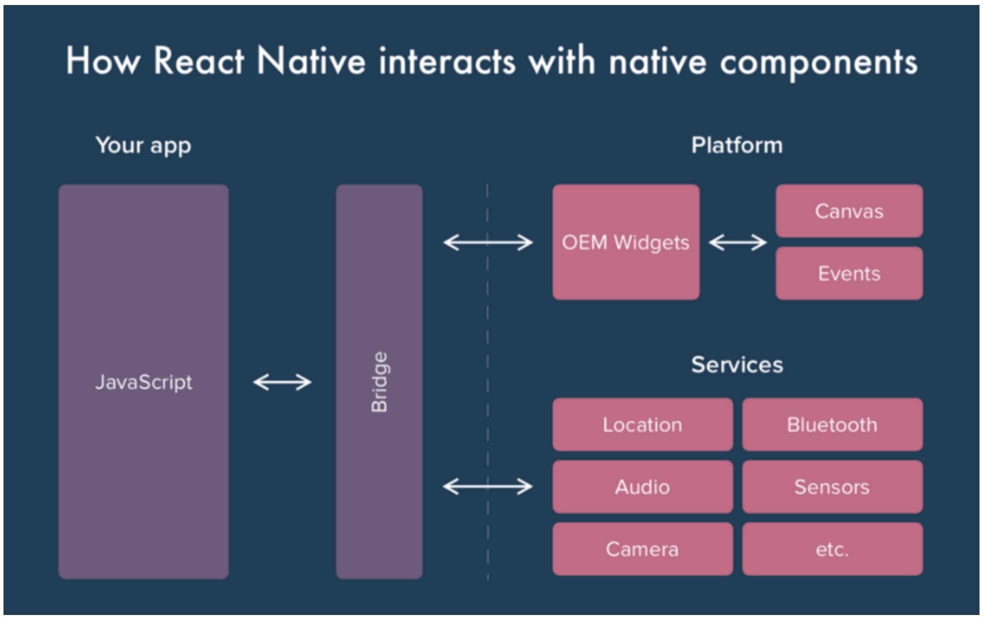


***Figure 20**. Implementation - React Native core concept*

5. 3. 1 Function Components and Hooks

In React, there are two ways to define a component: using a class (as a ES6 standars) or a function. Because of their reputation of representing a cleaner a simpler code, the Function components have become a new standard, so I fully relied on them while creating my application.

Hooks allow us to reuse stateful logic without changing the component hierarchy [18]. The hooks I use in my application are useEffect, which is a combination of componentDidMount, componentDidUpdate, and componentWillUnmount, and useState, which makes it possible to add React state to function components.

In Code Listing 1, I define a DisplaySetsScreen function component. I use the state hook for initializing a *sets* object, which is either of Record array type whose property keys are strings and whose property values are also strings, or of null type. The hook does all the state management, so I do not need to write custom setters. The *useEffect* hook, by its nature, gets called after each re-render. The second argument, called a dependency array, provides a condition defining the next run. Since I do not need any condition and need the hook to be run only after the first render, I provide an empty array as the second argument. After each first render, the sets are being fetched from the server and caught in a screen component with an Observable. In case of an error, an animated activity indicator is shown.

```
const DisplaySetsScreen = ({navigation}: Props) => {

  // Display Sets fetched list
  const [sets, setSets] = useState<Record<string, string>[] | null>(null)
;
  useEffect(() => {
    const setsObservable = ItemsBloc.getSets()?.subscribe(setSets, () =>
(
      <Indicator />
    ));
    return function cleanup() {
      setsObservable.unsubscribe();
    };
  }, []);
```

***Code Listing 1**. DisplaySetsScreen.tsx. Function components and hooks*

5. 3. 2 BLoC Pattern

BLoC (Business Logic Component) is a design pattern announced by Paolo Soares in the Google Dart Conference 2018 and originally associated with Flutter as an alternative way of managing states. The initial proposal was to reuse the code related to the business logic in different platforms, in this case, Angular Dart and Flutter. Later, a similar concept was adopted in the React and JavaScript environment.

According to official documentation [19], BLoC attempts to make state changes predictable by regulating when a state change can occur and enforcing a single way to change state throughout an entire application.

In Code Listing 2, I declare a BehaviorSubject, a subtype of an RxJS Subject which stores the latest value emitted to its consumers. Whenever a new Observer (a value consumer) subscribes, it will immediately receive the latest (or the current) value from the BehaviorSubject. Such way, tokens are being stored using the AsyncStorage. Then, I export an object containing a set of declared functions later to be used by the LoginScreen.tsx component. Login function processes a POST response from the API layer, which returns a Bearer token, and stores it in the BehaviorSubject stream. Tokens are later used for sending any other requests to the server within a session.

```
import AsyncStorage from "@react-native-async-storage/async-storage";
import {BehaviorSubject} from "rxjs";
import {API} from "../core/api";
import {Bloc} from "../blocs/Bloc";
import {ControllerBloc} from "./ControllerBloc";

const currentUser: BehaviorSubject<unknown> = new BehaviorSubject<unknown
>(
  AsyncStorage.getItem("currentUser"));

export const AuthBloc = {
  login,
  logout,
  getToken
};

async function login(username: string, password: string) {
  await API.login(username, password).then(user => {
    AsyncStorage.setItem("currentUser", user);
    currentUser.next(user);
    getToken().subscribe(res => {
      if (typeof res === 'string') Bloc.setAuthToken(res);
    });
  });
}

function logout() {
  AsyncStorage.removeItem("currentUser");
  currentUser.next(null);
  ControllerBloc.close();
}

function getToken() {
  return currentUser;
}
```

Code Listing 2*. AuthBloc.tsx. BLoC*

BLoC was designed with three core values in mind of the authors [19]: simple, powerful and testable.

Throughout the application, five BLoCs are used:

- Bloc.tsx as a root class component containing crucial information
- AuthBloc.tsx for authentication & token management using AsyncStorage as a local repository
- ItemsBloc.tsx for handling collections
- ControllerBloc.tsx for controlling playback
- ThemeBloc.tsx for theme switching (light/dark)

5. 3. 3 Project Structure

The whole project is structured as a classical Expo application and consists of four main modules: the source itself, configuration files, tests, and the assets directory for media files. The source directory includes */blocs, /components, /core, /screens, /styles, /utils* directories, as well as the *index.ts* as an entry point for navigation.

