Haaga-Helia

Developing Android health care app integration

Tomi Ahtola

Haaga-Helia University of Applied Sciences

Bachelor’s Thesis

2022

Degree Programme in

Business Information Technology

Abstract

Author(s) Tomi Ahtola
Degree Bachelor of Business Administration
Report/thesis title Developing Android health care app integration
Number of pages and appendix pages 21+1

This thesis report covers the background of the commissioned thesis and the software components that the project produced. The author will discuss and define the goals, scope and objectives of the thesis project in the introduction chapter.

After introducing and establishing the project management, author discloses some research and how it was applied in the product that the thesis project produced. This was done by referring to the Android Developer technical reference manual. The author also used an article as source and research material for writing unit tests how it was used in this project.

After research had been concluded author discussed software development methodologies and tools used in this project. The author also discussed how methodologies were adapted in the commissioning party.

Implementation of the product that was created by this thesis project was presented next. The author discusses how the implementation process started, how to achieve what was required in the description of the mininum viable product. The research done earlier is connected to the information and skills required in the implementation. The author references a 3rd party source and project in a public code repository, assesses its usefulness to this project and extracts necessary knowledge and utilizes it to create software component.

Product chapter discusses code review that is required to maintain the integrity and quality of the code commited to master repository of the product. The author had two code reviews, that he got feedback from. Necessary suggested changes and feedback were taken into account and the software components were commited to master repository. The author also visualises functionality of the created software components and its required behaviour.

Conclusions of the report discuss how the result of this project matches the expectations, goals and objectives set in the introduction and planning phase of the project. The author reflects on research, methodologies, tools, and practises and how they were used in the project. He also discloses the latest status of the product in the commissioning party, and how does its future look going in to the production environment.

Keywords
Software Development, Android developer, Code review, Project management, Unit test

Table of contents

1 Introduction

This report was written to support the project that was planned and executed to provide business value to the commissioning party in form of expanding upon the existing product.

There is an underlying implementation in the product that can be modified, integrated and/or used to create new supporting implementations for the new applications. The author used these to formulate better understanding of the product, needed information, knowledge, and skills to create improvements and support for the 3rd party applications.

The tasks of this project were to study, plan, create, implement, and document the solutions for supporting for 3rd party applications, so that the solutions met the functional requirements, the commissioning party's development guidelines and criteria, and can be deployed into production.

The target and the objectives of the project was to finish in the allocated time and using the available resources.

The project itself produced:

- Research
- Product
  - Software component
- Documentation
  - Report
  - System design specification documentation for the commissioning party

The product created in this project was necessary for the commissioning company. It was part of a commitment in an epic. The epic described a minimal viable product (MVP), that was to provide support and functionality to a 3rd application within the company's existing product. The MVP features that were required:

- application switching
- possible to launch and switch between the applications
- notifications
- would be shown in both applications

Relevance of this project was high. The product management had described and outlined the product and its requirements in the epic. The relevance for this project was decided by the efficiency of the implementation versus the goals and objectives that were set. Since the functional requirements were set early in the project, the author planned and executed the implementation within these requirements. To ensure the efficiency of the outcome the project adhered to company’s internal guidelines and procedures, such as the code review. Code review practises excluded most, if not all – irrelevant, inefficient, and out-of-scope issues, code, and solutions.

Research was conducted along the side of sub-tasks, tasks, and stories in the epic as required to proceed. More research was conducted as required by the author to fulfil technical requirements or challenges that appear during the project. The results from research supported the development efforts and made it possible to produce modern code, that adhered to latest conventions and standards set by professionals in the same field of expertise.

The author learned to investigate, research, and apply modern software development methods The author also learned how to work as a software developer in professional setting. The author planned, executed, and accomplished a set of goals, planned in the program increment planning meeting and in the sprint planning meetings.

The project did not cover:

- Any previously created solutions or code in the existing product that is unrelated to the project itself.
- Any functionality of the 3rd party application which was not described by product management.

This project was commissioned and conducted at the employ of the commissioning party. The commissioning party offered tools, software, equipment, training, and the environment to complete the project.

The commissioning party is an international software company, and an industry leader in its field. The company provides modern software, services, and tools in the collection of electronic Clinical Outcome Assessments (eCOA). One of the products of the commissioning party is a software product that is used to collect eCOAs, which are then capturing data and information electronically through clinical trials. It is a modern way to assure that the data created and collected in clinical trials meets the acceptance criteria of quality and integrity. The software component itself makes it easier to manage clinical trials in which they usually manage and process hundreds, thousands, tens of thousands or more clinical trial subjects.

