Culturebook, culture preservation app

George Sigalas

933287

A dissertation submitted in partial fulfilment
of the requirements for the degree of
Bachelor of Science
of the
University of Portsmouth.

School of Computing
Engineering Project

April 21, 2023

Declaration

No portion of the work contained in this document has been submitted in support of an application for a degree or qualification of this or any other university or other institution of learning. All verbatim extracts have been distinguished by quotation marks and all sources of information have been specifically acknowledged.

Date: April 21, 2023

Consent to Share

I consent for this project to be archived by the University Library and potentially used as an example project for future students.

Date: April 21, 2023

Acknowledgements

I would like to extend my sincere thanks to Sarah Guerin for her continued support and guidance with my requirements gathering. Additionally, I could not have undertaken this journey without my supervisor Dr Janka Chlebikova with her constant support and guidance throughout the project. Finally, and certainly not least, I would like to extend my thanks to "The Icon Z" for creating the awesome vector used in the app's logo (Z, n.d.).

Abstract

Intangible Cultural Heritage (ICH), the part of humanity's cultural heritage that includes all non-physical elements, is at a constant risk of disappearance. As will be discussed in this paper, the intangible part of a culture is often complementing the tangible part. However, due to its ethereal and individual-focused nature, it is difficult to document and preserve. Previous research indicates the rate at which cultural elements are transmitted to younger generations is slow compared to the aggressive urban expansion and globalisation. As such, there is a clear need for a cultural preservation solution that is user-friendly and engaging to keep up with the ever-changing societal growth. This project aims to build a digital framework for users to generate content based on their cultural experiences, by using a scrum-like approach to develop an Android app. The requirements of which, have been collected through interviews, research and surveys.

Keywords - *ICH; cultural heritage; intangible cultural heritage; preservation; tourism; climate change; politics*

Contents

List of Figures

List of Listings

Chapter 1

Introduction

Throughout history, humans have sought to preserve cultural elements. However, this effort is increasingly challenging in our era due to complex sociological and political phenomena. This paper aims to remedy the increasing endangerment of human traditions, customs and ethereal cultural elements with a socially engaging application. The following literature review defines, what Intangible Cultural Heritage is, and why its preservation is crucial. Additionally, it discusses the most significant factors contributing to its endangerment and, finally, reflects on how a mobile application could remedy the situation.

Chapter 2

Literature Review

2.1 Intangible Cultural Heritage

Humans are considered social beings, but this attribute extends to human culture also. Intangible Cultural Heritage (henceforth known as ICH) consists of all the knowledge, skills, practices, expressions, and representations that a human community considers part of its cultural heritage (UNESCO, 2003). ICH is also inherited, but it is also constantly evolving as the needs of the community change. The social contributions of ICH are undisputed since it provides a sense of identity to the group and the individual. Furthermore, it disseminates creativity and fosters respect for cultural diversity. Finally, ICH is aware of human rights whilst promoting mutual respect among communities (Ballard, 2008). Thus, its significance to the community is immense, and its preservation should be considered critical.

However, until the end of the 20th century, ICH was not officially considered part of the cultural preservation efforts. International treaties focused on preserving tangible cultural heritage, that is, the material cultural artefacts and monuments, in case of armed conflicts (UNESCO, 1954) or peace (UNESCO, 1972). Lenzerini, 2011 believes the cause was because it was common knowledge that ICH is inherited, causing global preservation efforts to deem it unimportant. Arguably, the rapid societal and technological growth and external factors, such as climate change and politics, pressured international institutions to act (Lenzerini, 2011). Thus, UNESCO officially identified and adopted ICH in cultural preservation efforts in 2003 (UNESCO, 2003). Alas, ICH is still endangered, as efforts to preserve culture continue (Marzuki, 2011).

2.2 The risk of ICH

> "*We have the UNESCO heritage status, but the problem is, it became too commercialized, and even we, local people, cannot recognize it. It lost its identity. [I-R3, (Tan et al., 2018)]*"

The above quote is an example of a problem in the current implementation of cultural preservation efforts in which, even though a place is labelled a World Heritage Site, rapid changes can endanger ICH and cause mental distress to residents (Tan et al., 2018). Factors such as tourism, as expressed in the above quote, climate change, and politics contribute to its deterioration. This chapter examines tourism, climate change, and politics since these are some of the most significant yet intriguing factors. Specifically, the available literature shows a correlation between tourism and ICH disappearance, but one cannot ignore its positive effect on the economy. Also, climate change is ever more relevant, with many coastal areas in danger of being submerged in the future. Finally, politics is engrossing since international institutions hold the preservation power, which contradicts what ICH stands for.

2.2.1 Tourism

Of all the factors impacting ICH, tourism is perhaps the most controversial, as tourism has a tremendously positive impact on the economy. Specifically, tourism is one of the most significant contributors to the global gross domestic product, with an approximately 24.1% contribution (WTTC, 2021). Furthermore, the quality of life of a place improves with tourism as described in Goeldner et al., 2000. For example, in the Langkawi Islands in Malaysia, Marzuki, 2011 studied the benefits and cost of tourism from a socio-economic perspective, and the residents of the islands feel like the growth of tourism has enhanced their quality of life, with an improvement to local infrastructure and economy (Marzuki, 2011). Notwithstanding this positive effect, there is a trend of negative socio-cultural impact, which as George, 2010 argues, is detrimental to ICH as the islands' cultural identity is silently fading away.

2.2.2 Climate change

On another note, intangible cultural heritage is deteriorating by rising global temperatures. Japan is a prime example of ICH damaged by climate change. Hanami, the traditional custom of cherry blossom viewing in Japan, had its dates shifted due to the warmer seasons growing longer (Brimblecombe & Hayashi, 2017). However, this is also an example of the intangible cultural heritage evolving as the environment around the community changes. Also, one could argue that this impact is minimal and should not constitute a concern. But, as natural events exacerbate and sea levels rise, it is only a matter of time before environmentally bound traditions get lost (Reimann et al., 2018).

2.2.3 Politics

Another factor contributing to the disappearance of ICH is politics, where, as discussed in Mountcastle, 2010, the misuse of international cultural preservation projects and programs could disparage minority cultures. Incidentally, higher-income countries fund most of the preservation efforts which target low- and middle-income countries. As such, these countries determine the scope and priorities of the preservation efforts (Simpson et al., 2022).

2.3 Importance of preserving ICH

As previously stated, ICH is significant in the cultural identity of communities, and in many cases, it provides context to the existing tangible cultural heritage. In addition, preserving ICH has been shown to help in creativity due to its capability to incorporate different areas of study and spread information that imbues individuals and enterprises (Cominelli & Greffe, 2012). Likewise, even though tourism threatens ICH, one cannot overlook its positive contribution to the economy and the quality of life improvements in the community. Finally, ICH positively affects the health of individuals (Reguant-Aleix et al., 2009).

2.3.1 Creativity

Creativity is a challenging term to define as it incorporates various interconnected layers. However, one of those layers is culture and society (Hennessey & Amabile, 2010) and the way that ICH enhances creativity is two-fold. One way described by Cominelli and Greffe, 2012 is the *technical contribution* in which ICH acts as a knowledge carrier, incorporating the experience and practices of previous generations and changing them to fit current needs. Such an example is the porcelain practice in France, where the traditional knowledge found application in other fields, such as medicine and porcelain prosthetics. (Cominelli & Greffe, 2012) The other way described by Cominelli and Greffe, 2012 is the *social contribution*, in which ICH evolves with continuous social interaction, thus examining existing solutions to offer more suitable alternatives.

2.3.2 Economy

The benefits of culture on the economy are undisputed. As previously discussed, tourism tends to alter the cultural dynamics of an area, though its benefits are significant. For example, the European Union estimates that 40% of all European tourists are cultural tourists (Commission, n.d.), with over 115 million people in 2018 travelling abroad (CBI, 2018). Additionally, cultural tourism is slowly changing into creative tourism, in which tourists actively participate in local traditions and customs (CBI, 2018). Specifically, Efentaki and Dimitropoulos, 2015 discusses the effectiveness of traditional dancing on tourism, resulting in the vast majority of tourists being willing to extend their stay should there be a folk dancing workshop. This notion, while also considering the large number of cultural tourists, suggests that there should be a need for an ICH application that tourists could use to engage with a community's ICH.

2.3.3 Social and Health

ICH is inherently a social interaction, and as such, it should not be surprising that there is a significant social value present. Specifically, holders of ICH are seeking social interaction, possibly to transmit their knowledge, as well as their desire for cultural and information transactions that will further enhance their wisdom (Cominelli & Greffe, 2012). In

addition, cultural diets such as the Mediterranean diet hold a spiritual role in celebrations and festivals such as the "*Great Lent*" found in many orthodox countries such as Greece (Mamouzelos, 2022). Consequently, The Mediterranean diet is also considered one of the most popular diets in the world (US News, 2022) due to its simplicity and the variety of dishes obtained with the least effort.

2.4 Mobile application solution

Existing ICH preservation efforts use technology to capture the cultural heritage, though the use of mobile technology is limited. Papangelis et al., 2016 attributes this effect to the preservation efforts merely focusing on documenting ICH and the fact that ICH is ever-changing. However, Papangelis et al., 2016 also believes that mobile devices are essential in reaching a greater audience and creating engaging interactions. Additionally, as discussed before, ICH is a socially engaging interaction. With that in mind, mobile devices being an extension of a person (Park & Kaye, 2019), it is only natural for a mobile application to exist such that the user can engage in ICH interactions.

2.4.1 Existing mobile applications for preserving ICH

Surprisingly, engaging ICH apps are exceedingly rare; The research for this thesis discovered three applications engaging in ICH preservation. Regardless, all lacked the social aspect, which is critical in ICH transmission. Specifically, Universidade do Algarve, 2022 and Digital Culture Monuments DCM IKE, 2022 provide geo-location-rich experiences, focusing on augmented and virtual realities. In addition, these apps focus on providing the context of tangible heritage and historical sites rather than ICH as a whole. A more interactive approach is Google LLC, 2022 in which the use of Artificial Intelligence and interactive features and games makes for a much more intriguing experience even though it still misses the critical social component. From the above literature, it is clear that there is a need for an application that users can use to access all and any ICH in their local community. This application must be engaging and intuitive while also being aesthetically pleasing and performant (Hoehle & Venkatesh, 2015).