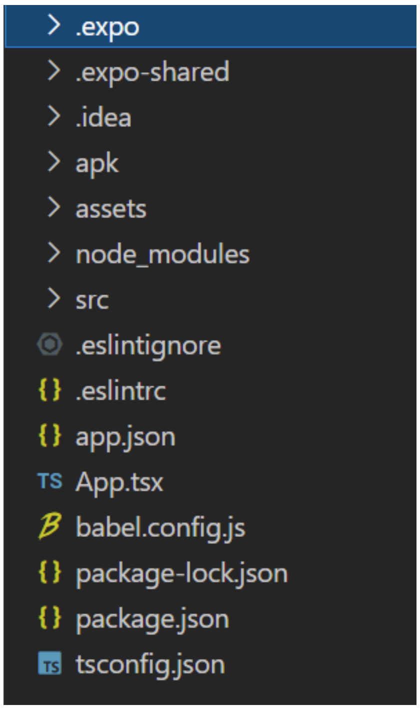

```
> .expo
> .expo-shared
> .idea
> apk
> assets
> node_modules
> src
.eslintignore
{} .eslintrc
{} app.json
TS App.tsx
babel.config.js
{} package-lock.json
{} package.json
tsconfig.json
```

***Figure 21**. Implementation - Project structure*

5. 4 Consuming REST API

Nowadays, there are two mainstream approaches of consuming REST APIs in React Native: the first is Axios, which is a promise-based HTTP client and Fetch API, which is a browser in-built XMLHttpRequest alternative providing a more powerful and flexible feature set [20]. Both share a similar functionality, with a difference that the Fetch does not require including any additional libraries and can be called directly. It supports such concepts as Cross-Origin Resource Sharing (CORS) and the HTTP Origin header semantics.

The *fetch* method takes one mandatory argument, the path to the resource you want to fetch. It returns a Promise that resolves to the Response to that request — as soon as the server responds with headers — even if the server response is an HTTP error status [20].

The wrapper from Code Listing 3 shows its usage.

```
async function get(url: string) {
  const token = Bloc.getAuthToken();
  const requestOptions = {
    method: "GET",
    headers: {Authorization: token}
  };
  return await fetch(url, requestOptions);
}
```

***Code Listing 3**. api.wrapper.tsx. Fetch API*

5. 5 Managing playback

5. 5. 1 Running autonomously

After the sets are fetched from the server, it is possible to run a chosen set autonomously in a loop. This is done by calling the */runautonomous* function, which starts playback of a set marked as "autonomous" on the server.

5. 5. 2 Playing chosen Tracks

When a set is chosen and tracks are fetched, the set is being prepared for playback by two steps:

- triggering */addplaylist* function
- updating the ControllerBloc for setting the current set.

The playback starts by triggering the Play button placed in the bottom bar. While it is possible to play next and previous tracks, unfortunately, the shuffle and loop functionality could not be implemented properly since it is not currently provided by the server.

5. 5. 3 Playing a Chapter

Chapter adding is implemented using *react-native-simple-time-picker* package. A track is being played starting from the given second by calling the */seek* function shown in Code Listing 4 and taking the number of seconds as a parameter.

```
const handleSeek = (track: Record<string, string>, second: number) => {
    ControllerBloc.seek(second)
      .then(() => {
        ControllerBloc.setCurrentTrack(track);
        const current = ControllerBloc.getCurrentTrack();
        if (current) handlePlay(current);
      })
      .catch(() => {
        showErrorMessage("Chapter cannot be played.");
      });
  };
```

***Code Listing 4**. DisplayTracksScreen.tsx. Seek*

5. 6 Requirements Fulfillment Issues

During implementation, the development server was being redesigned and the API routes and endpoints were changed significantly. The original VLC was changed to MPlayer, some of the functionality, originally available for the clients and API consumers were temporarily disabled. Such functionality include Shuffle and Loop functions. The video files and track previews were implemented quite late and, therefore, unfortunately, could not be included in the current mobile application version and were postponed until future upgrades.

5. 7 User Interface Improvements

Since we follow agile iterative approach, changes may occur during any of the applied stages. Thus, some changes in the user interface were approved and implemented. Feedback and error messages for both successful and erroneous operations were added. The side menu on the left was eliminated because of its redundancy and its functionality was replaced to a more compact and lightweight triple dot menu at the appbar. These changes resulted in improved heuristics and better usability.

6. Evaluation and Testing

6. 1 Test Strategy

The goal of this section is to verify that all the requirements are fulfilled according to business and technical specification. The specification itself is defined in the Analysis section in the form of requirements.

6. 1. 1 Test Design Techniques

There are two standardized testing design techniques based on different scientific models and widely accepted by many Quality Assurance professionals [21]:

- Static testing, which involves testing without program execution and can be done either manually or using tools
- Dynamic testing, which is being performed by running the whole system and after code deployment.

Some of the techniques I used are listed in Table 29:

Type	Subtype
Static	Informal review
	Static code analysis
	Compliance to coding standard
Dynamic	Use Case Testing
	System Testing

***Table 29**. Testing design techniques*

6. 2 Use Case Testing Report

For each use case, I defined a test case (scenario) derived from it (Tables 30-51 provided with screenshots as Figures 22-35) . A test case has a specific structure and can be defined as “a specification of the inputs, execution conditions, testing procedure, and expected results that define a single test to be executed to achieve a particular software testing objective, such as to exercise a particular program path or to verify compliance with a specific requirement [22].”

The tests were executed on four different presentations (see Input data).