2 Planning

2.1 Working plan

During the project, the author focused on topics and software development items that were on the sprint boards and were related to this project. The items were described in the epic that was established in the program increment planning meeting. The epic was then split into stories and the stories were then divided into tasks and sub-tasks. The tasks and sub-tasks were set, monitored, and modified daily or when necessary, according to the work that had been completed or needed to be completed. The author worked according to a modified version of software development framework called Scrum. The sprint board and status of the items assigned to this board were evaluated and planned during bi-weekly sprint planning meeting. To receive feedback and constructive criticism, the author participated in multiple sprint retrospective meetings that were held after the sprint planning meetings

The task list and the scheduling chart of the project phases were updated according to the issues, tasks and sub-tasks agreed upon the sprint planning or steering meetings. The author presented the changes in the next steering meeting.

3 Project management

To maintain up-to-date documentation, project plan and communication. The project manager:

- Used two hours bi-weekly for reviewing the project plan.
- Presented updated documentation and project plan
- Iterated, refined or re-planned based on feedback received or observed
- Handled and maintained the communication between thesis advisor, author, and commissioning party

Tracking time used for work and to bring transparency to the work the author:

- Presented status of the work during daily sprint meetings
- Adhered to the requirements, guidelines and acceptance criteria set by the commissioning party

For project plan presentation, steering meetings, and final project presentation, the project manager sent relevant material and invites to the meetings with notice of three or more (3+) business days.

The acceptance decisions of the created implementation were decided and reviewed inside of the author’s team at the commissioning company.

Any changes or amendments to project management procedures, materials, project plan, the project itself or agreed upon terms were visited in steering meetings.

In steering meetings and presentation meetings, the project manager wrote the meeting minutes. After the meetings, the project manager delivered meeting minutes within three (3) business days.

3.1 Work phases

First phase was approval phase. Where the author had to find a suitable epic, story, or an item from the commissioning company’s on-going or future projects that would match the approximate workload of the bachelor’s degree thesis. Once the topic was found, the topic was pitched to the author’s supervisor. After that approval was received in the pitch meeting, the author requested approval from the legal department, and the vice president of the department. For university’s approval, the author had to create a topic proposal for thesis coordinator and wait for the approval and get assigned a thesis advisor from the university.

Second phase was preplanning and planning. The author planned the preliminary schedule, content, and reviews.

Third phase was implementation, where the author conducted research and learning as required, developed, received feedback in form of code reviews, tested created solutions and snippets of code, and iterated on the work based on the feedback and reviews.

Fourth phase is the documentation for drafting the report, writing system design documentation and general reporting done at the company. This is a parallel work stage with implementation phase.

4 Research

4.1 Android for Developers

Android for Developers is the official website for Android app developers (Android Developer, 2022). It houses important topics for Android developers, such as Android Studio, which is the IDE used for development in this project, Android API reference, which is used as a source reference and a technical manual and Google Play which is Google's platform for distributing Android software.

In this project the usage of the Android API reference was high. It was followed to develop consistent and uniform software implementations. Referencing API literature made it easy to create additional functionality and make sure that the methods and classes written adhered to the basics of Android development.

4.1.1 Activity

An activity is a single user faced action. It is usually programmed to interact with the user as an UI. This means that there is a view inside of the activity window (Android Developer, Activity, 2022).

In activities lifecycles an initialization needs to be executed when drawing a view, interacting with widgets, or retrieving data to be displayed. To end an activity lifecycle, code could call finish(), which indicates that activity is done and should be terminated, the result of this operation is called ActivityResult which is then propagated back to the original activity that launched it via onActivityResult() (Android Developer, Activity, 2022).

The project uses activities to start applications, for example in this specific use case it is declared in project the AndroidManifest.xml file to give it permission to be launched from within code. In the code user can launch the application using the initialized Context, startActivity, and providing an intent as a parameter.

4.1.2 Context

Context is implemented and provided by the Android system. It is used to interface the global information in the application environment (Android Developer, Context, 2022). In the project we initialize an instance of context and pass that down from parent to child component. That way when we use specific initialized context to start activities, we can easily observe them and switch them to foreground or background while using another application, stop and kill the processes which run the activities.