2.5 Methodology review

The development process chosen for such an application is the scrum methodology, which incrementally builds a system in conditions where its requirements are constantly changing (Rising & Janoff, 2000). Specifically, the product owner generates and gathers all the system requirements, which are called stories, and all the stories compose the Product Backlog(Saranya P., 2017). In addition, this methodology uses small time frames, known as Sprints, which are less than four weeks in length and produce a working version of the system. Also, in scrum development teams, the task assignment involves the whole team (Despa, 2014). Finally, each agile iteration produces data, which is used to evaluate the efficiency of the development team (Martin, 2019).

In the book "*Clean Agile*", Martin R. mentions that at the core of project management lies the *Iron Cross*. This notion implies that no software project can achieve *good quality*, *completeness (scope)*, *low expenses (staff)*, and *fast speed of development* simultaneously. As such, agile methodologies often use *data* produced on each iteration to monitor the efficiency of the development team and make adjustments where necessary. This project will use the scrum methodology and its flexibility to deliver an Android application within the time frame specified.

In addition to the Scrum methodology, Jira has been chosen as the task management tool for the development process of the Android application. Jira is a popular tool that is commonly used to manage Agile projects and is well-suited to Scrum methodology. It provides essential features such as backlog management, sprint planning, and progress tracking, which are fundamental to effective project management. Jira enables the creation and prioritization of user stories, assignment of tasks to individuals, and real-time tracking of task progress. It also provides useful metrics such as burndown charts, velocity charts, and sprint reports that help evaluate performance and make data-driven decisions to enhance efficiency. Furthermore, Jira's customizable workflows allow for the tailoring of the tool to specific requirements and work processes. For example, custom statuses can be created, fields can be added to issues, and notifications can be configured to

suit individual needs. Overall, Jira is an excellent choice for task management in Scrum methodology, providing the necessary tools to manage work efficiently and effectively. By leveraging Jira and Scrum together, a successful outcome for the Android application development project can be ensured.

Chapter 3

Requirements

The requirements elicitation process of a software system is the most critical procedure of the development process. Errors in the requirements elicitation part propagate through the development process, and specifically, approximately 7 in 10 system errors are due to problems in the requirements specification (Rajagopal et al., 2005). Consequently, interviews, the available literature, surveys, brainstorming, and interface analysis will be utilised to gather the project's requirements.

ID	Feature	Requirement
H1	Authentication	Allow the user to register and login into the system.
H2	Authentication	Allow the user to register and log in using Google Sign-In.
H3	Nearby Elements	Allow the user to view their nearby elements in a list of cards.
H4	Nearby Elements	Provide a way to filter by element type and sort by nearest, popularity or recent.
H5	Nearby Elements	Allow the user to add a new element.
H6	Nearby Elements	Provide search functionality to search for any culture, element or contribution.
H7	Nearby Elements	Provide further information on a cultural element when it is clicked.
H8	Element Details	Allow the user to react and comment on every cultural element.
H9	Element Details	Allow the user to add a cultural element to their favourite list.
H10	Element Details	Allow the user to hide, block or report an element.
H11	Element Details	Allow the user to share an element.
H12	Element Details	Allow the user to check the element's contributions.
H13	Element Details	Allow the user to stream media related to the element.
H14	Contribution Details	Allow the user to react and comment on every contribution.
H15	Contribution Details	Allow the user to add a contribution to their favourite list.
H16	Contribution Details	Allow the user to hide, block or report a contribution.
H17	Contribution Details	Allow the user to share a contribution.
H18	Contribution Details	Allow the user to stream media related to the contribution.
H19	Add an element	Allow the user to select a location.
H20	Add an element	Allow the user to select a culture from a list of cultures.
H21	Add an element	Allow the user to add a culture that does not exist.
H22	Add an element	Allow the user to select an element type.
H23	Add an element	Allow the user to select an element title.
H24	Add an element	Allow the user to continue as a contribution if duplicate element.
H25	Add an element	Allow the user to add textual information.
H26	Add an element	Allow the user to add images, video or sound images.
H27	Add an element	Allow the user to link other elements.
H28	Add an element	Allow the user to select event information if it is an event.
H29	Add an element	Allow the user to continue using the application while uploading.
H30	Account	Provide a way for the user to manage their favourites.
H31	Account	Provide a way for the user to edit their display name, email and password.
H32	Account	Allow the user to log out.
H33	Account	Allow the user to delete their account.

Table 3.2: High priority requirements

ID	Feature	Requirement
M1	Account	Allow the user to upload a profile picture.
M2	Account	Allow the user to request a verification status.
M3	Account	Allow the user to review their black-listed content.
M4	Account	Allow the user to see and edit their added elements.

Table 3.4: Medium priority requirements

ID	Feature	Requirement
L1	Social	The user should be able to add and remove other users as friends
L2	Social	The user should be able to receive achievements and badges based on their interaction with the app.
L3	Social	The user should be able to select a group of friends, and a group of elements and create an "adventure".
L4	Social	The user should be able to ask and answer questions relating to a specific culture.

Table 3.6: Low priority (optional) requirements

ID	Feature	Requirement
NF1	Scalability	Make use of cloud services.
NF2	Usability	Follows the Material 3 UI guidelines.
NF3	Usability	Easy to use, users should not need to spend more than a couple of minutes of training to understand and use the app.
NF4	Performance	The system shall use XML over Jetpack Compose for the UI layer.
NF5	Performance	The application shall launch in at most 2 seconds.
NF6	Performance	The system shall not take more than 200 milliseconds on each call.
NF7	Security	All traffic shall be encrypted.
NF8	Security	The content shall be protected.
NF9	Support	The application will require Android SDK level 23 or later.
NF10	Interface	A PostgreSQL database for all data information.
NF11	Interface	Firebase Analytics for user behaviour analytics.
NF12	Interface	AWS Rekognition for content moderation.

Table 3.8: Non-Functional requirements

Chapter 4

Design

The design process for a system is a critical phase in software development, as it lays the foundation for the successful implementation of the solution (Jaiswal, 2019). In this regard, the choice of the design process and tools used plays a significant role in determining the outcome of the system design. For this particular system, the design process will adopt the C4 modelling process to create clear, concise, and comprehensive diagrams of the system architecture. The C4 modelling process is a structured approach that helps to identify the components of the system and their relationships, making it easier to understand the overall system design. Additionally, Figma, a user interface (UI) design tool, will be utilised to create mock designs of the application. Figma is a powerful tool that enables designers to create and collaborate on high-fidelity designs that accurately reflect the final product's appearance and functionality. By utilising these design tools, the resulting system will be well-structured and visually appealing, effectively meeting the needs of the intended users.

4.1 C1 - High-level Overview

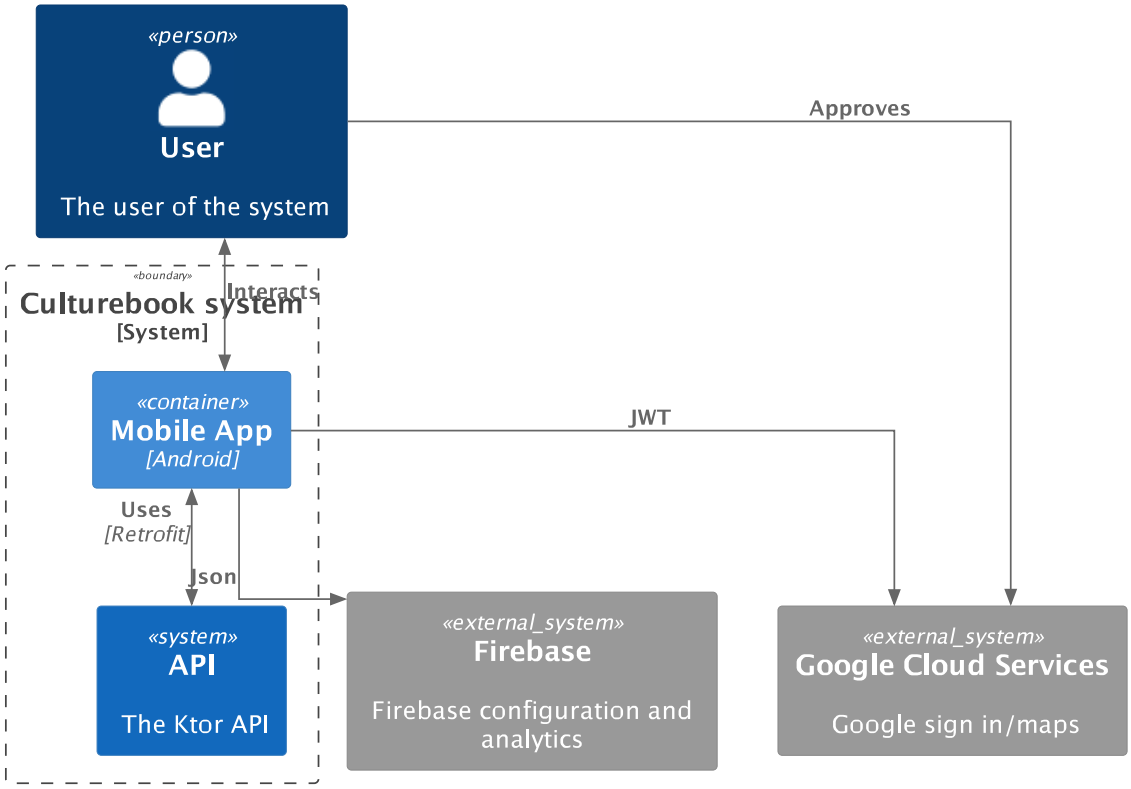

Figure 4.1: High-level overview of the system (C1)

Figure 4.1 is a diagram that employs the C4 model approach to provide a clear understanding of the system components and their interactions. The diagram consists of several system components, including a mobile app client, a Ktor API back-end, and external systems such as Google Cloud Services and Firebase. The diagram also includes relationships between system components, including API relations between the mobile app client and the Ktor API back-end using the Retrofit interface, and user relations such as the user's interaction with the mobile app client and approval of Google services.

The system is designed with a layered architecture, where each layer is responsible for a specific set of tasks. The top layer is the client-side mobile application, which communicates with the back-end API through an API interface. The middle layer is the back-end API, which provides the necessary services to the mobile application. The bottom layer is the database, which stores the data used by the API.

