Bachelor's Thesis

Information and Communications Technology

2022

Yolanda Theodorakis

Modern Mobile Application Development

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Information and Communications Technology

2022 | 30 pages

Yolanda Theodorakis

Modern Mobile Application Development

Developing mobile applications can be challenging, especially with so many ways to build the application and different technologies existing.

The objective of this thesis was to introduce one option for developing a full-stack cross-platform mobile application using React Native, Express, and MongoDB. The objective was achieved through a case study of an application that makes it possible for women to send their location information quickly via text messages to pre-saved phone numbers whenever they do not feel safe. This type of mobile application is especially significant in Finland because there do not exist any viable mobile apps regarding women’s safety that are available in Finnish.

As a result, a first prototype of the application was created. The user can create credentials and use them to log in, save and delete contacts to and from the app, and send their location data through text messages to multiple contacts simultaneously. The functions worked as anticipated, and the used technologies can be an excellent choice for a modern mobile application. However, it is critical to assess what the requirements for the application being developed are and whether this particular solution is right for it.

Keywords:

mobile application development, software development, React Native, Express, MongoDB

Opinnäytetyö (AMK) | Tiivistelmä

Turun ammattikorkeakoulu

Tieto- ja viestintätekniikka

2022 | 30 sivua

Yolanda Theodorakis

Moderni mobiilisovelluskehitys

Mobiilisovelluskehitys voi olla haastavaa, etenkin kun olemassa on monia eri teknologioita sekä tapoja rakentaa sovellus.

Tämän opinnäytetyön tavoite oli esitellä yksi tapa kehittää full-stack, alustariippumaton mobiilisovellus käyttäen React Nativea, Express-viitekehystä sekä MongoDB:tä. Päämäärä saavutettiin kehittämällä tapaustutkimussovellus, joka mahdollistaa naisille tavan lähettää sijaintitiedot nopeasti tekstiviestillä ennalta määritettyihin puhelinnumeroihin, jos he eivät tunne oloaan turvalliseksi. Tämän kaltainen mobiilisovellus on erityisesti merkittävä Suomessa, sillä varteenotettavia naisten turvallisuuteen liittyviä sovelluksia ei ole saatavilla suomen kielellä.

Lopputuloksena luotiin sovelluksen ensimmäinen prototyyppi. Käyttäjä voi luoda käyttäjätunnukset ja kirjautua niillä sisään, tallentaa yhteystietoja sovellukseen sekä poistaa niitä ja lähettää sijaintitiedot tekstiviestillä useille yhteystiedoille samanaikaisesti. Toiminnot toimivat odotetusti ja käytetyt teknologiat voivat olla erinomainen valinta modernille mobiilisovellukselle. On kuitenkin tärkeää määrittää kehitettävän sovelluksen vaatimukset ja arvioida onko juuri tämä ratkaisu sopiva sille.

Asiasanat:

mobiilisovelluskehitys, ohjelmistokehitys, React Native, Express, MongoDB

Content

Pictures

Tables

List of abbreviations

CLI	Command Line Interface
DBMS	Database Management System
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JS	JavaScript programming language
JSON	JavaScript Object Notation
JSX	JavaScript XML
NoSQL	Not only SQL
OS	Operating System
PC	Personal Computer
PWA	Progressive Web Application
QR Code	Quick Response Code
SQL	Structured Query Language
UI	User Interface
UX	User Experience
VCS	Version Control System
VS Code	Visual Studio Code
WCAG	Web Content Accessibility Guidelines
XML	Extensible Markup Language

1 Introduction

This thesis concerns the development tools that are needed for creating a modern, full-stack mobile application. It will do so by going through a case study of a mobile application, from idea to development.

The thesis will begin with a more thorough description of the purpose of the project and outline and clarify the contents of the case study. After that the thesis will go through the planned features and functionalities as well as user interface and user experience design for the application in the planning section.

The thesis will then go through a set of tools and technologies chosen for the development process of this particular mobile application in the development section. Each of the technologies and their key features are introduced and, in addition, some viable alternative options are proposed if such alternatives exist.

The results section will then evaluate to which extent the planned features and functionalities were executed and whether they work as anticipated. The developer will also provide a commentary on what could have been done differently. The next steps for further development are also outlined.

2 The purpose of the project

This section outlines the purpose of the product, ergo the reason that the project was undertaken and the problem it hopes to solve.