Test Case (ID: Name)	1: Login - Basic flow
Use Case ID	MOB-1, MOB-12
Input data	Username: admin Password: nimda
Expected result	Employee is authenticated and redirected to the Display Sets page. Display Sets are readable either as a list or grid. Server HTTP response status is 200. Run Autonomous button is disabled, no checkboxes are checked.
Real result	PASS

***Table 30**. Test Cases - 1: Login - Basic flow*

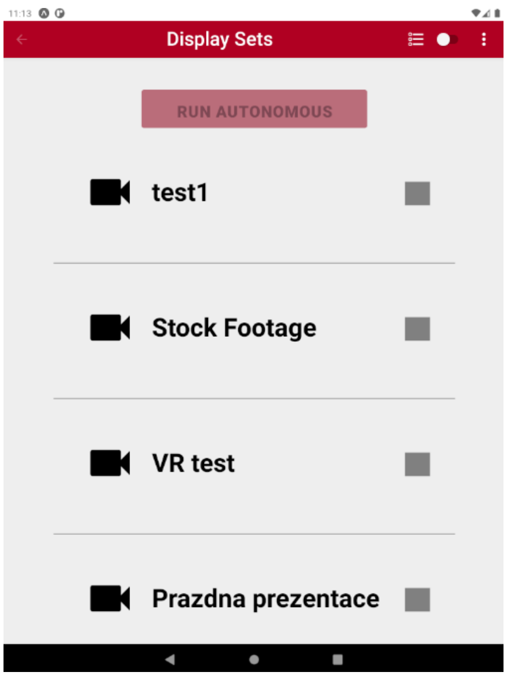


***Figure 22**. Evaluation - Sets as list*

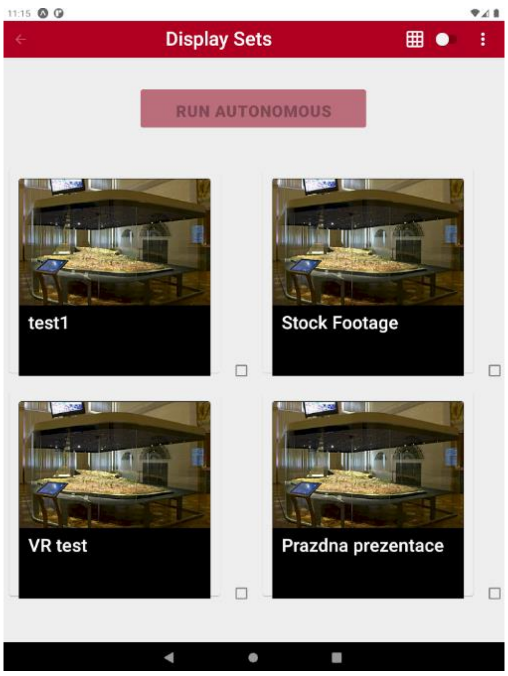


***Figure 23**. Evaluation - Sets as grid*

Test Case (ID: Name)	2: Login - Alternate flow - Wrong input format
Use Case ID	MOB-1
Input data	Username: admin Password:
Expected result	A "Fields cannot be empty" notification is shown.
Status	PASS

***Table 31**. Test Cases - 2: Login - Alternate flow - Wrong input format*

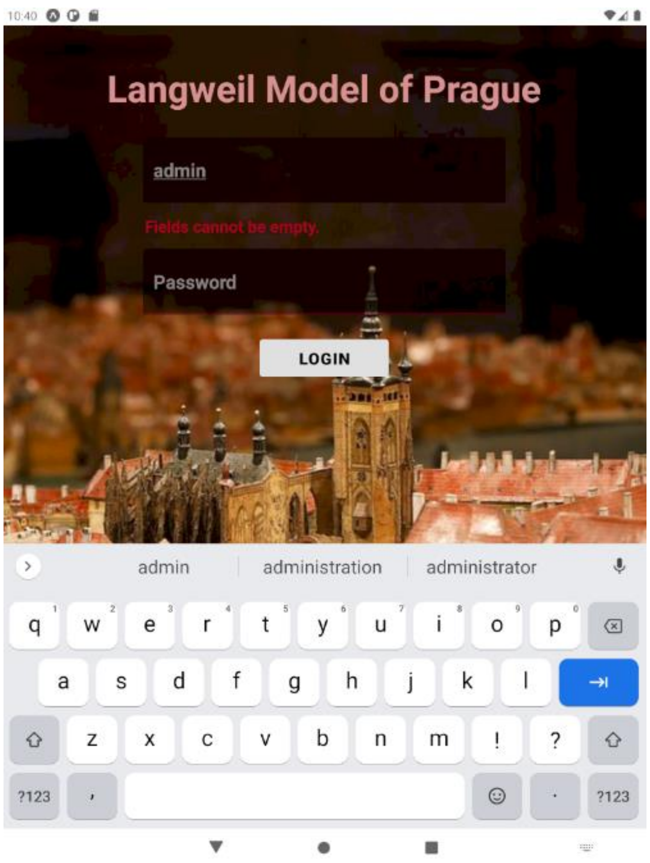


***Figure 24**. Evaluation - Login, empty fields*

Test Case (ID: Name)	3: Login - Alternate flow - Wrong user data
Use Case ID	MOB-1
Input data	Username: admin Password: nimda3
Expected result	A "Wrong userdata" notification is shown.
Status	PASS

***Table 32**. Test Cases - 3: Login - Alternate flow - Wrong user data*

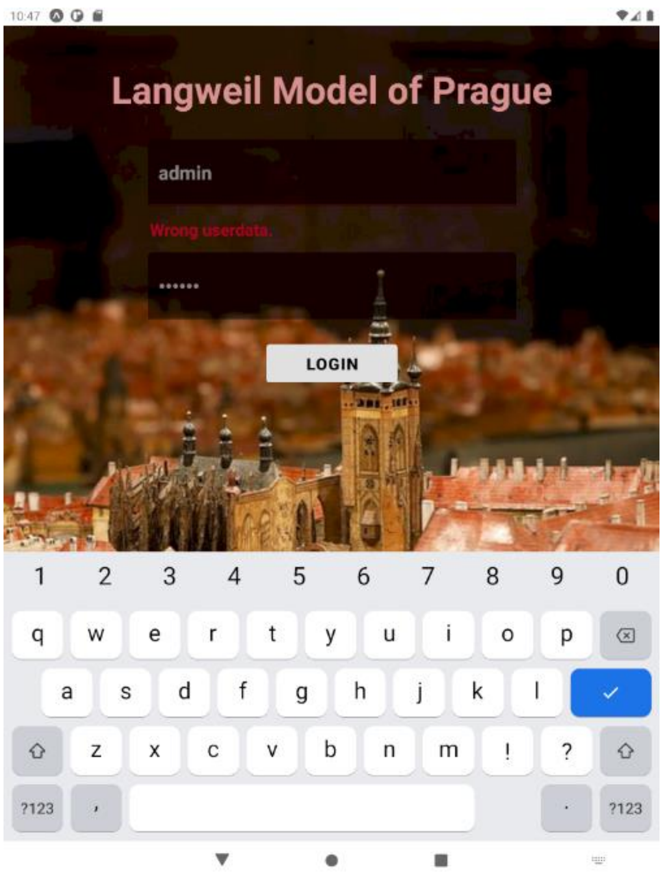


Figure 25. Evaluation - Login, wrong user data

Test Case (ID: Name)	4: Login - Exception flow - No sets to show
Use Case ID	MOB-1
Input data	-
Expected result	An animated activity indicator is shown.
Status	PASS