4.1.3 Intent

Intent is an abstract description of the operation to be performed (Android Developer, Intent, 2022). In the project we use it multiple times, most relevant use case is to assign it package and class names, and additional flags. Project has a component that resolves activity information, where it dynamically resolves the package name and name of the activity, that the code requires to start. To create the activity, it needs to be initialized with new Intent. Setting the package name and activity name to the ClassName, which requires two String parameters. Package name implements the component name required, and the class name which is the name of the application package (Android Developer, Intent, 2022).

The flags used are FLAG_ACTIVITY_NEW_TASK and FLAG_ACTIVITY_REORDER_TO_FRONT. First one is used to make sure that there are no duplicate activities started and if there is a task running in this activity then it is brought to the front. Second flag is to prevent system from running and applying multiple activity transition animations (Android Developer, Intent, 2022).

4.1.4 Service

“A Service is an application component representing either an application's desire to perform a longer-running operation while not interacting with the user or to supply functionality for other applications to use.**”** – Android Developer, Service, 2022.

To have a functioning service class it needs to be declared in the package’s AndroidManifest.xml (Android Developer, Service, 2022). The project uses two different kinds of services, first one is the overlay service, which is discussed in the view chapter 5.1.6. Second service is NotificationListenerService, which is the next chapter 5.1.5.

4.1.5 NotificationListenerService

NotificationListenerService receives calls from the system when notifications are being posted or removed, or their ranking has changed (Android Developer, NotificationListenerService, 2022). To extend and use this project we bind it in the AndroidManifest.xml file. It requires the name of the service, the permission to bind the listener service, and an optional label.

In this project it is used to listen for notifications that are being posted to the Status bar, which is an interface element at the top of the screen on Android devices, which usually can have several icons. When notification has been posted the package name of the application that posted it can be retrieved and then compared to the application package name list of approved or allowed application or package names. This way there can be dynamic variables that can possibly support multiple applications and their notifications.

4.1.6 View

View is a user interface component (Android Developer, View, 2022). In the project it is created in an overlay service which creates a button that remains on top of the application that was launched from the component. The button can be tapped to return to the original application.

4.1.7 Instrumentation

Instrumentation is a class for implementing application-level instrumentation code. It is used to instantiate classes before any application code is executed (Android Developer, Instrumentation, 2022). For example, starting activities in the unit tests in the project required to instantiate the activity monitor before launching the activities. That way the unit test was reliable, repeatable, monitorable and mockable.

4.2 Writing unit tests

Unit tests are written to test a specific, class, method, behaviour, and functionality of the code. When writing unit tests the testable components are isolated. They should not be dependant on each other, or any other components. Unit test developer should do this by mocking everything that is possible. This also means that the component should be testable locally without requiring any network resources, any 3rd party software, or any hardware devices. If it requires other resources, it might be a system test or an integration test. After isolation is achieved it should also be independent and depend on any previous results or outcomes. To achieve this, it should have a setup phase, and if it is repeatable and has multiple different blocks in the same unit test a teardown should ne executed after each single block of tests (Hodges, 2019).

Procedure to write the actual unit test should be simple and streamlined. Unit test should test how the code behaves and see if it reveals any bugs in the implementation. The naming of the unit test should be precise and accurate. The name of the test should be descriptive as possible and take as many characters as required to explain the use and classes and / or methods to be tested in the test.

Testing exceptions is important. The unit tests should be written to ensure that exceptions will get raised when the exception is supposed to happen, this procedure is crucial for debugging the code when bugs or issues get raised (Hodges, 2019).

It is important to write a unit test for new components, and it is important to change existing unit tests to match the changed implementation. They should also be run as a set every time there are changes in the source code. That way you can see any bugs, conflicts, or misbehaviour early in the development, and address them appropriately.

5 Methodologies and tools

The author used modern software development frameworks and methodologies that are regularly used at the commissioning company.

- Development frameworks and methodologies
- Modified adapted Scrum
- Agile

Development tools, software, and languages such as:

- Integrated Development Environment (IDE)
    - Android Studio (AS), Microsoft Visual Code (VSCode)
- Terminal
    - Git Bash, PowerShell, Command Prompt
- Physical devices
    - Mobile devices, laptop
- Programming languages
    - Java
- Version Control System (VCS)
    - Git, Gerrit

5.1 Agile

Agile is a project management and software development approach. It is iterative and accommodates teams to deliver value to customer. Basics of the approach is to deliver small amounts regularly, in increments. The requirements, plans and results are continuously being evaluated during and after the project to enable teams to have natural and logical mechanisms to respond to the changes quickly (Atlassian, 2022).