As a young woman living in Finland, the author of this thesis has learned from a very young age to be cautious, when being alone out and about, especially at nighttime. The world is full of cases about women leaving, for example, their friends' place and never arriving home on time. Instead, they are raped, kidnapped, or murdered by men. A recent case that has stuck with the author is the murder of Sarah Everard [1]. If a woman has never experienced violence herself, she certainly knows someone who has.

To make women feel safer an idea of a women's safety application occurred. This type of app is especially significant in Finland because there do not exist any viable mobile apps regarding women's safety that are available in Finnish. The project's goal is to develop a mobile application that allows its users to save phone numbers to the app and send preconfigured text messages containing the user's current location data to all the numbers simultaneously. Text messages were chosen for the means of communication because calling someone in some situations might not be an option. To send the text messages as quickly as possible, maximum of two button presses are allowed to make the procedure the most efficient.

In addition to the practical reasons for building this application, there were also technical reasons. Building a mobile application can be overwhelming and challenging, especially with so many ways to build the application and different technologies existing. The purpose of this thesis is to go through a case study described above, and with that introduce one way of developing a full-stack cross-platform mobile application using React Native, Express, MongoDB, and various other tools and technologies. This thesis does not aim to define the absolute best way of developing modern mobile applications. Professional experience should always be used to identify the most appropriate tools and techniques for the project at hand.

3 Planning the project

3.1 Features and functionalities

The product owner/developer had a clear idea of what they wanted the finished product to do. These planned features were consolidated into a table. Using a table to break down intended project outcomes and features is flexible and extendable as the table's rows and columns can easily be edited. Such tables of requirements can also often be used as checklists during the project, which was the case in this project. Table 1, below, will be repurposed in the results section when evaluating the project's success.

Table 1. The functional requirements of the project.

Requirement	**Done?**
User can create new credentials.	
User can use credentials to log in.	
User can save contacts to the app.	
User can delete contacts from the app.	
User can send an SMS to all the saved contacts simultaneously.	
Sending an SMS requires maximum two button presses.	
The SMS contains user's location data.	
User can log out.	
User can navigate between the four different screens.	

3.2 Design and UX

As seen in the previous section, one of the planned features for the application is clean and simple user interface (UI). The app is intended to be used in potentially stressful and sensitive situations so the UI and user experience (UX) must make it easy and quick for the user to send for help. The user base of the application is

also going to be diverse, including people with disabilities. Because of that this project is being made following the Web Content Accessibility Guidelines (WCAG) 2.1 as closely as it is possible at this point in the development phase [2].

Due to the small scale of the project, it was decided that only simple wireframing is going to be made with Balsamiq Wireframes to act as a guideline for the developer. Below can be seen two pictures that present the wireframes for all four different screens that are going to be implemented.

Picture 1. Wireframes for Home screen and Contacts screen.

Picture 2. Wireframes for Log in screen and Create user screen.

The user interface was not going to have many visible features, so the designer/developer was comfortable with deciding the typography, color, and other design choices when the features were being developed. This, however, is not the best practice and is not recommended for bigger projects, as with more than one developer parts of the UI can become mismatched without specific UI design that all the developers can follow.

4 Development of the app

4.1 Development environment

This subsection introduces the smaller elements that are needed for the full development experience for this kind of a project. While the choosing of operating system (OS), integrated development environment (IDE), and terminal/shell might seem trivial things to consider, they have a great effect on the project flow, and thus are important things to choose wisely.

As the developer's primary development machine is a personal computer (PC) with Windows 11 Pro operating system, it was an obvious choice for this project. Windows 11 offers a snapping layout feature which allows multiple tabs to be snapped to place on the screen in different layouts based on screen size and resolution. This brings comfort to the development work as you can easily see multiple tabs at once and switch between them more efficiently. Windows 11 also runs Android apps natively without the need of an emulator.

The code editor used in this project was Visual Studio (VS) Code. Developed and released by Microsoft in 2015, this open-source code editor has rapidly become the most used developer environment tool among developers worldwide, with over 70% of 82 000 developers naming VS Code as their code editor of choice [3]. VS Code offers, among other things, syntax highlighting, intelligent code completion, debugging, and embedded Git version control system. It is also highly customizable with configurable themes, keyboard shortcuts, and various installable extensions.

The operating system's command prompt, or commonly known as cmd.exe or cmd, was used as a shell in this project. It is a command-line interpreter that supports a set of commands and can run programs. The cmd also has its own programming language for writing shell scripts.