Table 33. Test Cases - 4: Login - Exception flow - No sets to show

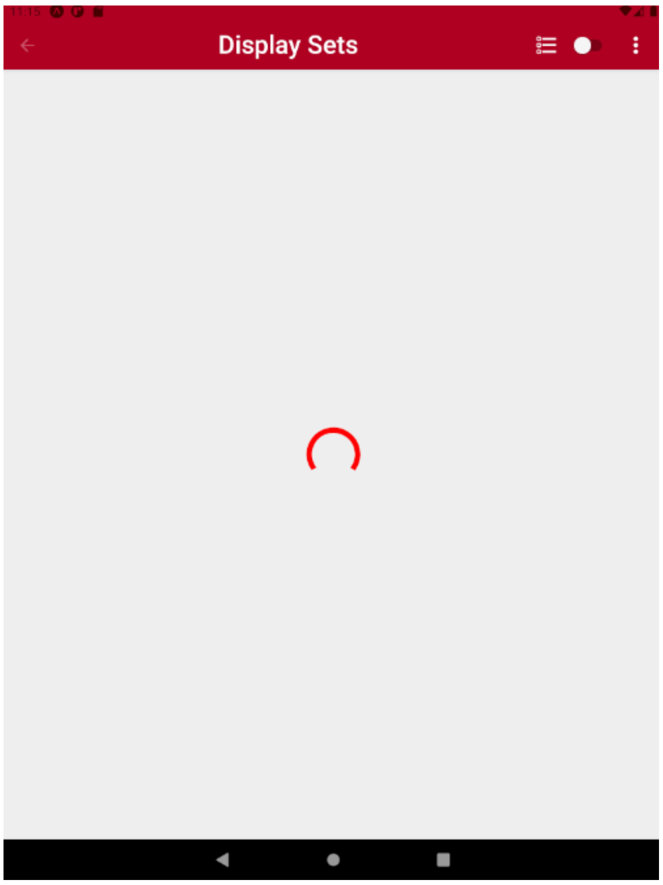


***Figure 26**. Evaluation - Waiting indicator*

Test Case (ID: Name)	5: Display Tracks - Basic flow
Use Case ID	MOB-2
Input data	Presentation 1, 2, 3
Expected result	Display Tracks are readable as a list, server HTTP response status is 200.
Status	PASS

***Table 34**. Test Cases - 5: Display Tracks - Basic flow*

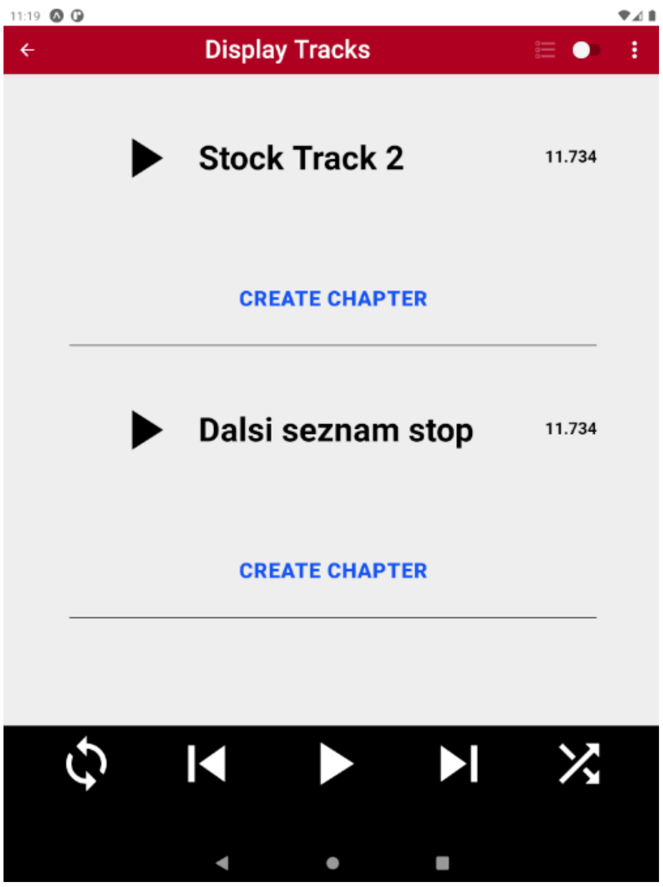


Figure 27. Evaluation - Tracks

Test Case (ID: Name)	6: Display Tracks - Alternate flow - No tracks to show
Use Case ID	MOB-2
Input data	Presentation 4
Expected result	An animated activity indicator is shown.
Status	PASS

Table 35. Test Cases - 6: Display Tracks - Alternate flow - No tracks to show

Test Case (ID: Name)	7: Play Track - Basic flow
Use Case ID	MOB-3
Input data	Presentation 1, 2, 3
Expected result	Display Track is being played, server HTTP response status is 200.
Status	PASS