4.2 C2 - Detailed High-level Overview

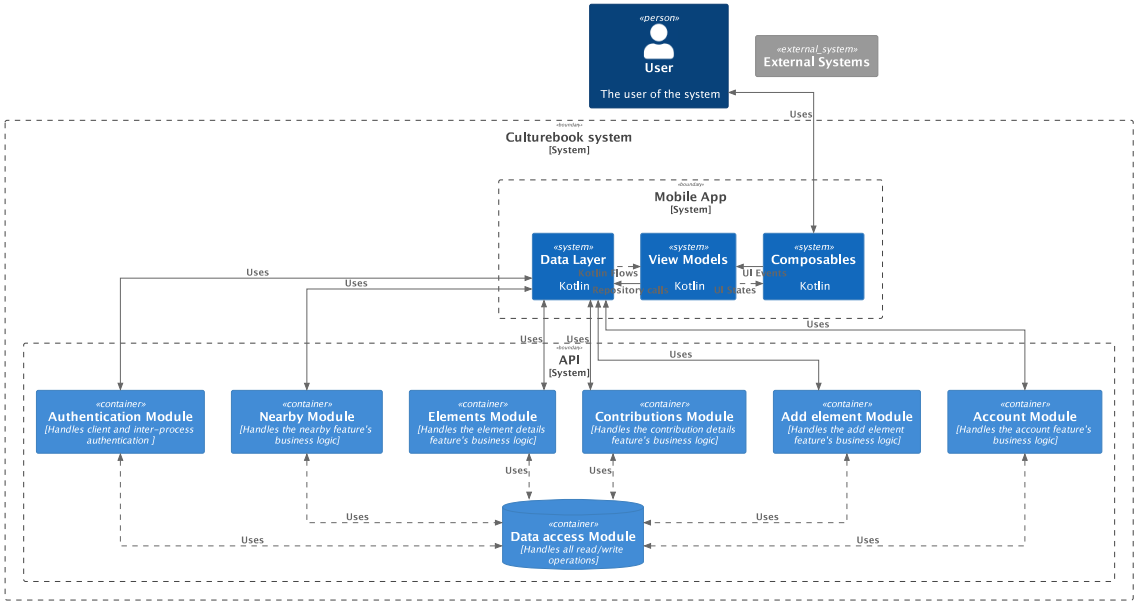

Figure 4.2: More detailed overview of the system (C2)

The Culturebook system is designed using a layered architecture approach as can be seen in the figure 4.2. The system consists of a client mobile app and an API back-end, which are encapsulated in their respective system boundaries. The mobile app client comprises three layers, including the Composables, View Models, and Data Layer, while the API back-end is composed of several modules, including Authentication, Nearby, Elements, Contributions, Add Element, Account, and Data access modules.

The diagram also shows the interdependencies between the different layers and modules, which are essential for the system's operation. For example, the data layer in the mobile app client uses relationships with the authentication, nearby, element details, contribution details, add element, account, and data access modules in the API back-end.

The system is designed to interact with external systems, as depicted by the External Systems component in the diagram.

Overall, the system is designed to be modular, flexible, and scalable, with a clear separation of concerns between the different layers and modules.

4.3 C3 - Detailed Overview

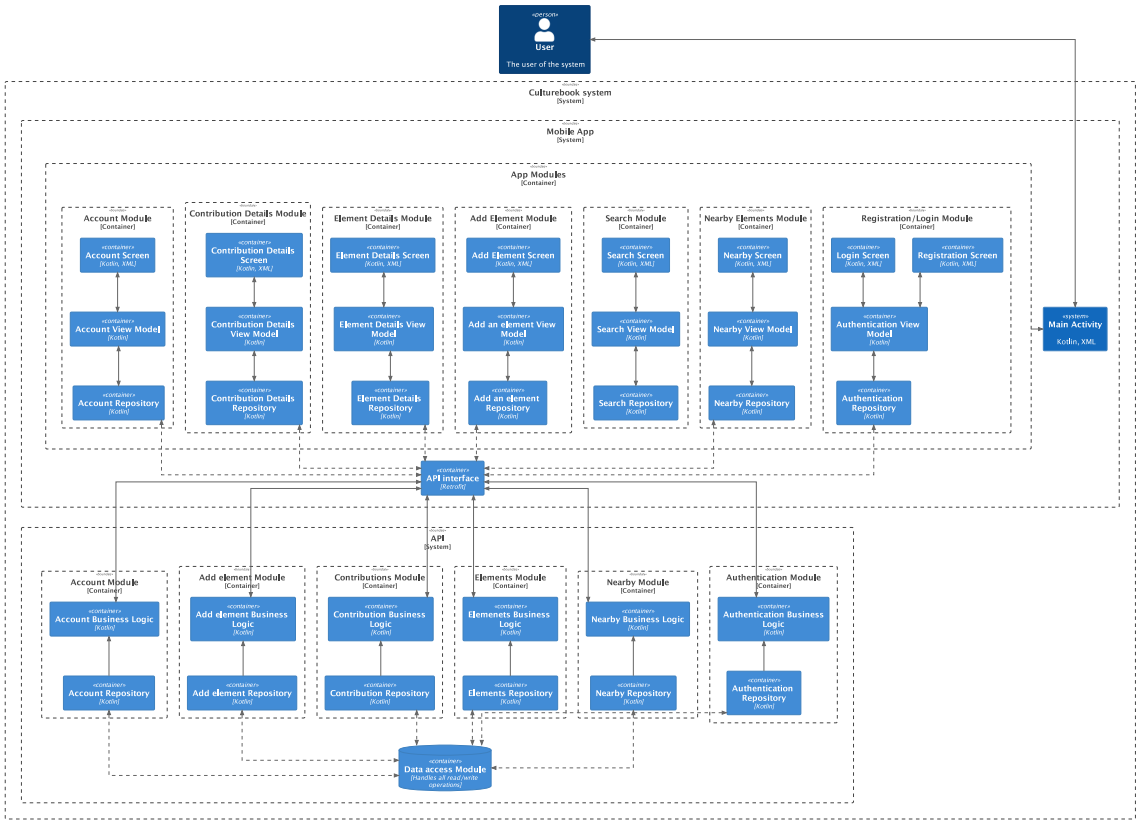

Figure 4.3: Detailed overview of the system (C3)

Figure 4.3 shows the system design which is divided into two main parts, a client-side mobile application, and a back-end API. The mobile application is further divided into multiple modules, each with its own set of containers. Each container represents a component of the module, such as screens, view models, and repositories. The modules of the mobile application are the Registration/Login Module, Nearby Elements Module, Search Module, Add Element Module, Element Details Module, Contribution Details Module, and Account Module. The API has modules for Authentication, Nearby, Elements, Contributions, and Add elements, each of which has its own set of containers and is responsible for performing specific tasks. For example, the "Authentication Repository" container is responsible for storing and retrieving user authentication data. The "Authentication Business Logic" container is responsible for handling the authentication logic.

4.4 Entity relationship diagram - ERD

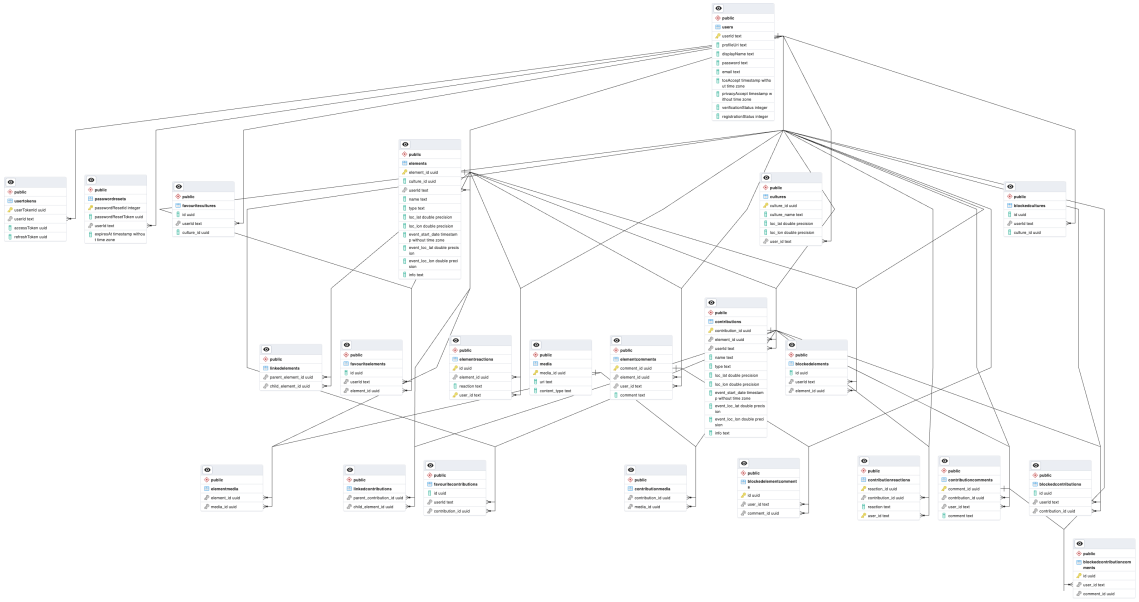

Figure 4.4: Entity relationship diagram

The database schema comprises several tables that enable users to interact with contributions, cultures, elements, and associated media files. These tables are designed to track user actions such as comments, reactions, and favourites, as well as enable the linking of contributions and elements. In particular, the schema includes tables for storing information about blocked contributions, cultures, and comments, as well as for linking elements with their parent or child contributions. Moreover, the schema includes tables dedicated to storing user-generated content, such as contributions and comments, along with associated media files. Collectively, these tables enable users to interact with and manage contributions, cultures, elements, and media files, thereby fostering a rich and dynamic community of users.

4.5 User Interface design

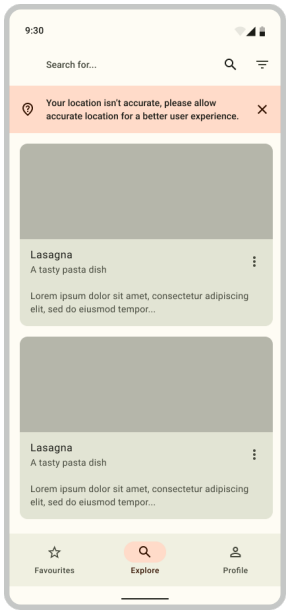

Figure 4.5: Example mock design from Figma

In the design process for the Culturebook system, the Material 3 design system was employed to create a consistent and coherent user interface (UI) design. Material 3 is a design language developed by Google that provides guidelines and resources for creating modern, responsive, and intuitive user interfaces. Moreover, Material 3 offers accessibility features that ensure that the application is usable by a wide range of users, including those with disabilities. The use of Material 3 in the design process of the Culturebook system was a decision based on the need to provide a visually appealing and user-friendly interface for the intended users. By adopting Material 3, the app is designed to not only be aesthetically pleasing but also functional and easy to use, thereby enhancing the user experience. Additionally, the use of Material 3 ensures that the application is consistent with other applications that use the same design system, which enhances familiarity and usability. Overall, the adoption of the Material 3 design system in the system design process was a prudent decision that resulted in a visually appealing, consistent, and user-friendly application that meets the needs of the intended users.