In the commissioning company Agile approach is the basis for the adapted Scrum framework that is utilized in software development and project management. It is used to get feedback in regular cycles and make sure that continuous improvements happen in the project management and software development. Agile is also used to estimate the work in sprints, as story points to measure what can be suitable to be completed in each development cycle.

5.2 Scrum

Scrum is a software development framework which helps teams solve and address complex problems while maintaining visibility and transparency of the work being conducted (Scrum.org, 2022).

In the commissioning company the way of working can be described as an adapted Scrum framework.

A Program Increment (PI) plan is planned together with development team, product management, such people include product owners and product managers, and program managers.

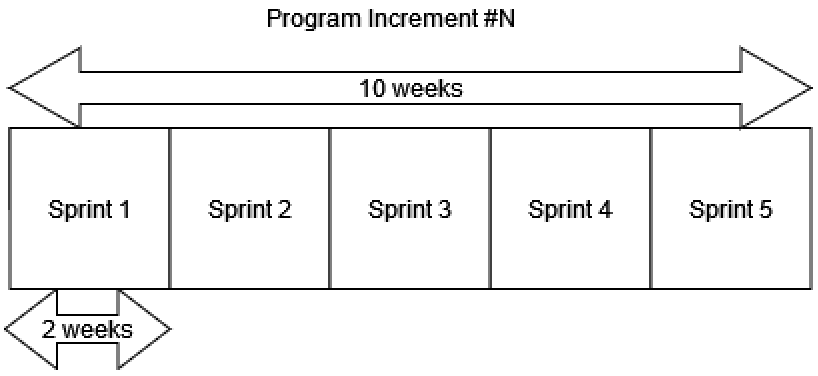

Figure 1 Program Increment

As displayed in Figure 1. Program Increment is 10 weeks, which consists of 5 sprints, and each sprint lasts 2 weeks. At the end of a previous PI next PI is planned on a PI-planning day, where the plan goes through drafts and is presented to the management and is available for modifications and suggestions. At the end of each sprint is a sprint planning meeting, where teams select development stories, and epics and add minimum viable product descriptions to the items included in the stories and epics. Then those items are planned for the next sprint.

After the planning a retrospective meeting is held. There the teams evaluate what went well, what did not go so well, where the work could be improved upon. It is also possible to add action points or goals for team members to complete during the sprints or until the next retrospective meeting. Retrospective meeting is also a good place to raise concerns about some issues that have affected the team, for example production issues, or dependency issues that are coming from outside of the team but affect the team negatively. It can also go to the other end of the spectrum and team members can give their appreciation for a particular member or a shoutout to a specific group in the team for a work well done.

On a daily level, a daily meeting is held, where team members present their work that they did previously and announce the work that is going to be tackled next. It is also a platform for the team members to discuss issues, problems or questions that affect or interest other members of the team or ask for help.

The cycle that was described is completed multiple times in a year. It is a description of the software development and project management of the work that is and was conducted at the employ of the commissioning party.

5.3 Integrated Development Environment

Main integrated development environment used in this project was Android Studio. Which is based on IntelliJ IDEA, it is also the official IDE for Android app development (Android Developer, Studio, 2022). In this project the author wrote the code and unit tests using Android Studio. It was also used for debugging.

Secondary IDE used in this project was Visual Studio Code. It was used to inspect specific files, such as test files, and objects, such as JSON-files.

5.4 Terminals

Terminals used in this project where Git Bash, which was used to execute git version handling commands. PowerShell, and Command Prompt, were used to install, manage, and modify existing software and tools.

5.5 Hardware

Hardware required for this project were a work laptop. Two mobile devices, which had different Android versions and device platforms.

5.6 Version Control System

VCS used were git and Gerrit.
“Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency”- (Chacon, Scott, 2022). Git is used in this project by the company to manage and coordinate repositories, develop projects, and keep track of file changes.