Table 36. Test Cases - 7: Play Track - Basic flow

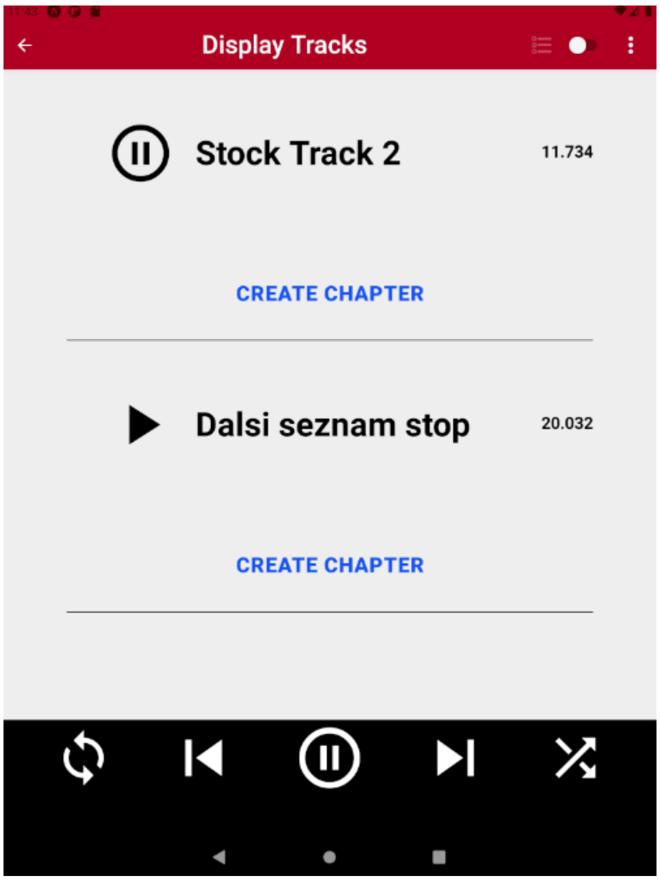


__Figure 28__. Evaluation - Playing Track

Test Case (ID: Name)	8: Pause Track - Basic flow
Use Case ID	MOB-4
Input data	Presentation 1, 2, 3
Expected result	Display Track is paused, server HTTP response status is 200.
Status	PASS

__Table 37__. Test Cases - 8: Pause Track - Basic flow

Test Case (ID: Name)	9: Play Next Track - Basic flow
Use Case ID	MOB-5
Input data	Presentation 1, 2, 3
Expected result	Next Track is being played, server HTTP response status is 200.
Status	PASS

__Table 38__. Test Cases - 9: Play Next Track - Basic flow

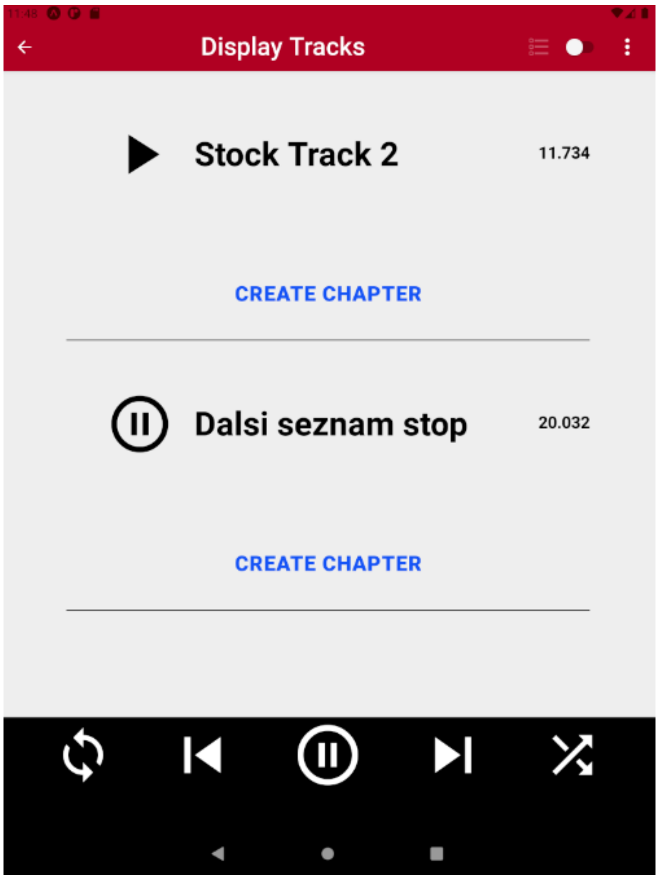


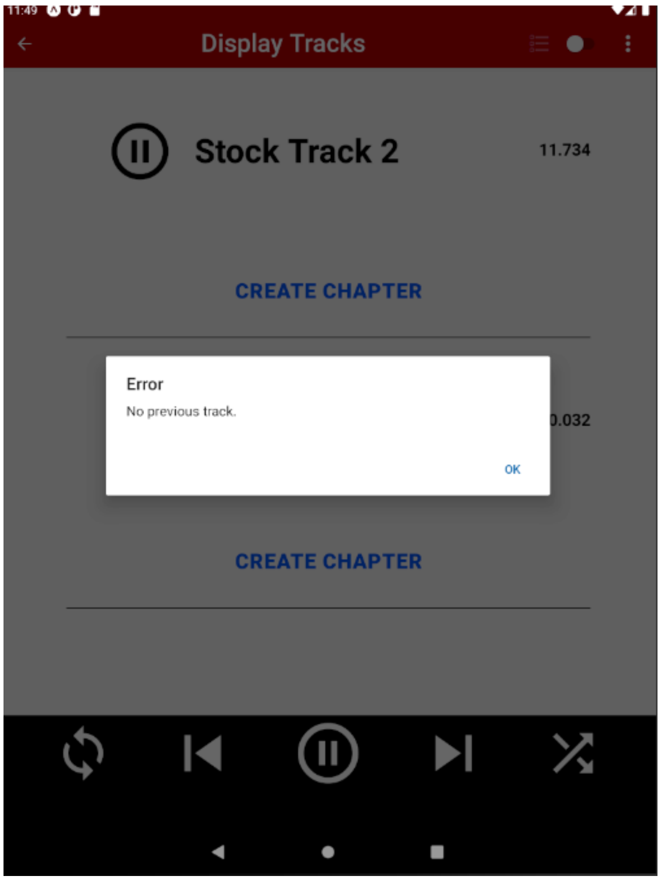
Figure 29. *Evaluation - Playing Next Track*

Test Case (ID: Name)	10: Play Previous Track - Basic flow
Use Case ID	MOB-6
Input data	Presentation 1, 2, 3
Expected result	Previous Track is being played, server HTTP response status is 200.
Status	PASS

Table 39. *Test Cases - 10: Play Previous Track - Basic flow*

Test Case (ID: Name)	11: Play Previous Track - Exception flow - Playing from the first track
Use Case ID	MOB-6
Input data	Presentation 1, 2, 3
Expected result	An error alert is shown.
Status	PASS