Chapter 5

Implementation

5.1 Software

In the implementation process of the Culturebook system, diverse software tools were employed, which varied according to the respective layer of the system. Specifically, the presentation layer was designed as an Android application, and therefore the software utilised in its development comprised Android Studio (v2022.1.1). The middle layer, or API, involved the use of multiple software tools, including JetBrains's IntelliJ IDEA (Community Edition, v2022.3.3), Docker Desktop (v4.17.0 (99724)), and the cloud-based hosting environment known as "Render". Finally, the database layer was supported by the cloud-based database provider known as "Supabase". In addition, several external services were used to aid the development, integration and deployment needs of the system.

Moreover, several external services were used to facilitate the development, integration, and deployment requirements of the system. The Version Control needs of the system were fulfilled by using "GitHub". The project's deployment needs were met by utilising "AppCenter", while "Firebase" was used to analyse app crashes and execute remote configuration. Additionally, the system uses the "IONOS" web-hosting service for domain registration and email needs. Although there was a plan to incorporate AWS Rekognition to filter inappropriate media, it was deemed unnecessary due to time constraints.

In conclusion, the implementation of the Culturebook system employed a range of software tools, both internal and external, to support its various layers and achieve its intended functionalities. Future work could potentially incorporate AWS Rekognition to enhance the system's filtering capabilities.

move to future work?

5.2 Methodology

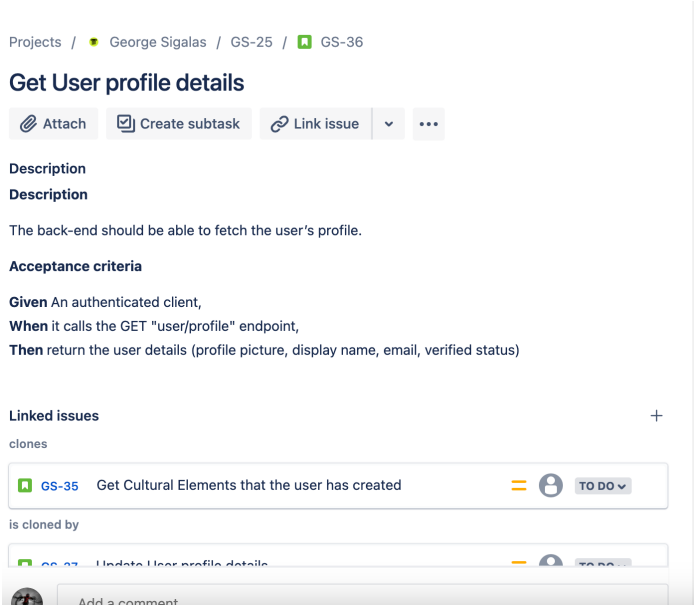

Figure 5.1: Example Jira task

The system development process utilised the Agile Scrum methodology, whereby the implementation was split into six sprints, each dedicated to a specific system feature. For instance, sprint 3 focused on the implementation of nearby cultural elements display within the application. The duration of each sprint ranged from two to four weeks, depending on the complexity of the feature. However, it should be noted that certain sprints exceeded their anticipated duration due to unexpected complexities. Moreover, the project workflow included multiple task states, such as the "To do" state, indicating tasks awaiting assignment, the "In progress" state, signifying tasks under active development, the "Document and Test" state, where tasks were evaluated against their requirements and documented, and lastly, the "Done" state, indicating the completion of the task. The chosen task management tool for the above workflow was Atlassian's Jira, due to its extensive use in the professional space.

5.3 System set-up

The requirements section **??** mandates that the system must exhibit high performance, security, and scalability. Additionally, different system environments should be established to ensure data integrity. Furthermore, given that the system interfaces with external services that necessitate the use of sensitive credentials, it is essential to provide a secure and flexible means of credential management. This mechanism must enable quick revocation and restoration of compromised credentials without adversely affecting the availability of the service.

5.3.1 CI/CD

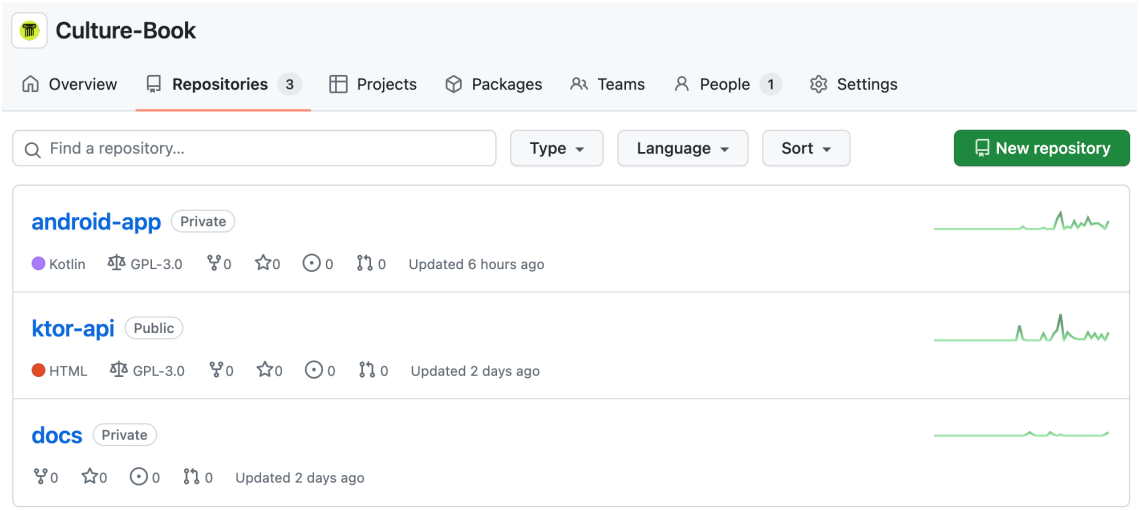

Figure 5.2: Github repositories for the Android app and ktor api

The CI/CD pipeline plays a crucial role in ensuring code quality by testing the API and Android app with every push to the version control system. This is achieved by utilising GitHub Actions, including the API Action and the Android Action. The API Action tests the source code by executing Gradle tests and using environment variables to pass necessary database information. The Android Action automates testing and deployment tasks for the app, triggered on push events to any branch, and generates a matrix of different Android API levels to test against. The workflow runs on macOS, setting up Java, caching Gradle dependencies and emulator files, creating an AVD, and running unit and instrumentation tests. Finally, the app is deployed to App Center using Fastlane, with three other actions utilised for testing and deploying the app to different environments.

```
    [...]
  - name: Create emulator and generate snapshot for caching
    if: steps.avd-cache.outputs.cache-hit != 'true'
    uses: reactivecircus/android-emulator-runner@v2
    with:
      api-level: ${{ matrix.api-level }}
      force-avd-creation: false
      emulator-options: -no-window -gpu swiftshader_indirect -noaudio
                        -no-boot-anim -camera-back none
      disable-animations: false
      script: echo "Generated AVD snapshot for caching."

  - name: Run instrumentation tests
    uses: reactivecircus/android-emulator-runner@v2
    with:
      api-level: ${{ matrix.api-level }}
      force-avd-creation: false
      emulator-options: -no-snapshot-save -no-window -gpu swiftshader_indirect
                        -noaudio -no-boot-anim -camera-back none
      disable-animations: true
      script: ./gradlew connectedDebugAndroidTest
    [...]
```

Listing 1: Snippet of the Android workflow

Challenges

The ktor API CI/CD pipeline implementation took around two weeks and was simple to execute, requiring only the creation of a Docker image. However, the development of the Android pipeline was challenging, with unit testing being relatively straightforward but integration testing being complicated due to the need for an Android device. Cloud-based environments struggled with the requirement for newer emulator versions for hardware acceleration, as well as the asynchronous operation and potential absence of Google Services integration. Emulator screen size and system animations further complicated matters.

Following thorough research, the issue surrounding the emulator was appropriately resolved by setting MacOS as the designated target operating system for the actions, which inherently supports hardware acceleration. Furthermore, the challenges concerning the asynchronous behaviour of the emulator, along with the issues regarding screen size and animation, were effectively addressed through the utilisation of the *reactivecircus/android-emulator-runner@v2* job. The aforementioned job effectively

```
suspend fun <T> dbQuery(
    coroutineContext: CoroutineContext? = null,
    block: suspend () -> T
): T =
    newSuspendedTransaction(
        context = coroutineContext ?: Dispatchers.IO
    ) {
        block()
    }
```

Listing 2: The dbQuery function.

manages the asynchronicity of the emulator, thereby simplifying the process of passing several Android Debug Bridge (ADB) arguments, ultimately resolving the remaining issues.

5.3.2 Database

The configuration of the database schema interaction in the ktor API is simplified through the inclusion of the Exposed SQL library and the JDBC API. To configure the database, environment variables and configuration files, particularly the *application.conf* HOCON file, are utilised to create a JDBC URL for the database connection. This configuration is then applied by a data source object to establish the database connection in the database module. Such configuration provides flexibility in separating the database and API, enabling the utilisation of diverse databases and SQL dialects by merely changing the API configuration.

Moreover, blocking Database queries is a major concern in web applications, which is addressed by the application's use of Kotlin's Coroutines. A utility function named dbQuery is created using Coroutines to perform database queries in a non-blocking manner and as such is employed throughout the API. The implementation of the function can be seen in Listing 2.

The setup and configuration of the database module were quite straightforward and the Exposed library simplifies all SQL queries, as such, the database module was built in the expected timeframe without issues.

5.3.3 API

The ktor API was initially intended to be constructed using a microservices architecture. However, due to the intricacy of such a system and the project's time constraints, it was determined that a monolithic approach would be followed instead. Nevertheless, the API has been partitioned into distinct modules, each responsible for a specific aspect of the system, as described in 4.2 of the design section. To achieve this separation of concerns, the modules are constructed using the *application.conf* configuration file. This guarantees that the communication between the modules is restricted and, as a result, maintains modularity within the system.