Gerrit is code review tool to execute peer reviews of the committed code in project files. One or more peers give review for the code, and it must receive a score of at least +3 to pass. There are patch sets, which means that original commit has changed, and changes

has been posted, which answer and address the comments or changes required in the code review. The scoring is that one or more peers who are not the author of the code or an automated system, can give the commit different scores which mean:
-1, it needs work and will not be accepted like this and requires major or minor changes.
0, any comment and will not affect the final score.
+1, peer reviewer is not sure about everything and partly approves the commit. Or it could be an automated system which has compiled the project and ran initial unit, integration, and system tests which it has passed, but still requires +2 from peers.
+2, score means that the commit will be approved as-is to the repository, or the necessary requirements have been met and is now ready to be committed to the repository. Or it could mean that the commitment is simple and does not require in-depth review.

6 Implementation

The implementation process started when author received the .apk-file from the 3rd party vendor, which is called product B. The author needed to enable application switching between commissioning party’s product A, and product B. Since product B was a standalone application, it was able to automatically run side-by-side with the product author is developing.

To be able to launch the product B from author’s application author needed to add permissions to AndroidManifest.xml. This is done by listing activities in the AndroidManifest.xml with their package name in it wrapped in manifest-application-activity and including an intent filter. Intent filters are used with implicit intents, so the Android operating systems finds correct components with the matching intent (Android Developer, Intent, 2022).

After wrapping it in AndroidManifest.xml it can be launched inside of product A. Since product A is multiplatform product, author needed to add logic to check which platform is currently in use and use that to determine how to launch the application. Relevant way of launching activities in this project was the startActivity().

Product A has an established component that has capability to call for specific actions to be executed. This meant that the author had to create a new variable where the call to startActivity() component could be used. Then the new component that launches the product B with startActivity() could use that to switch the product A to the background and launch or switch to product B.

To ensure that users could return to product A, author re-used an existing solution that creates an overlay service view with a button (Figure 2 below), which can be tapped to return from product B to product A.

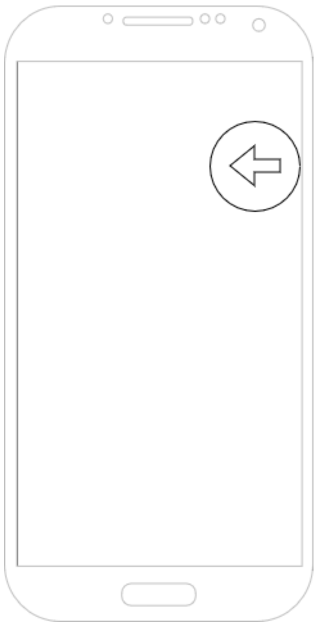

Figure 2 Mock-up of Overlay service view button

Project required a component that would listen to the notifications on the device. To make it easier to formulate and develop, the author found a project in GitHub repository (Pereira, 2021) with the service called NotificationListenerService (details about the service and repository in Chapter 5.1.5). The existing repository was a good place to start thinking about how to implement the service. It was used to formulate how application package names could be used to see what applications would be allowed to send notification to the Android system. The example is decently reliable, since it has been forked 67 times at the time of writing this report, and by author's logic, since it was not forked for this project author could assume that other professionals would not fork it either, and it might be mainly used as a learning opportunity, or as a hobby project purposes.

The service is used to catch the notification when it is posted to the Android system. In the code component retrieves package's name from the Android package manager. Once the information is retrieved it is compared to the allowed list of application package names. This way it can be dynamically modified to add more applications or remove existing ones. After the approved package name is resolved, it is used to resolve the application name in the android package manger. Once resolved it is passed down to the notification manager that is called via initializing the application context and application instance and passing the application name to it. Then the application name is compared to different types of alarms and programmed logic selects the new notification type that was created for this purpose. It is a notification type that includes localized titles and messages.

Displaying and posting notifications to Android operating system from product A required permissions to be added to AndroidManifest.xml-file. In addition, device had to have special app permissions allowed in the notification access.

To have the product B fully operational along the side product A, user must log in once to product B. After that existing logic maintained will manage the background and foreground application state of product A.

7 Product

7.1 Code review in practise

The code review was held twice since the software components were split into two (2) stories. The author implemented required changes, improvements and suggestions that were received from the code reviews. The feedback gotten from the code reviews made it possible for the author to meet all possible guidelines and criteria regarding code conventions and mandates required in the commissioning party’s internal documentation.
After passing the code reviews, integration tests and the automated unit tests software components were successfully merged to the master code repository and an application was built.