Table 40. *Test Cases - 11: Play Previous Track - Exception flow - Playing from the first track*



Error

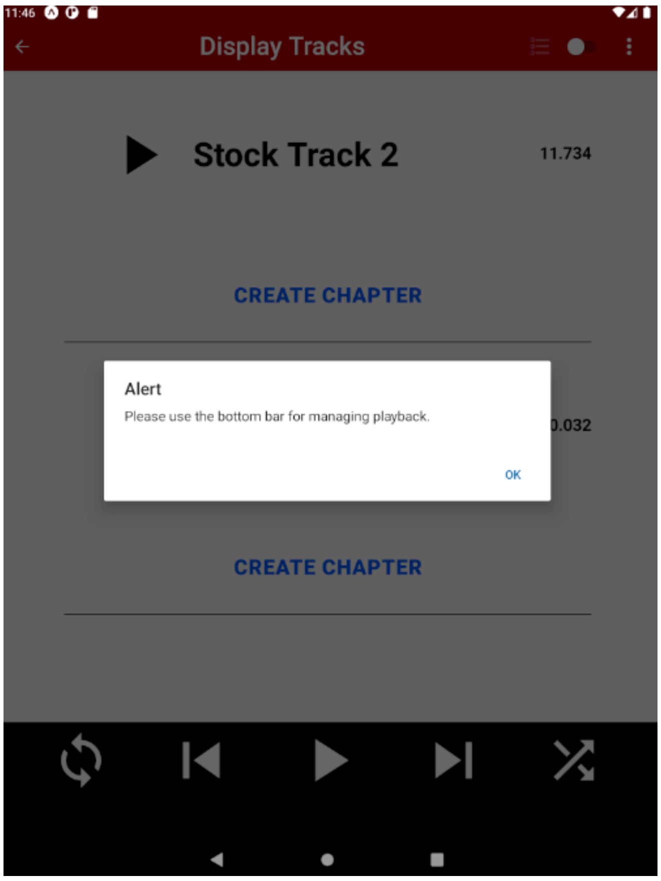
No previous track.

OK

Figure 30. Evaluation - Error

Test Case (ID: Name)	12: Play Track - Exception flow - Playing Track outside of the bottom bar
Use Case ID	MOB-6
Input data	Presentation 1, 2, 3
Expected result	An error alert is shown.
Status	PASS

Table 41. Test Cases - 12: Play Track - Exception flow - Playing Track outside of the bottom bar



Alert

Please use the bottom bar for managing playback.

OK

***Figure 31**. Evaluation - Alert*

Test Case (ID: Name)	12: Shuffle Track - any flow
Use Case ID	MOB-7
Input data	-
Expected result	-
Status	NOT EXECUTED (See subsection 5.6)

***Table 42**. Test Cases - 12: Shuffle Track - any flow*

Test Case (ID: Name)	13: Loop Track - any flow
Use Case ID	MOB-8
Input data	-
Expected result	-
Status	NOT EXECUTED (See subsection 5.6)

***Table 43**. Test Cases - 13: Loop Track - any flow*

Test Case (ID: Name)	14: Video Files list - any flow
Use Case ID	MOB-9
Input data	-
Expected result	-
Status	NOT EXECUTED (See subsection 5.6)

***Table 44**. Test Cases - 14: Video Files list - any flow*

Test Case (ID: Name)	15: Create video chapter - Basic flow
Use Case ID	MOB-10
Input data	Presentation 1, 2, 3 Chapter name: Prague 1400 Start time: 0:15 End time: 2:04
Expected result	Video chapter is created, the button is appeared, the time is calculated correctly.
Status	PASS

Table 45. Test Cases - 15: Create video chapter - Basic flow

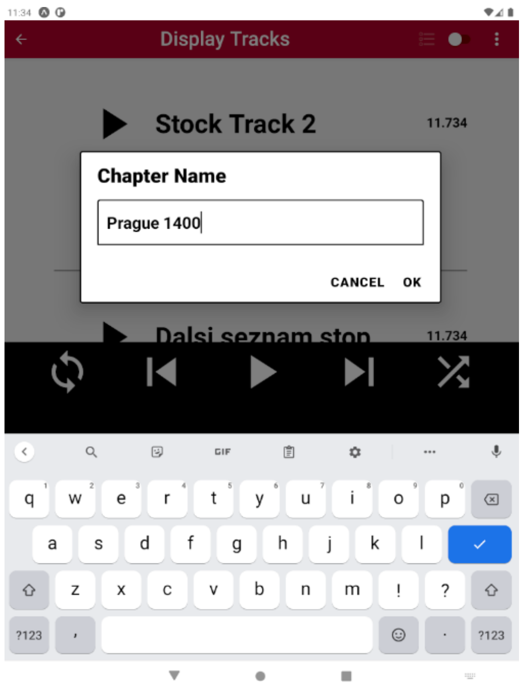


Figure 32. Evaluation - Chapter name

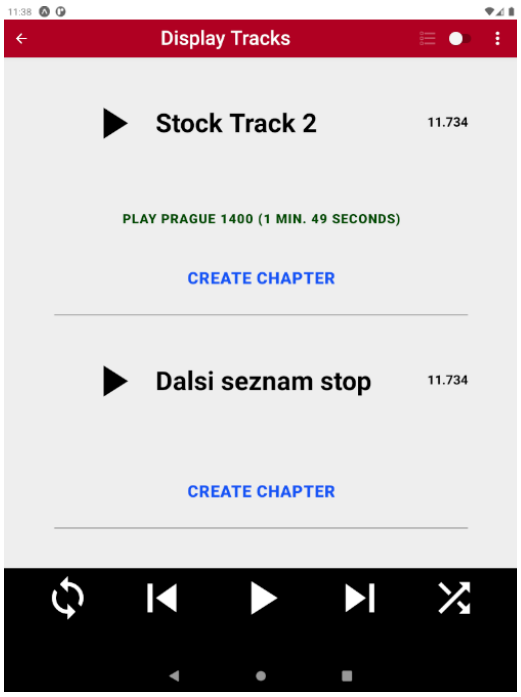


Figure 33. Evaluation - Chapter button

Test Case (ID: Name)	16: Create video chapter - Exception flow - End time before start time
Use Case ID	MOB-10
Input data	Presentation 1, 2, 3 Chapter name: Prague 1400 Start time: 0:10 End time: 0:02
Expected result	Video chapter is not created, an error alert is shown.
Status	PASS