Challenges

Despite the original intention of implementing a clean architecture in the module design of the ktor API, due to the project's time constraints and the author's limited experience in back-end APIs, the result was an amalgamation of clean architecture modules and improvisations. For example, to abstract the data layers and implementation, several modules employ a database access object (DAO) interface, which proved to be helpful, but constant changes to the DAOs during development negatively impacted the process and were later decided to be dropped. Furthermore, although the Exposed library streamlined simple SQL queries, it also introduced boilerplate code and did not support certain SQL queries, such as "INSERT ... ON CONFLICT DO UPDATE," leading to inefficiencies.

5.3.4 Android

The Android application has been developed in a modular manner, which involves breaking it down into its core libraries and features. The core libraries, including the data, UI, navigation, and common libraries, were built in the current development sprint.

The data library employs the Retrofit library to handle HTTP connections and encapsulate them into API responses that can be processed by the application. During the wrapping process, the library performs checks for exceptions and errors, which are recorded in Firebase's Crashlytics for efficient debugging of future issues. Additionally, the external services required for the application, such as Firebase, Google Maps, and Google Sign-In,

were set up. This necessitated encrypting several credentials using Base64 and decoding them during the deployment process. Lastly, the "kotlinx.serialization" library was used to fulfil the application's serialisation requirements.

The UI library utilises the Material library for Jetpack Compose, which provides various UI components used throughout the application. Furthermore, the application's design system and app theme were implemented in this phase, including typography, spacing, and colour systems. Finally, common components, such as text fields and dialogs, were developed to serve as a reference for all UI components. This centralises the UI components and ensures that any changes in the future will not affect the application's consistency.

The navigation library handles the application's navigation and includes the app routes, deep links, and various navigation utilities.

Lastly, the common library comprises the common libraries and components utilised by several features.

Challenges

During the development process of the data library, several issues were encountered which negatively impacted the length of the sprint. One of the most significant problems was the inclusion of the "kotlinx.serialization" library. This serialisation library uses the name of the attributes of a class tagged with "@Serializable". On its own this is not a problem, however, in Android when building for a production/pre-production release it is customary to go through a process called obfuscation using a tool called R8. R8 performs various optimisations, such as dead code elimination, method and class renaming, and inlining of methods. This aims to make code significantly harder to reverse-engineer and also makes it more compact so that it takes less size. As such, libraries such as "kotlinx.serialization" that depend on the actual name of the attributes break relatively easily. The solution to this problem after thorough research was to include several "ignore" rules in the R8 configuration to ignore classes and attributes that are tagged with "@Serializable", however, this has the implication that the production artefact will be somewhat bigger in size and

easier to reverse engineer.

The decision to adopt Jetpack Compose as the preferred UI library was primarily driven by the need to develop faster user interfaces. While this objective was largely met, it is worth noting that Jetpack Compose is a relatively new technology, having been released only in 2021. As such, several UI components that are deemed essential and well-supported in the XML View system are either absent or their implementation is suboptimal.

The navigation library is responsible for managing the application navigation and presented one of the most challenging aspects of development. Jetpack Compose's navigation employs a string-based approach that differs from the XML navigation system which has been highly regarded. Passing arguments and objects through the navigation library proved unexpectedly difficult. The library assumes that each composable has a string-based route, requiring the use of the navigate() method with the route string as its argument for navigation. While this method is not problematic in itself, the process of navigating with arguments is obscured by the use of a URL argument styling that is not adequately explained in the documentation. Additionally, the NavController object required to execute the navigate() method must be passed from the root composable all the way down to the relevant composable or callbacks must be employed, resulting in the issue of "callback hell" and a violation of the separation of concern principle. Finally, the need to navigate to a screen and completely clear the navigation back stack to prevent a user from returning to the previous screen (e.g., in a Login screen) was found to be problematic. The current implementation of the library incorporates a "popUpTo()" method within a navigation builder to specify the destination at which the back stack will be popped up to. However, this method fails to function as intended, necessitating the creation of a custom method called "navigateTop()" to correctly clear the back stack and navigate to the next screen.

5.4 Features

5.4.1 Sprint 1 - Authentication

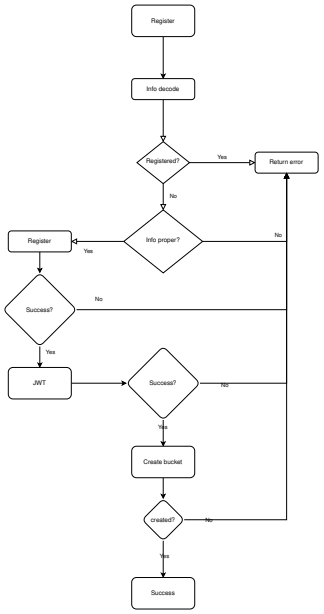

Figure 5.3: Registration flow

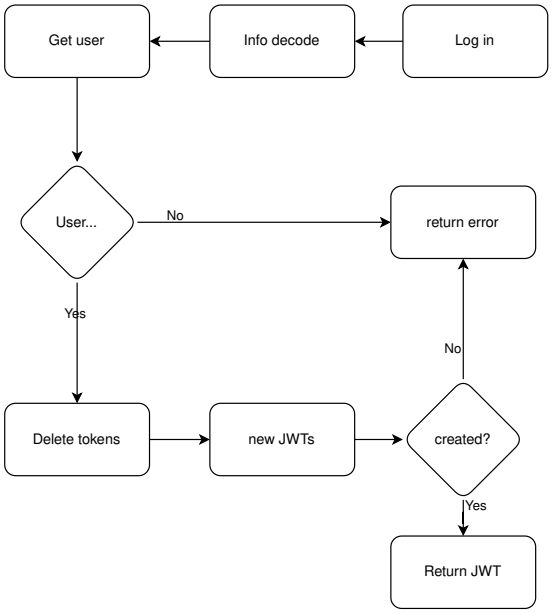

Figure 5.4: Login flow

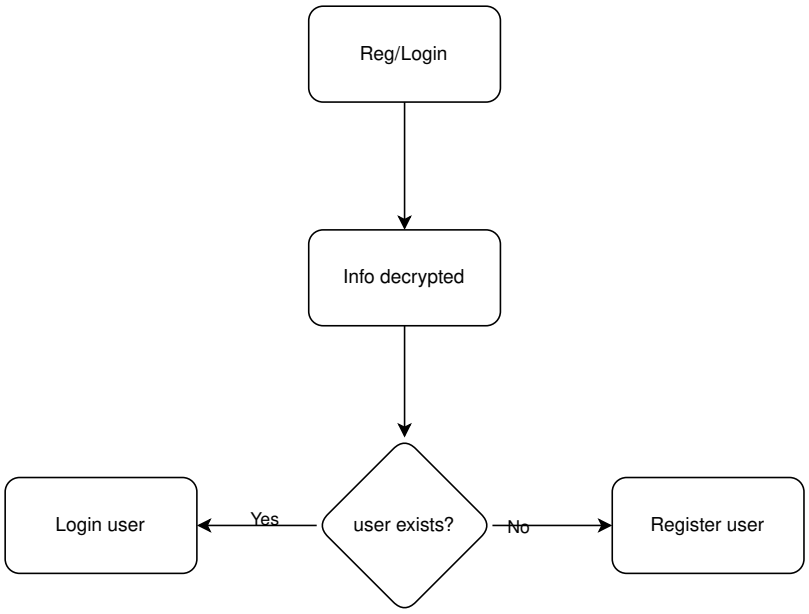

Figure 5.5: Registration or Login flow

As explained in **??** and as with most online applications today, there is a need for secure authentication services and registration/login functionality. Particularly, as this app is a social app in its nature, a user is required to have an account so that they can interact with the cultural elements in the app.

Implementation

Models: User, UserToken, VerificationStatusRequest, PasswordUpdateRequest, PasswordResetRequest, PasswordReset

Algorithms: HMAC256 (Kelly & Frankel, 2007), RSA (Moriarty et al., 2016)

The system employed the JSON Web Token (JWT) authentication technique to authenticate requests and users, chosen for its numerous advantages beyond enhanced security. This approach obviates the need for accessing the database for user lookup, as the token can be easily decoded to access all requisite authentication information.

In the context of the app, this authentication information encompasses an access token, a refresh token, and a user ID. Each request is accompanied by an Authorization header and a cookie. The API reads and decodes these elements, and if their respective tokens match, the user is authenticated. To enhance security, the JWTs associated with the cookie and

the header use different encryption keys.

Furthermore, prior to user login or registration, all client-sent traffic is encrypted using a public key provided by the API. This encryption ensures that all pre-authentication traffic is secure and resistant to malicious interception.

The authentication module relies on several data models, including User, UserToken, VerificationStatusRequest, PasswordUpdateRequest, PasswordResetRequest, and PasswordReset. These models each store data relevant to their designated use case. For example, the UserToken data model maintains records of the various tokens utilised for user authentication.

Challenges

Figure 5.6: Differences between Android and Ktor Encryption methods

During the development process, a significant challenge arose in relation to the encoding and decoding of JSON Web Tokens (JWTs) between the Android app and the ktor API. Specifically, issues arose during the generation of key pairs for the system using the RSA algorithm. The Android app employed a slightly modified Base64 decoder, which ultimately resulted in failed requests for correct encoding and decoding of the JWTs.

To address this issue, a solution was devised involving the use of a specific key format, PKCS8, which is universally recognised in Java encryption libraries. This format leverages the RSA algorithm to generate key pairs and, by utilising OpenSSL, enables the conversion of the private key format. This approach ultimately resolved the difficulties encountered during the encoding and decoding of JWTs.

An additional concern associated with the use of JSON Web Tokens (JWTs) is the increased size of each request. This is not considered desirable in terms of optimal system performance. In the future, steps could be taken to optimise the process by including only essential information within the token or implementing a web socket system that eliminates the need for verification with each request. Such measures would ultimately help to address the concern and improve the efficiency of the system.

5.4.2 Sprint 2 - Add Element

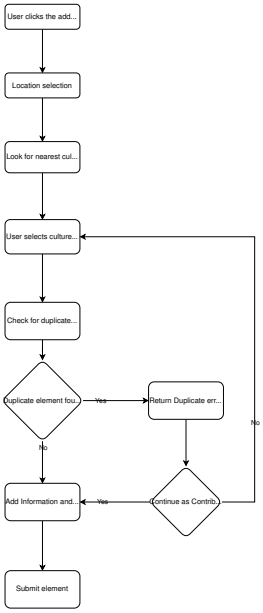

Figure 5.7: Add new cultural element flow

The functionality to add cultural elements is a crucial part of the system as explained in the previous chapters. This sprint deals with the requirements set in **??**. The goal of the sprint is for an authenticated user to be able to add a culture, a cultural element and a contribution to the system. This sprint took a big chunk of time as it laid out the data models, the relationships and the algorithms that are later used by the rest of the app.