7.2 Testing the software components

To make sure that software components work properly and there were no bugs, the application went through scrum testing, which is type of testing where the created software components are tested via manual, automated or hybrid tests. After passing the scrum tests, software components and the product the components were implemented into is designated to an official product, that has the release date and official versioning of the product included. To be accepted to production the product created must pass validation testing, which means an independent team is testing the product with the documentation and plans created by the releasing team, in this case the thesis author’s scrum team.

7.3 Components

Product A is a software application for mobile devices. It is developed for Android operating system. It can be used to collect eCOA. Users fill out specific information using tools provided in the application, such as electronic diaries or video appointments.

The functionality for commissioning party’s product A has the functionality to launch 3rd party vendor’s application, product B from the user interface of the application. User can choose to launch product B via button inside from product A, blue button figure 3, mock 1.

Application switching is initiated, and product B is now in the foreground, and product A is placed in the background. After that the overlay service view button is created. User can either use product B normally as they would like or exit back to product A via the button in figure 3, mock 2.

After the button was tapped the product B is now in the background and product A is switched back to foreground.

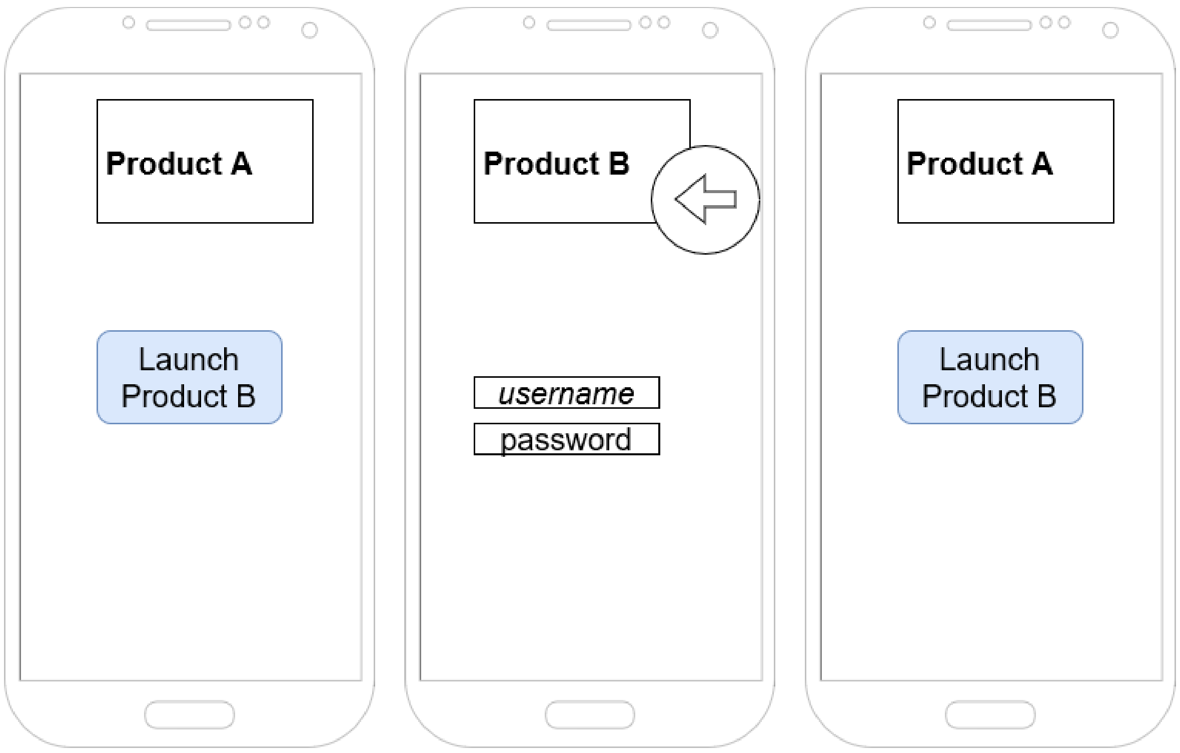

Figure 3 Visual presentation of application state (mocks 1-3 from left to right)

As for the notification listener functionality, product A catches notifications of product B, figure 4, mock 1. If device is asleep, like in figure 4, mock 2, product A will vibrate and make a sound. Parallelly to waking up the device and catching the notification, the component created in this project will create localized alert, figure 4, mock 3, in product A to remind the user to check notifications from product B.