Table 46. Test Cases - 16: Create video chapter - Exception flow - End time before start time

Test Case (ID: Name)	17: Play chapter - Basic flow
Use Case ID	MOB-11
Input data	Presentation 1, 2, 3
Expected result	The Track is being played from the stated time, server HTTP response status is 200.
Status	PASS

__Table 47__. Test Cases - 17: Play chapter - Basic flow

Test Case (ID: Name)	18: Dark mode - Basic flow
Use Case ID	MOB-13
Input data	Presentation 1
Expected result	Dark mode is switched quickly with no delay.
Status	PASS

__Table 48__. Test Cases - 18: Dark mode - Basic flow

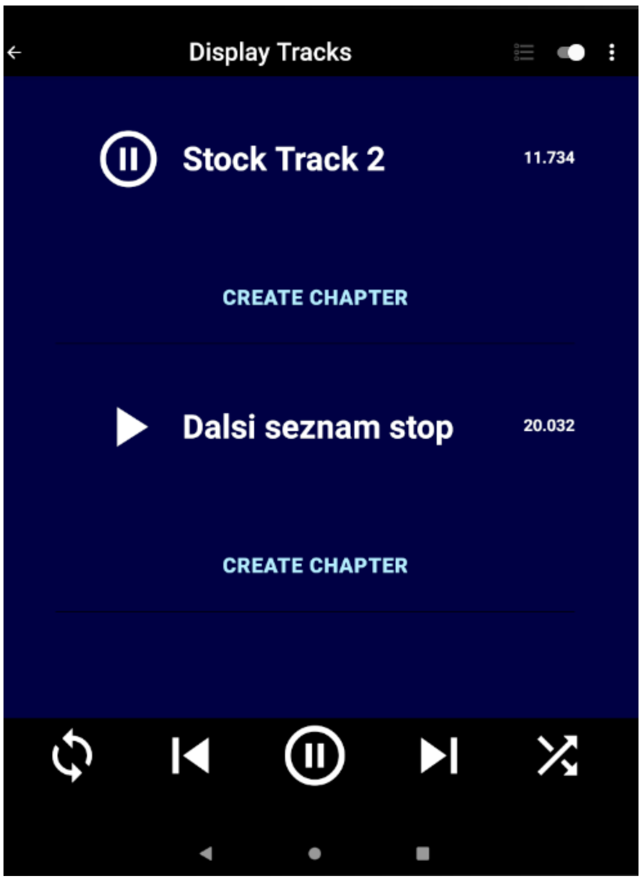


__Figure 34__. Evaluation - Dark mode

Test Case (ID: Name)	19: Run Autonomous - Basic flow
Use Case ID	MOB-14
Input data	Presentation 1, 2, 3
Expected result	A Set is checked, Run Autonomous button is enabled, the button text changed to "Stop", server HTTP response status is 200.
Status	PASS

Table 49. Test Cases - 19: Run Autonomous - Basic flow

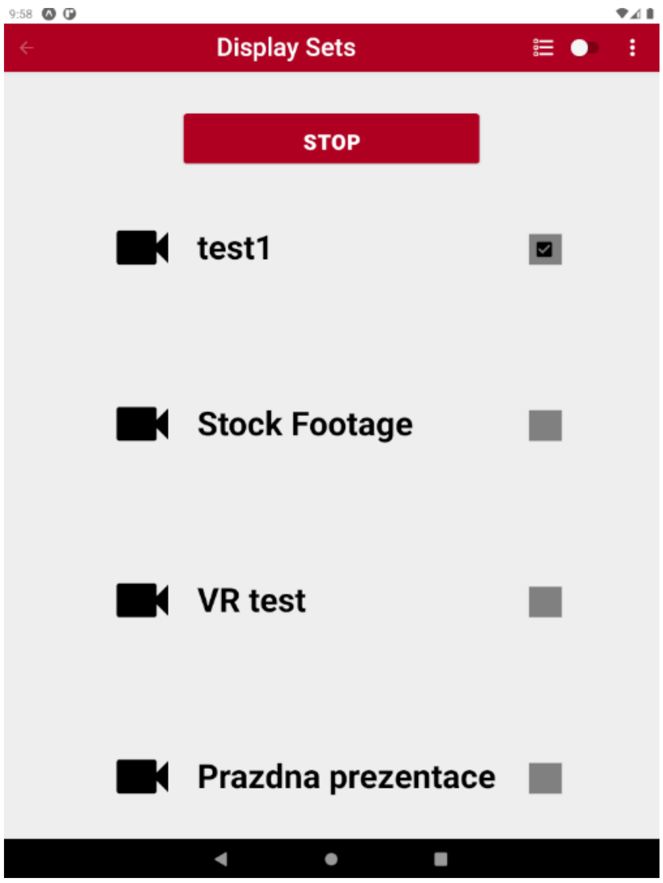


Figure 35. Evaluation - Running autonomous

Test Case (ID: Name)	20: Run Autonomous - Alternate flow - Choosing another set
Use Case ID	MOB-14
Input data	Presentation 1, 2, 3
Expected result	Another Set is checked, all the others are unchecked.
Status	PASS

Table 50. Test Cases - 20: Run Autonomous - Alternate flow - Choosing another set

Test Case (ID: Name)	21: Logout - Basic flow
Use Case ID	MOB-15
Input data	Presentation 1, 2, 3
Expected result	Employee is logged out and redirected to the Login page. Data are prefilled.
Status	PASS

***Table 51**. Test Cases - 21: Logout - Basic flow*

6. 3 Usability Testing Report

A usability test is intended to determine the extent an interface facilitates a user's ability to complete routine tasks. Users are asked to complete a series of routine tasks. Sessions are recorded and analyzed to identify potential areas for improvement to the product [23]. I conducted an onsite usability test using the current version of my application with participation of three testers. All of them are of different age categories, one of them was familiar with software testing. In every case, an Android device of a version 5 or higher was used. The purpose of the test was to assess the usability of the user interface design, information flow, and information architecture.