4.2 Native vs. cross-platform development

When choosing a programming language for mobile applications the most important thing to determine is whether you want the app to run on Android or iOS devices. In most cases, however, the app needs to run on both platforms. This was also the case in this project.

For native iOS devices Apple has developed a programming language called Swift. Swift is open-source, very readable compared to its predecessor Objective-C, and it makes the programming experience interactive as the output can be seen while coding. The official programming language for Android apps is Kotlin developed by Google. Like Swift, Kotlin is also open-source and more readable than its predecessor Java. Building an application separately for both of the platforms has its perks. The code base and app ecosystem are usually more simple which results in faster performance and faster development. Faster performance is especially good for gaming apps and apps that have heavy animations. Native development also offers better user experience due to platform-customize UI/UX components, and easier bug and tech issue prevention. The development gets significantly slower when adding new features for both iOS and Android as two code bases need to be updated. The developer also needs to have knowledge of multiple technologies which might result in higher costs. [4]

As the mobile application in this particular project is fairly simple and lightweight, the developer chose to develop a cross-platform application which runs on both iOS and Android with just one programming language and code base. This makes the programming and future updating faster and easier, as the developer has to master only one programming language. The programming language chosen for this project was JavaScript (JS). According to Stack Overflow JavaScript is world's number one programming language [3]. Its main task is to bring functionality and interactivity to web pages, but as it is a multi-paradigm scripting language, it is exceptionally versatile and is used in both client- and server-side

as well as in non-browser environments [5]. Nowadays JavaScript is also used in embedded software [6].

Mobile apps done in plain vanilla JavaScript are called progressive web apps. They are browser-based, yet they have the same look and feel as native apps. PWAs however perform weaker on iOS and are not available in app stores. Because of this, making use of JavaScript's third-party libraries and frameworks was chosen for this project. The next two chapters introduce these frameworks. [4]

4.3 Front-end framework: React Native

4.3.1 Key concepts of React Native

React Native is an open-source front-end framework developed and released by Meta Platforms Inc. in 2015. React Native runs on React, which might sound more familiar as it is widely used in modern web applications and has become a popular UI framework in the software industry [7].

React Native uses JSX syntax which stands for JavaScript Extensible Markup Language (XML) which allows for writing elements inside JavaScript. For example, using curly braces makes it possible to embed any JavaScript expression, like variable, constant, or function call inside a <Text> element.

```
1   import React from 'react';
2   import {Text} from 'react-native';
3
4   const Introduction = () => {
5     const name = 'Yolanda';
6
7     return (
8       <Text>Hello, my name is {name}!</Text>
9     );
10  }
11
12  export default Introduction;
```

Picture 3. Constant embedded into a <Text> element.

JSX separates concerns loosely into components that contain both the rendering logic and other UI logic. Components can be nested inside other components to create new components that are independent and can be reused. This component architecture is appealing because it provides for reusable components that may be imported anywhere in an application while preserving all of their functionality. However, depending on where a component is presented on the app, it is possible that it will need to act differently. It might be that a button needs to have different text in different places, for example. This is where the

concept of “props” comes to help. Props are React Native’s so called “creation parameters” that can pass data down to a child element. In Picture 4 below, a button component receives the text it displays as a prop. This allows for the reuse of the button component across an application, with a different text depending on its intended purpose. As stated previously with the example in Picture 3, JSX also makes it possible for any valid JavaScript expression to be a prop. [7]

```
1   import React from 'react';
2   import {Button} from 'react-native';
3
4   /* ButtonComponent.js */
5   const ButtonComponent = (text) => {
6     return (
7       <Button title={text} />
8     );
9   }
10
11  /* App.js */
12  const App = () => {
13    return (
14      <ButtonComponent text={'Learn More'} />
15    );
16  }
```

Picture 4. A button component receives its title via prop called “text”.

Sometimes you might want to save the information passed down in props. This is done using “state”. The state allows components to change their output over time in response to, for example, user interactions, whereas props are read-only and should not be altered. The state is initialized every time the application renders. [8]

Expo

Expo is an open-source platform for developing native applications for Android, iOS, and the web. It comes with a universal runtime and libraries that allow you to write React and JavaScript to create native apps [9].

This project was created using Expo command-line interface (CLI). Expo offers a local development server which was used in this project [10]. When the development server is run, Expo generates a quick response (QR) code, which then can be scanned with the Expo Go mobile application and the project will run on the device. The Expo Go app was used to test and run this project on an Android phone.