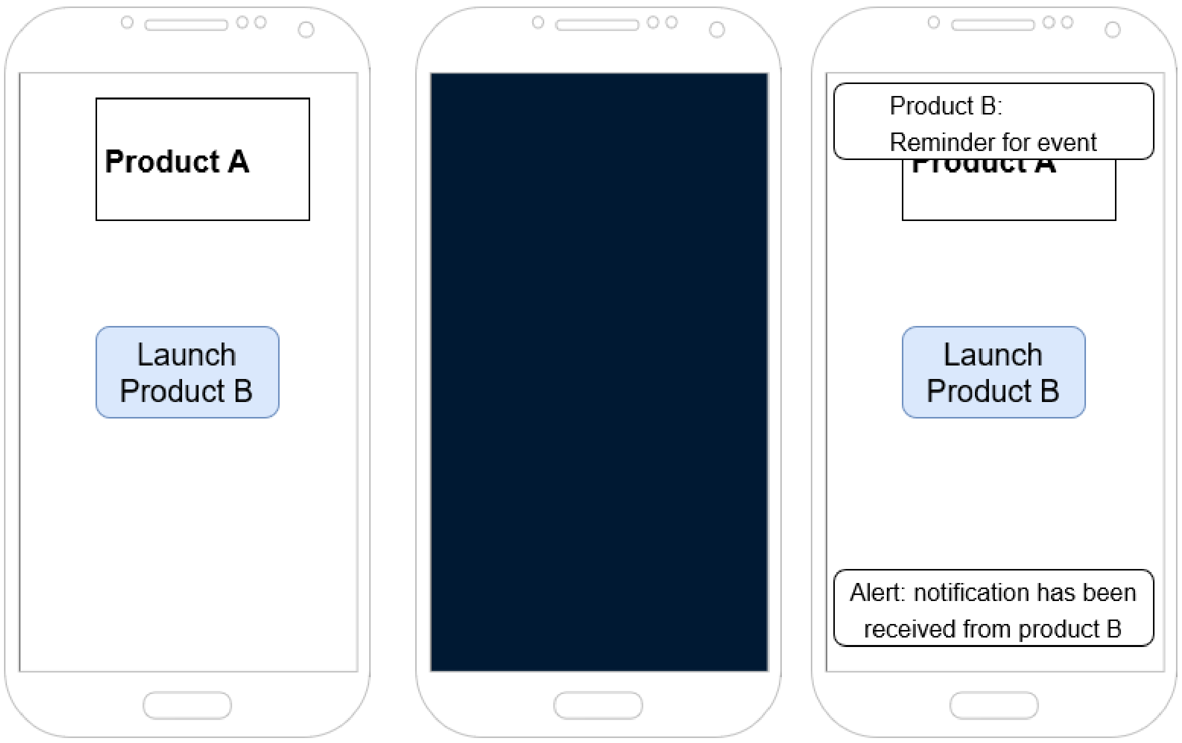

Figure 4 Notification received from product B (mocks 1-3 from left to right)

8 Conclusions

The result of the project reflects upon the entire project and the product, that was the software components integrated to the existing product. The adapted Scrum framework was used fully, and the author adhered to the decisions and goals set in the daily meetings, bi-weekly planning meetings and the bi-weekly retrospective meetings. This resulted to the software components were finished on time, fulfilled the functional requirements described by the product management, adhered to the specifications in the MVP, and were documented in the system design documentation.

Research used in the project was mainly technical manual called Android API (Application Programming Interface) reference (Android Developer, Reference, 2022). It provided almost all the necessary examples, conventions and ways to use established methods and classes to write and develop modern Java code.

The integrated development environment (IDE) Android Studio (AS) was sufficient and is at the time of writing this report the official IDE used in the commissioning party to develop this product. It helped to format the code in the components to be syntactically correct.

For the repository and version control tool Git was used, and it is the official tool for version control management of the code base in the commissioning party. It was mainly used from the Git bash Shell terminal. The command prompt was only used to run automatic unit tests. For the author the management of the product code was difficult at start. The author had to merge other commits that were bug fixes to his project, to ensure that there would not be any merge conflicts at the end code review.