Three questions were put in front of the participants:

1. What parts of the mobile app did you like the most? Why?
2. What do you think about the way features and information were presented?
3. If you had a choice, why would you keep using this mobile app? Why would you not [24]?

Table 52 contains the participant feedback, satisfactions ratings and recommendations for improvements. A 5-point scale was used for the rating with 1 as the worst up to 5 as the best.

Participant ID	Question ID	Feedback	Recommendations	Overall satisfaction rating
1	1	Color choices and theming	Add more functionality, make it more interactive	4
	2	The grid view is a bit messy		
	3	I would, because I enjoy its good design		
2	1	The fact that it is scaled for a tablet	No recommendations	5
	2	Everything is good		
	3	I would, because I like its big resolution and it is pleasant to handle		
3	1	Color choices and theming	Add more functionality,	

	2	More information about tracks could be presented	make it more interactive	3
	3	I would not, because the functionality is limited and it needs many upgrades		

***Table 52**. Usability Testing - results*

6. 4 Summary

Several types of testing design techniques were applied in this work. Results of both showed overall success and proved the application to be useful and as functional as it could be at the moment of writing the Thesis. There were no significant drawbacks detected in terms of user interface design, performance, or functionality. While Use Case Testing showed no failure, usability methods helped to gain more insight into user experience issues and gather qualitative data for future consideration and deeper analysis.

7. Conclusion

A substantial upgrade has been made to the Multiprojection control system project being currently developed at the Department of Computer Science. A completely new version of a mobile application aimed at remote presentations management, which is supposed to be performed by the City of Prague Museum employees, has been implemented and tested. I consider the results of this work to be successful and worthwhile, although not all of its parts was completed according to the initial plan.

The Thesis goals were achieved exactly as they were stated and all the problems with the existing mobile application version mentioned in the Motivation section were eliminated.

In this chapter, I will go through the Thesis goals' realization process.

Goal 1. Design an upgraded version of a mobile application scaled for tablets, eliminate all the problems mentioned above and represent it with a high-fidelity clickable prototype.

The goal completion is described in detail in the Design chapter.

Goal 2. Implement the designed solution through creating a multi-platform application mainly targeted at Android devices.

The goal completion is described in detail in the Implementation chapter.

Goal 3. Test the application on four different presentations.

The goal completion is described in detail in the Evaluation and Testing chapter.

7. 1 Future Work

For future development and upgrades, recommendations made in the Usability Testing results table should be taken into account. Temporarily missing functionality should be completed and new features may be implemented. The Android package size should be reduced by removing unnecessary dependencies and cleaning out native code generated by Expo framework.

Bibliography

[1] Kling, Rob, *The Organizational Context of User-Centered Software Designs*, MIS Quarterly, 1977

[2] Norman, D. A., *User-Centered System Design: New Perspectives on Human-Computer Interaction*, L. Erlbaum Associates Inc., 1986

[3] *Usability.gov*. https://www.usability.gov/what-and-why/user-centered-design.html, accessed on 10.05.2021

[4] *Interaction-design.org*. https://www.interaction-design.org/literature/topics/user-centered-design, accessed on 10.05.2021

[5] *IBM*. https://developer.ibm.com/depmodels/reactive-systems/articles/defining-the-term-reactive/, accessed on 10.05.2021

[6] *ArhitecturalDigest*. https://www.architecturaldigest.com/story/digital-art-and-design-exhibitions-get-lost-in-from-home accessed on 05.01.2021

[7] Bc. Ondřej Trojan, *Multiprojection Control Systems*, Czech Technical University in Prague, 2020

[8] Alain Abran, James W. Moore, Pierre Bourque, Robert Dupuis, *Guide to the software engineering body of knowledge*, Computer Society Press, 2004

[9] *Systems Engineering Fundamentals*, Defense Acquisition University Press, 2001

[10] Michael Jesse Chonoles, *OCUP Certification Guide*, Morgan Kaufmann, 2018

[11] *Google*. material.io, accessed on 25.11.2020

[12] *Nielsen Norman Group*. https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation, accessed on 05.01.2021

[13] *Nielsen Norman Group*. https://www.nngroup.com/articles/ten-usability-heuristics, seen on 05.01.2021

[14] Elaine Weiss, *Making Computers People-Literate*, Jossey-Bass Publishers, 1993

[15] *RxJS*. *https://rxjs-dev.firebaseapp.com*, accessed on 02.04.2021

[16] *React Native Paper*. https://callstack.github.io/react-native-paper, accessed on 08.04.2021

[17] *Dev.to*. https://dev.to/goodpic/understanding-react-native-architecture-22hh, accessed on 03.04.2021

[18] *ReactJS.org*. https://reactjs.org/docs/hooks-intro.html, accessed on 10.03.2021

[19] *BlocLibrary.dev*. https://bloclibrary.dev/#/whybloc, accessed on 14.03.2021

[20] *Mozilla.org*. https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API, accessed on 07.03.2021

[21] *ArtOfTesting.com*. https://artoftesting.com/test-design-techniques#Static_Test_Design_Techniques, accessed on 12.05.2021

[22] *24765-2010 - ISO/IEC/IEEE International Standard - Systems and software engineering -- Vocabulary*, IEEE, 2010

[23] *Usability.gov*. Usability Test Report Template, accessed on 13.05.2021

[24] *PlaybookUX.com*. https://www.playbookux.com/usability-testing-questions, accessed on 13.05.2021

Appendices

A. Installation

Applicable to Windows 10 environment.

A. 1 Android Studio

Build an APK file after Expo environment installation as described in subsection A.2 with `expo build:android -t apk` command.
Open Android Studio and choose "Configure" --> AVD Manager. Create a virtual device and run the emulator. Drag the APK file onto the emulator screen. An APK Installer dialog will appear. When the installation completes, the application is available in the apps list.

A. 2 Expo client

Node.js v. 14 or higher is required. No APK needed. Go the project's root directory and run:

1. `npm install --global expo-cli` to configure Expo command line tools
2. `npm install` to configure all the necessary packages and dependencies
3. `expo-start` to start Metro Bundler (a HTTP server compiling the TypeScript code)

Go to the specified server address and use the QR code to open the application on an Android tablet device (Expo client needed) or choose the "Run on Android device/emulator" option.