Implementation

Models: Element, Culture, Contribution, Media, Location

During the implementation phase, the creation of data models was necessary for the system to operate effectively. Among these models was the Culture data model, which was responsible for representing a culture. Furthermore, the cultural aspects of each culture were represented by Element and Contribution objects, which were associated with several Media objects to provide users with an enriching experience.

As illustrated in Figure 5.7, the process of adding a new cultural element follows a specific sequence. Initially, the user is required to select the location of the element, which is crucial data used by the application to determine what content to display to users. Subsequently, the application searches for cultures in proximity to the selected location. The user can then either choose a culture from a list or create a new one. In the event that the user opts to create a new culture, the application ensures that the new culture name is not too similar to existing ones. This is achieved by utilising a custom fuzzy search algorithm that leverages cosine similarity, as delineated in Equation 5.1. The algorithm functions by transforming two strings into individual hash maps, which enumerate the total distinct characters in the string and calculate their cosine similarity.

After the user selects the location of the element and its corresponding culture, the user is prompted to specify the element type and title. These details are used to detect any duplicate elements. In the event that a duplicate is found, the user is prompted to continue adding a contribution. Subsequently, the user is requested to add further information such as media and textual content, as well as link other elements.

Challenges

The task of adding an element to the application was a complex undertaking, requiring direct manipulation of byte streams. Numerous considerations arose during the process, ranging from the element object itself to the handling of uploaded files. A significant portion of the implementation phase involved designing and constructing the media system for the application.

In Android applications, media uploading is typically accomplished by loading the entire

file to memory (Jaiswal, 2019) (Rahman & Gao, 2015). However, this method has notable drawbacks. Loading the entire file to memory is highly inefficient, can deplete the app's memory resources and lead to crashes, and negatively impacts battery life and performance. Additionally, directly accessing files can be insecure and necessitates explicit user permission to enable this functionality.

To address these challenges, I devised a solution that involved copying the file's content URI into the application's cache, along with the serialised element object. Subsequently, an "Upload Worker" was launched in the background, leveraging the *"androidx.work"* library collection, provided that sufficient battery power and network connectivity were available. This worker streamed multiple files from their original source, which proved to be more efficient than copying the file to a local directory, did not require additional permissions, and executed asynchronously. Furthermore, the backend implementation was designed to accommodate the streamed data input and appropriately forward it to the cloud services for storage.

The cosine similarity between two vectors $\vec{u}$ and $\vec{v}$ is defined as:

$$\text{similarity}(\vec{u},\vec{v}) = \frac{\vec{u}\cdot\vec{v}}{|\vec{u}||\vec{v}|} \tag{5.1}$$

where $\vec{u}\cdot\vec{v}$ is the dot product of the vectors $\vec{u}$ and $\vec{v}$, and $|\vec{u}|$ and $|\vec{v}|$ are the magnitudes (lengths) of the vectors.

5.4.3 Sprint 3 - Nearby Elements

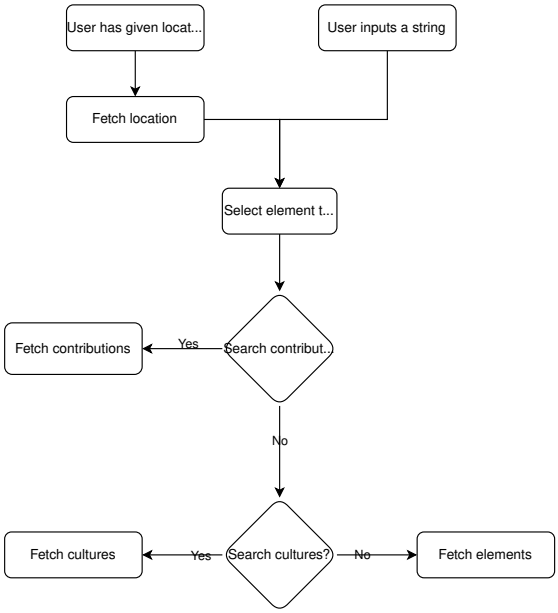

Figure 5.8: Nearby elements flow

Implementation

Models: BlockedElement, BlockedContribution, BlockedCulture, FavouriteElement, SearchCriteria

The nearby elements feature utilises the location API provided by Google services to determine the user's location. This information, together with various filters, constitutes the SearchCriteria object, which is employed to retrieve the element, contribution, or culture objects. Besides obtaining and displaying nearby elements, this sprint introduced the blocking and favouring functionality. The blocking feature enables users to click the "block" button for a specific element, contribution, or culture, thereby adding the corresponding object to a table. Subsequently, when querying any type of object, the API queries the table to check for any blocked objects before returning the results. Analogously, the favouring functionality adds the object's id to a table that is queried to determine if the user has favourited the object in question.

It is noteworthy that the objects retrieved from the nearby screen have been optimised to reduce the amount of text displayed to only two sentences and to decrease the quality of the media. Furthermore, the image and video media are cached on disk to conserve

network bandwidth and enhance performance.

Challenges

A significant challenge that arose during the development of the sprint was the lack of a satisfying implementation of the modal bottom sheet component in the Material library for Jetpack Compose. Specifically, the modal bottom sheet dialog XML equivalent is fragment-based and can be easily called from any part of the codebase, thus aligning well with the separation of concerns principle. In contrast, the implementation of the modal bottom sheet dialog in Jetpack Compose requires wrapping the entire application's content within it, thereby undermining the separation of concerns principle and making it challenging to include multiple bottom sheets. Therefore, a considerable amount of time was expended in developing a custom modal bottom sheet that emulates the XML equivalent. The custom composable enables the display of a bottom sheet dialog that can be swiped up and down. It accepts several parameters that enable customisation of the dialog's appearance and behaviour, including header and footer composables, as well as onDismiss and onConfirm callbacks.

5.4.4 Sprint 4 - Element Details

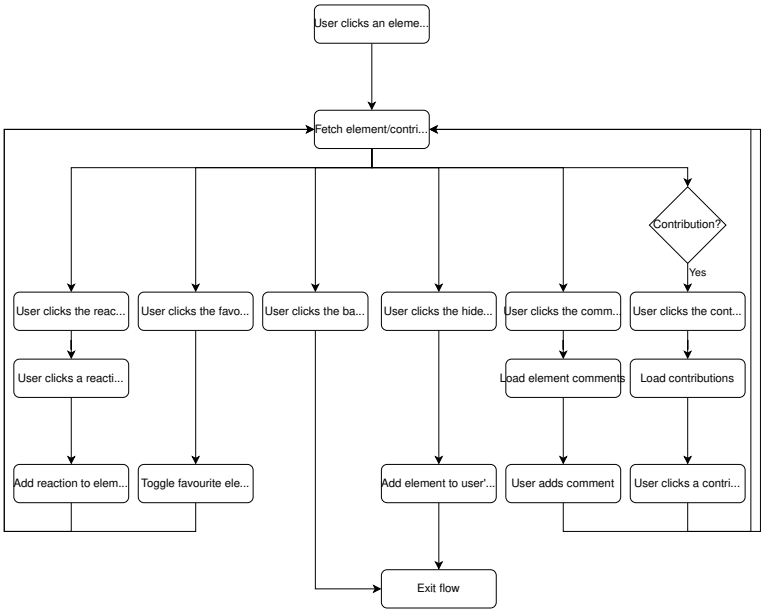

Figure 5.9: Element details flow

Implementation

Models: Comment, Reaction, BlockedComment

This sprint centred on the development and implementation of the element and contribution details screen, with emphasis on introducing the comment and reactions functionality. While the backend implementation had been established in prior sprints, this iteration relied on the existing system to build upon.

Regarding the comment feature, the client would post a comment, which would then be linked to the corresponding element or contribution by the server. Additionally, a hide/block/report feature, similar to other user-generated content within the application, was incorporated. This functionality did not pose any great challenges as it was built upon existing systems.

However, implementing the reactions functionality posed a greater challenge. Each reaction was represented by a UTF-8 encoded string that signified an emoji. To render these emojis accurately, the application leveraged the *"androidx.emoji2"* library collection, which identified the text encoded in UTF-8 and displayed the appropriate system emoji. The system also ensured that each user could only provide a single reaction, achieved through the removal of previous reactions before adding a new one. While this solution may be inefficient, it was found to be effective in the current implementation.

Challenges

As has been previously noted, the navigation component of Jetpack Compose demonstrates several areas in which it is lacking, particularly during the development of the current sprint. A significant amount of effort was dedicated to understanding the navigational arguments, which employ a URL-style navigation approach as opposed to the traditional Android standard, resulting in a somewhat unintuitive implementation.

Furthermore, the implementation of the element details screen and the contribution details screen proved challenging, as the two screens share many similarities with only a few

differences. As a result, the same components were utilised with conditional statements, which some may perceive as violating the principle of separation of concerns.

Finally, the use of the emoji2 library within Android's View system to display emojis, in conjunction with the necessity of nested Lazy components for the emoji bottom sheet, resulted in suboptimal performance when displaying the bottom sheet containing the emojis.