4.3.2 Alternatives for React Native

Although React Native is perhaps the most used tool for cross-platform mobile applications, some less known but viable options still exist. As recently as in 2018, Google launched Flutter. Flutter is developed with Dart programming language, and it offers fast development, easy debugging, and low costs. Its userbase is rapidly growing but it still lacks an extensive resource base, and it does not have the community that builds third-party libraries and packages like React Native has. [11]

Another option for React Native is Xamarin which uses C#, developed by Microsoft in 2011. Xamarin offers also fast development, flexibility, and scalability. However, like Flutter, Xamarin, too, has a limited community. It is also not considered an ideal option for graphics-heavy apps. [11]

Further information about React Native

More comprehensive information about React Native can be found on their website and GitHub repository linked below.

Website: https://reactnative.dev/

GitHub: https://github.com/facebook/react-native

4.4 Back-end framework: Express.js

4.4.1 Key concepts of Express.js

JavaScript's main purpose has always been to bring functionality into browser. This changed, however, in 2009 with the creation of Node.js. Node.js is a platform that allows JavaScript to run environments as a server. This is significant, because it enables the development of full-stack applications using only one programming language. [12]

Typical server functionalities include, for example, routing, handling requests, serving static and dynamic content, and connecting to databases. Node.js has several frameworks for assisting in creating those functionalities. The most used of these frameworks is Express.js. Express.js provides speed to the development process with its methods and modules, and a stable structure on which to build the application, especially when an application is built from scratch, which was the case in this project. [12]

4.4.2 Alternatives for Express.js

As mentioned, Node.js has several frameworks. One good alternative for it is Koa.js which is made by the same developers who made Express.js. Koa.js offers a focus on a library of methods that are not offered in Express.js. Another viable option is Total.js, which is built on the core Hypertext Transfer Protocol (HTTP) module and is known for its high-performance request and response handling. [12]

Further information about Express.js

More comprehensive information about Express.js can be found on their website and GitHub repository linked below.

Website: https://expressjs.com

GitHub: https://github.com/expressjs/express

4.5 Database: MongoDB

4.5.1 Key concepts of MongoDB

MongoDB is an open-source database management system (DBMS) developed by MongoDB Inc. in 2009. MongoDB is not a Structured Query Language (SQL) database management system, it is usually referred to as a Not only SQL (NoSQL) database. NoSQL databases store data in a different format and do not use relational algebra the same way as relational databases do. [13]

MongoDB stores data in documents, meaning that all the data for a given document tends to be in a single document, not spread across many tables like in relational databases. MongoDB documents store data in field and value pairs and follow JavaScript Object Notation (JSON) format. MongoDB allows documents to be nested to express hierarchical relationships and to store structures such as arrays or objects. These structures correspond to native data types in many programming languages such as JavaScript or Python. The documents are stored in collections, which are analogous to tables in relational databases. Collections are highly flexible as they do not enforce one set schema for all documents in the same collection, rather each document can have their own schema. [14]

When creating a database in MongoDB, at least two copies of the data are created automatically. This group of three or more MongoDB instances are called a replica set. A replica set replicates data continuously between the instances, offering automatic failover and data redundancy. [14]

Although MongoDB can be installed and used locally, MongoDB Atlas was chosen for this project. MongoDB Atlas is essentially MongoDB in browser and cloud-hosted on Amazon Web Services, Microsoft Azure, and Google Cloud Platform. MongoDB Atlas paired with a graphical user interface (GUI) MongoDB Compass the developer found the development experience to be smooth and interactive.

4.5.2 Alternatives for MongoDB

There are many viable alternatives, although not as popular, for MongoDB if one wants to make use of document-based DBMSs. Some considerable choices are, for example, CouchDB and RavenDB. Out of the three, MongoDB offers support to significantly more programming languages, and also makes use of predefined data types.

Further information about MongoDB

More comprehensive information about MongoDB can be found on their website and GitHub repository linked below.

Website: https://www.mongodb.com/

GitHub: https://github.com/mongodb/mongo

4.6 Version control: Git

4.6.1 Key concepts of Git

Git is an open-source version control system (VCS) authored by Linus Torvalds in 2005. The purpose of a VCS is to maintain a history of every version of the code base and to allow the developers to work simultaneously without the fear of overwriting each other's changes. [15]

The feature which makes Git stand out from other version control systems is its branching model. A branch is essentially a completely separate clone containing the code of the branch that it is created from. Branching is highly effective. It makes switching between branches seamless. Different branches might have different roles, one for production, one for testing new features, and several others for general work, for example. The ease of branch creation also makes feature-based workflow possible. Each new feature can get its own branch which allows for experimentation and quick disposing of the branch if something does not work out. [16]

There is a significant amount of graphical user interfaces available for Git, for example, GitHub Desktop, SourceTree, and Git Extensions to name a few. GUI can make working with Git smoother and quicker as you can see all the repositories, branches, and changes to code with a glance and without the need to type any commands. GitHub Desktop was the preferred choice for this project.