The implementation of the software components went well. Author managed to create an implementation that had more functionality than described in the MVP. All the components implemented worked as expected. When reflecting upon the functional requirements, author could assume that product could help the commissioning party sell studies using this product, since it now supports 3rd party vendor’s application. As a criticism author could describe the short time allowed to sufficiently plan the project, and could not take part in the initial design, system specification and functional requirement meeting.

The product created in this project was tested and met the functional requirements of the story that were set by product management and described by senior developers, developers, author of the report and product owner. The software components that were created were functional, passed unit tests and scrum testing. At the time of writing, first story that had the first component was validation tested and is now released to production. The second story is pending scrum testing and will get designated a release version and release date and is going to get validated.

To ensure the integrity and functionality of the product A after integrating the new software components to its source code. Author’s automated and manual unit tests were executed, and they passed successfully.

The personal relevance for author was impactful. Project was the first steppingstone to software development career and provided the author an opportunity to prove the acquired skills and competencies he had received from his previous position as a junior test automation engineer. This thesis project marked the transition in the author’s career from testing organization to the development organization.

The project taught the author how to work as software developer in a modern work environment. The technical competencies he deepened was Java development, unit testing, writing system design specification documentation and summarising the progress made in daily and biweekly meetings as part of a scrum team. He also learned how to work on stories that were part of the MVP, to compartmentalize and divide the stories into tasks and subtasks and how create efficient solutions to produce functioning software components.

Glossary of terms

apk	Android Package, file format used by Android operating system that is compiled from any data and resource files (Android Developer, Guides, Fundamentals, 2022)
AS	Android Studio
eCOA	electronic Clinical Outcome Assessment
IDE	Integrated Development Environment
MVP	Minimum Viable Product
Product A	Commissioning party’s application
Product B	3rd party vendor application

References

Android Developer, Android Studio, Intro. 2022. URL: https://developer.android.com/studio/intro. Accessed 24 January 2022.

Android Developer, App, Activity, 2022. URL: https://developer.android.com/reference/android/app/Activity. Accessed 24 January 2022.

Android Developer, App, Instrumentation, 2022. URL: https://developer.android.com/reference/android/app/Instrumentation. Accessed 24 January 2022.

Android Developer, App, 2022. Service. URL: https://developer.android.com/reference/android/app/Service. Accessed 24 January 2022.

Android Developer, Content, Context, 2022. URL: https://developer.android.com/reference/android/content/Context. Accessed 24 January 2022.

Android Developer, Content, Intent, 2022. URL: https://developer.android.com/reference/android/content/Intent. Accessed 24 January 2022.

Android Developer, Guides, Fundamentals, 2022. URL: https://developer.android.com/guide/components/fundamentals. Accessed 1 February 2022.

Android Developer, Notification, NotificationListenerService, 2022. URL: https://developer.android.com/reference/android/service/notification/NotificationListenerService. Accessed 24 January 2022.

Android Developer, Reference, Android API reference, 2022. URL: https://developer.android.com/reference. Accessed 02 February 2022.

Android Developer, View, View, 2022. URL: https://developer.android.com/reference/android/view/View?hl=en. Accessed 24 January 2022.

Atlassian, Atlassian, Agile, 2022. URL: https://www.atlassian.com/agile. Accessed 25 January 2022.

Chagall, Fábio Pereira, Notification Listener Service Example - GitHub, 2021. URL: https://github.com/Chagall/notification-listener-service-example. Accessed December 2021.

Gerrit Code Review, About, 2022. URL: https://www.gerritcodereview.com/about.html. Accessed 24 January 2022.

Gerrit Code Review, Index, 2022. URL: https://www.gerritcodereview.com/index.html. Accessed 24 January 2022.

Git, Git, 2022. URL. https://git-scm.com/. Accessed 24 January 2022.

Git, Site, 2022. URL: https://git-scm.com/site. Accessed 24 January 2022.

Hodges, N. October 7, 2019. 13 Tips for Writing Useful Unit Tests, betterprogramming.pub. URL: https://betterprogramming.pub/13-tips-for-writing-useful-unit-tests-ca20706b5368. Accessed 24 January 2022.

Pereira, F. 2021. Notification Listener Service Example, GitHub. URL: https://github.com/Chagall/notification-listener-service-example. Accessed December 2021.

Scrum, Scrum events, 2022. URL: https://www.scrum.org/resources/what-is-scrum. Accessed 25 January 2022.

Appendix A