5.4.5 Sprint 5 - Account

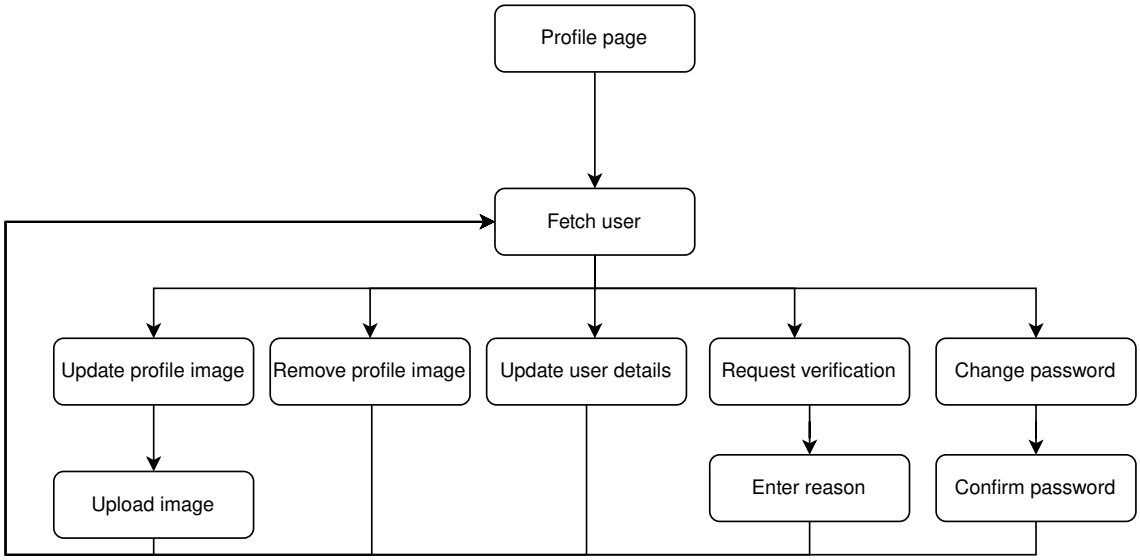

Figure 5.10: Account - Profile

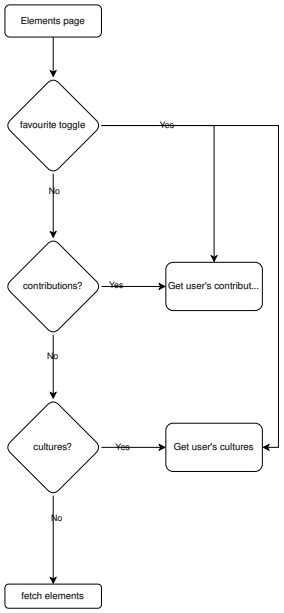

Figure 5.11: Account - My Elements

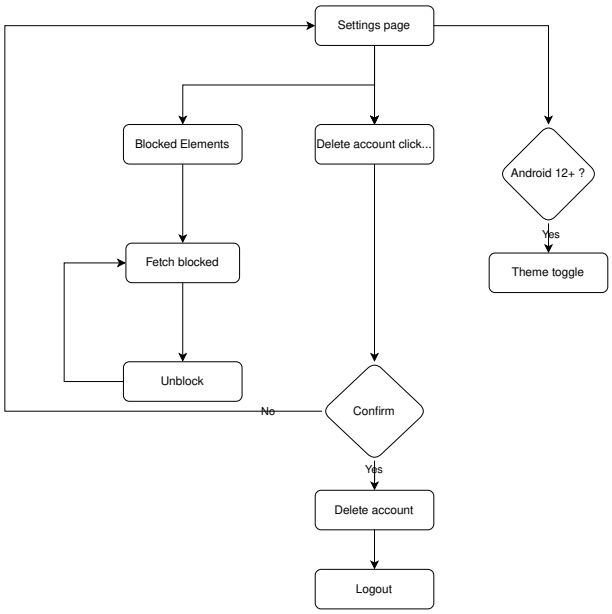

Figure 5.12: Account - Settings

The culminating phase of the project involved integrating the "account" section of the mobile application, which encompasses both user-oriented functionalities and application-specific settings. The account section is subdivided into four primary categories, namely profile, "My Elements", settings, and about sections, each of which serves a distinct purpose. The profile section, for instance, facilitates the user's ability to update their personal information.

Profile

During the development of the profile section, a range of functionalities had to be incorporated. These functionalities included enabling users to modify their information, such as email, display name, password, and profile image. Furthermore, users were provided with the option to request a verification status, which is indicative of their credibility as a source of information. The verification process is currently executed manually. Specifically, when a user requests a verification status, an email is dispatched to the application's main email address ("no-reply@culturebook.co.uk"), which specifies the user's id and reasoning for the request.

My Elements

The elements section of the application exhibits information regarding the elements generated by the user or those marked as favourites. Users are permitted to delete their created elements. However, due to time constraints, an editing functionality has not been incorporated yet.

Settings

The settings section of the application offers users an option to delete their accounts as well as to review blocked content. The "Exposed" object-relational mapping library provides a convenient means to eliminate user-generated content by defining the deletion strategy of the relevant tables. Consequently, once the user object is deleted, the deletion cascades through the database, removing all linked data. However, in light of the app's objective of preserving cultural elements, a deliberate decision was taken not to delete the elements, contributions, or cultures generated by a user. This is due to the fact that removing these objects would result in the deletion of other dependent objects. This would result in a considerable loss of data if a user, especially a verified user with a large volume of content, elects to delete their account.

About

Finally, the About section of the application provides comprehensive information regarding the project in its entirety, the Terms and Conditions governing the use of the application, and the Privacy Notice. Both the Terms and Conditions and Privacy Notice documents are stored in a Firebase storage container to facilitate efficient updating.

Challenges

During the development of the sprint, the primary challenge encountered pertained to the deletion of content. Notably, the user table was found to be referenced by several other tables, including the Elements, Contributions, and Cultures tables. Consequently, any decision by a user to delete their account would have a cascading effect on these tables, leading to unintended deletions. To circumvent this challenge, the reference column of the user table was made nullable so that objects associated with the user are not deleted upon

account deletion. Moreover, to comply with data protection laws, no identifiable information is contained within the element, contribution and culture objects. In addition, the element object deletions can also affect contribution objects. As such, the contributions element id column was made nullable to address this issue. In the future, self-repairing algorithms may be incorporated into the deletion processes of element objects to guarantee the integrity of contribution objects by identifying and assigning them to the next nearest related element.

Chapter 6

Evaluation

ID	Feature	Implemented	Notes
H1	Authentication	Yes	Registration/Login functionality working.
H2	Authentication	Yes	Registration/Login through Google Sign-In working.
H3	Nearby Elements	Yes	Displays nearby elements in a card list.
H4	Nearby Elements	Partially	Filters working but limited to the element type.
H5	Nearby Elements	Yes	User is able to add an element from the nearby screen.
H6	Nearby Elements	Yes	Search functionality working.
H7	Nearby Elements	Yes	Element navigates to details when clicked.
H8	Element Details	Yes	Comments/Reactions working.
H9	Element Details	Yes	Favourite functionality working.
H10	Element Details	Partially	Hide/Block functionality works, Report hides the content but doesn't report it.
H11	Element Details	Yes	Share functionality working.
H12	Element Details	Yes	Element contributions working.
H13	Element Details	Yes	Media streaming working.
H14	Contribution Details	Yes	Contribution Comments/Reactions working.
H15	Contribution Details	Yes	Contribution favourite functionality working.
H16	Contribution Details	Partially	Contribution hide/block working, Report hides the content but doesn't report it.
H17	Contribution Details	Yes	Contribution share functionality working.
H18	Contribution Details	Yes	Contribution media streaming working.
H19	Add an element	Yes	Location selection working.
H20	Add an element	Yes	Culture selection working.
H21	Add an element	Yes	Culture insertion working.
H22	Add an element	Yes	Element type selection working.
H23	Add an element	Yes	Element title selection working.
H24	Add an element	Yes	Contribution insertion working.
H25	Add an element	Yes	Textual information insertion working.
H26	Add an element	Yes	Media insertion working.
H27	Add an element	Partially	Linked elements can be added, but currently they do nothing.
H28	Add an element	Yes	Event information insertion working.
H29	Add an element	Yes	Asynchronous uploading working.
H30	Account	Yes	Favourite management working.
H31	Account	Yes	User details edit working.
H32	Account	Yes	User log out working.
H33	Account	Yes	Account deletion working.

Table 6.2: High priority Requirements

ID	Feature	Implemented	Notes
M1	Account	Yes	User profile image can be updated.
M2	Account	Yes	User can request verification status.
M3	Account	Yes	Users can manage their blocked content.
M4	Account	Partially	Users can see and delete their content, but not edit them.

Table 6.4: Medium priority Requirements

ID	Feature	Implemented	Notes
L1	Social	No	Not implemented due to time constraints.
L2	Social	No	Not implemented due to time constraints.
L3	Social	No	Not implemented due to time constraints.
L4	Social	No	Not implemented due to time constraints.

Table 6.6: Low priority (optional) Requirements

ID	Feature	Implemented	Notes
NF1	Scalability	Yes	The system makes use of cloud services.
NF2	Usability	Yes	The app follows the Material3 guidelines.
NF3	Usability	Yes	By following the Material3 guidelines, the app is easy to use.
NF4	Performance	No	The app uses Jetpack compose. The decision to choose this is explained in the implementation.
NF5	Performance	Yes	The app launches in 853ms.
NF6	Performance	Partially	Due to the use of a free cloud-hosting service, the back-end VM can take up to 60 seconds to boot up.
NF7	Security	Partially	Only unauthorised traffic is encrypted.
NF8	Security	Yes	Content is accessed only by authorisation.
NF9	Support	No	The Android app targets SDK level 26+. This is due to some libraries' requirements.
NF10	Interface	Yes	The "Supabase" cloud database employs a PostgreSQL database.
NF11	Interface	Partially	Firebase analytics are implemented, however, not all user actions are recorded.
NF12	Interface	No	Not implemented due to time constraints.

Table 6.8: Non-Functional Requirements

Chapter 7

Testing

Various methods and processes were employed to evaluate the system's functionality. As delineated earlier, the system's development and integration processes rely on a continuous integration and continuous development pipeline. This pipeline utilises a version control system to execute unit and instrumentation tests, which are complemented by code quality screening to ensure the absence of any critical issues. The infrastructure for conducting these tests was facilitated through the use of the Gradle build tool and multiple bash scripts. Each layer of the system was subjected to distinct testing methodologies.

7.1 Back-end

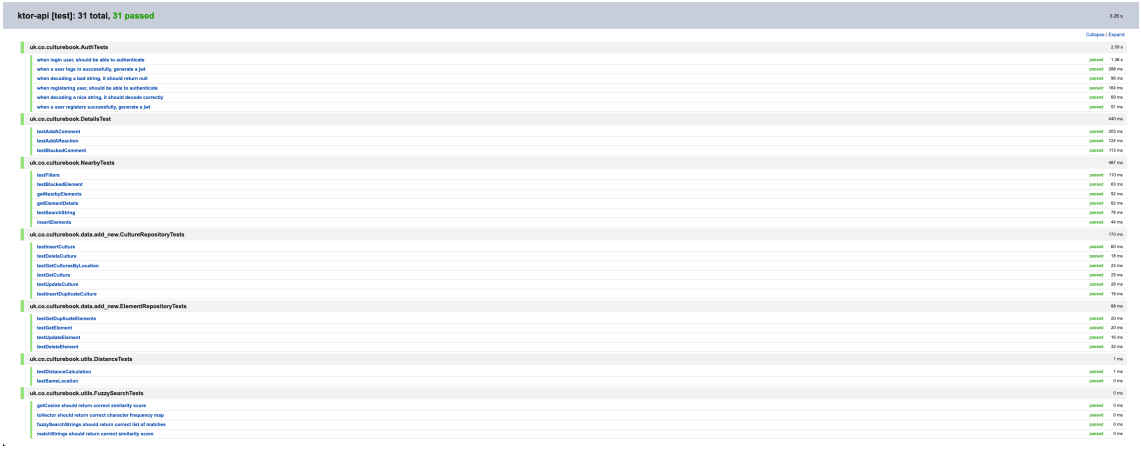

Figure 7.1: Test Results on Ktor API

The backend component of the system underwent numerous testing approaches, comprising unit testing to verify the accuracy of utility methods, such as the fuzzy search algorithm and encryption/decryption techniques. Moreover, integration testing was conducted to assess various areas of the system, such as the addition of an element or the retrieval of nearby elements. Additionally, the majority of the tests aimed to substantiate the fulfilment of each system requirement, consequently adopting a combination of integration and acceptance testing methods.