4.6.2 Alternatives for Git

Git is considered superior VCS among developers, there really is not a good challenger for it. Other version control systems of course exist, for example, Azure DevOps Server, Mercurial, and AWS CodeCommit, but the author of this thesis would not recommend using them.

Further information about Git

More comprehensive information about Git can be found on their website and GitHub repository linked below.

Website: https://git-scm.com/

GitHub: https://github.com/git/git

5 Results and discussion

The functional requirements were defined previously as follows below in Table 2. It also demonstrates to which extent the requirements were delivered.

Table 2. The extent to which functional requirements were delivered.

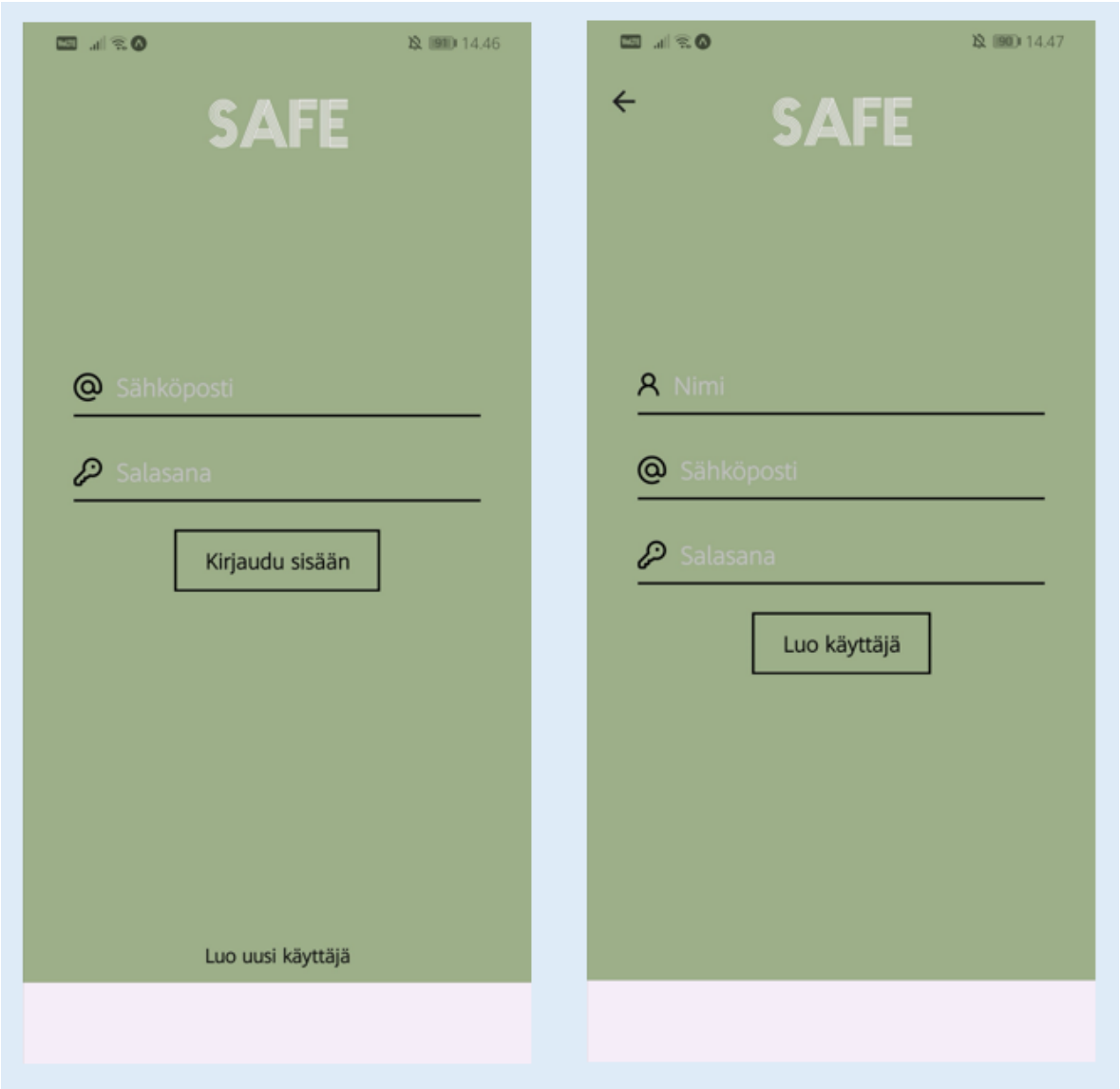
Requirement	**Done?**
User can create new credentials.	x
User can use credentials to log in.	x
User can save contacts to the app.	x
User can delete contacts from the app.	x
User can send an SMS to all the saved contacts simultaneously.	x
Sending an SMS requires maximum two button presses.	x
The SMS contains user's location data.	x
User can log out.	x
User can navigate between the four different screens.	x