In addition, by employing the CodeQL analysis tool, the backend was able to detect several API keys that were accidentally pushed to the repository. Furthermore, it provided suggestions for the encryption algorithm to make it more secure.

7.2 Front-end

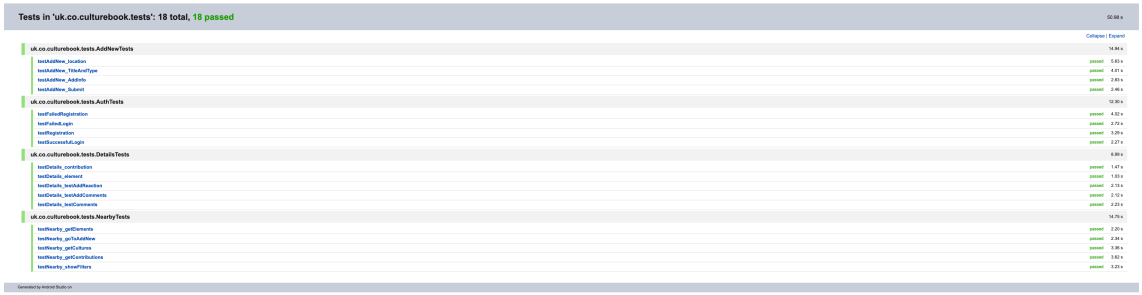

Figure 7.2: Test Results on Android

The frontend part of the system employed an instrumentation testing strategy to complement the backend testing. This is because most of the logic is happening on the backend, and therefore, the need for specific unit testing on the Android application is not necessary. As such, the frontend testing safeguards the app against future issues that may break the navigation and UI elements.

Similarly to the backend, the system uses a CodeQL analysis tool. This tool picked up several leaked API keys that were accidentally leaked in the Firebase default values XML file. The CodeQL analysis tool also provided suggestions for making the encryption method more secure.

Chapter 8

Conclusion

8.1 Future work

8.2 Reflection

References

Ballard, L.-M. (2008). Curating intangible cultural heritage. *Anthropological Journal of European Cultures*, *17*(1), 74–95. http://www.jstor.org/stable/43235374

Brimblecombe, P., & Hayashi, M. (2017). *Perception of the relationship between climate change and traditional wooden heritage in japan* (Dawson, C. Nimura, E. Lopez-Romero, & M.-Y. Daire, Eds.). Oxbow Books.

CBI. (2018). The european market potential for cultural tourism [Accessed: 2022-11-10]. https://www.cbi.eu/market-information/tourism/cultural-tourism/market-potential

Cominelli, F., & Greffe, X. (2012). Intangible cultural heritage: Safeguarding for creativity. *City, Culture and Society*, *3*(4), 245–250. https://doi.org/https://doi.org/10.1016/j.ccs.2012.10.003

Commission, E. (n.d.). Cultural tourism [Accessed: 2022-11-10]. https://single-market-economy.ec.europa.eu/sectors/tourism/offer/cultural_en

Despa, M. L. (2014). Comparative study on software development methodologies. *Database Systems Journal*, *05*(03), 44–45.

Digital Culture Monuments DCM IKE. (2022). Culture app [Accessed: 2022-11-12]. https://play.google.com/store/apps/details?id=com.cv.culture.app

Efentaki, K., & Dimitropoulos, V. (2015). Economic perspectives of intangible cultural activities [Proceedings of the 3rd International Conference on Strategic Innovative Marketing (IC-SIM 2014)]. *Procedia - Social and Behavioral Sciences*, *175*, 415–422. https://doi.org/https://doi.org/10.1016/j.sbspro.2015.01.1218

George, E. W. (2010). Intangible cultural heritage, ownership, copyrights, and tourism. *International Journal of Culture, Tourism and Hospitality Research*.

Goeldner, C., Ritchie, J., & McIntosh, R. (2000). *Tourism: Principles, practices, philosophies*. Wiley. https://books.google.co.uk/books?id=pUXrAAAAMAAJ

Google LLC. (2022). Google arts & culture [Accessed: 2022-11-12]. https://play.google.com/store/apps/details?id=com.google.android.apps.cultural

Hennessey, B. A., & Amabile, T. M. (2010). Creativity [PMID: 19575609]. *Annual Review of Psychology*, *61*(1), 569–598. https://doi.org/10.1146/annurev.psych.093008.100416

Hoehle, H., & Venkatesh, V. (2015). *Mobile application usability: Conceptualization and instrument development.* (tech. rep.). MISQ/2015/39.2.08. https://doi.org/10.25300

Jaiswal, M. (2019). Software architecture and software design. *International Research Journal of Engineering and Technology (IRJET) e-ISSN*, 2395–0056.

Kelly, S., & Frankel, S. (2007). *Using hmac-sha-256, hmac-sha-384, and hmac-sha-512 with ipsec* (tech. rep.).

Lenzerini, F. (2011). Intangible cultural heritage: The living culture of peoples. *European Journal of International Law*, *22*(1), 101–120. https://doi.org/10.1093/ejil/chr006

Mamouzelos, G. (2022). Sarakosti, signifying 40 days of great lent [Accessed: 2022-11-10]. https://greekcitytimes.com/2022/03/07/sarakosti-signifying-40-days-lent/

Martin, R. (2019). *Clean agile: Back to basics*. Pearson. https://books.google.co.uk/books?id=vZCZxAEACAAJ

Marzuki, A. (2011). Resident attitudes towards impacts from tourism development in langkawi islands, malaysia. *World Applied Sciences Journal*, *12*(2), 25–34.

Moriarty, K., Kaliski, B., Jonsson, J., & Rusch, A. (2016). *Pkcs# 1: Rsa cryptography specifications version 2.2* (tech. rep.).

Mountcastle, A. (2010). Safeguarding intangible cultural heritage and the inevitability of loss: A tibetan example. *Studia ethnologica Croatica*, *22*(1), 339–359.

Papangelis, K., Chamberlain, A., & Liang, H.-N. (2016). New directions for preserving intangible cultural heritage through the use of mobile technologies, 964–967. https://doi.org/10.1145/2957265.2962643

Park, C. S., & Kaye, B. K. (2019). Smartphone and self-extension: Functionally, anthropomorphically, and ontologically extending self via the smartphone. *Mobile Media & Communication*, *7*(2), 215–231. https://doi.org/10.1177/2050157918808327

Rahman, M., & Gao, J. (2015). A reusable automated acceptance testing architecture for microservices in behavior-driven development. *2015 IEEE Symposium on Service-Oriented System Engineering*, 321–325. https://doi.org/10.1109/SOSE.2015.55

Rajagopal, P., Lee, R., Ahlswede, T., Chiang, C.-C., & Karolak, D. (2005). A new approach for software requirements elicitation. *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network*, 32–42. https://doi.org/10.1109/SNPD-SAWN.2005.5

Reguant-Aleix, J., Arbore, M. R., Bach-Faig, A., & Serra-Majem, L. (2009). Mediterranean heritage: An intangible cultural heritage. *Public Health Nutrition*, *12*(9A), 1591–1594. https://doi.org/10.1017/S1368980009990413

Reimann, L., Vafeidis, A. T., Brown, S., Hinkel, J., & Tol, R. S. J. (2018). Mediterranean unesco world heritage at risk from coastal flooding and erosion due to sea-level rise. *Nature Communications*, *9*(1), 4161. https://doi.org/10.1038/s41467-018-06645-9

Rising, L., & Janoff, N. (2000). The scrum software development process for small teams. *IEEE Software*, *17*(4), 26–32. https://doi.org/10.1109/52.854065

Saranya P., P. A., Monica V. (2017). Comparative study of software development methodologies. *International Research Journal of Engineering and Technology (IRJET)*, *04*(05), 176.

Simpson, N. P., Clarke, J., Orr, S. A., Cundill, G., Orlove, B., Fatorić, S., Sabour, S., Khalaf, N., Rockman, M., Pinho, P., Maharaj, S. S., Mascarenhas, P. V., Shepherd, N., Sithole, P. M., Ngaruiya, G. W., Roberts, D. C., & Trisos, C. H. (2022).

Decolonizing climate change–heritage research. *Nature Climate Change*, *12*(3), 210–213. https://doi.org/10.1038/s41558-022-01279-8

Tan, S.-K., Tan, S.-H., Kok, Y.-S., & Choon, S.-W. (2018). Sense of place and sustainability of intangible cultural heritage – the case of george town and melaka. *Tourism Management*, *67*, 376–387. https://doi.org/https://doi.org/10.1016/j.tourman.2018.02.012

UNESCO. (1954). Convention for the protection of cultural property in the event of armed conflict. https://en.unesco.org/protecting-heritage/convention-and-protocols/1954-convention

UNESCO. (1972). Convention concerning the protection of the world cultural and natural heritage. https://whc.unesco.org/en/conventiontext/

UNESCO. (2003). Text of the convention for the safeguarding of the intangible cultural heritage. https://ich.unesco.org/en/convention

Universidade do Algarve. (2022). Iheritage [Accessed: 2022-11-12]. https://play.google.com/store/apps/details?id=pt.ualg.milage.iheritage%5C&hl=en%5C&gl=US

US News. (2022). The best diets overall 2022 - expertly reviewed [Accessed: 2022-11-10]. https://health.usnews.com/best-diet/best-diets-overall?src=usn_pr

WTTC. (2021). Travel & tourism economic impact [Accessed: 2022-11-08]. https://wttc.org/research/economic-impact

Z, T. I. (n.d.). Historical building [Accessed: 2022-11-10]. https://thenounproject.com/browse/icons/term/historical-building