As can be seen from the table's "Done?" column, all of the requirements of the projects were successfully met. The logic for creating user credentials and logging in/out was executed in the back-end and other functionalities in the front-end part of the application. This was an intentional decision, because the developer was not very familiar with using Express.js and MongoDB with React Native. Reflecting back on the decision, the sending of the text messages could have also been executed through the back-end, because although the current solution works as intended, with two button presses (one to open the device's text messaging app and one to send the text), the user interaction would be faster and more efficient with only one button press. The developer believes that this could be achieved with moving the functionality from the front-end to the back-end.

The saving and deleting contacts functionality was not connected to MongoDB. Instead, it was achieved with a React Native community package Async Storage, which makes saving data locally to the device easy [17]. Async Storage has been tried and tested in multiple previous projects that the developer has contributed on. The user's location data was handled as coordinates in the text messages. This is not optimal as very few can straight up read coordinates and know what place they mean.

The app's navigation between the four screens was executed smoothly. The UI was simple, clear, and functional as can be seen in Picture 5 and Picture 6 below.

Picture 5. The user interfaces of Home screen and Contacts screen.



Picture 6. The user interfaces of Log in screen and Create user screen.

Further development

This application could, in theory, be used in its current state. It would, however, need to be developed further for it to become as functional and well-known among women as it can be so that it would be considered a good tool for enhancing women's safety. The most crucial issue is the lack of permissions. The app needs to ask user for permission to use the text messaging app and the device's location. Without those permission the app will not be approved to the app stores. Another functionality issue is the location data which is displayed as

coordinates as stated previously. The coordinates need to be reverse geocoded, meaning that they need to be converted into actual understandable addresses.

Now, in addition to the changes mentioned above, the application would also benefit from a few more features. These features could include but not be limited to, for example, the ability to start recording video and voice with a button press, showing safe houses in Finland on a map, and an anonymous discussion feed where users could discuss with each other about their experiences and, for example, public places where they have faced misogyny, harassment, or violence. In addition to a Finnish version of the application, an English version and a Swedish version would also be good to consider for increasing accessibility and userbase of the app.

6 Conclusion

The technology options for building a modern full-stack mobile application are endless. The purpose of this thesis was to introduce one possible stack of technologies for building such application. This was achieved successfully through a case study application which was built with React Native, Express.js, MongoDB, and other various tools and technologies that make up the development environment. The technologies worked well together and there is no doubt why these particular technologies are a popular stack among mobile developers at the time of writing this thesis. However, it is critical to assess what the requirements for the application being developed are and whether this particular stack of technologies is right for it.

The case study app itself intended to provide a tool to make women in Finland feel safer, as there does not exist an application dedicated to women's safety that would also be available in Finnish. In comparison to the planned features and functionalities, the app worked as anticipated and it is a good first prototype, but it would need further development for it to become functional enough for the Finnish mobile app market and for the intended userbase.

References

[1] Morton, B. 2021. Sarah Everard: How Wayne Couzens planned her murder. Online: https://www.bbc.com/news/uk-58746108 [Accessed 12.05.22]

[2] W3C 2018. Web Content Accessibility Guidelines (WCAG) 2.1 Online: https://www.w3.org/TR/WCAG21/ [Accessed 24.05.22]

[3] Stack Overflow 2021. Developer Survey. Online: https://insights.stackoverflow.com/survey/2021 [Accessed 29.03.22]

[4] Webiotic 2021. Best Language for Mobile App Development - What You Need to Know. Online: https://www.webiotic.com/best-language-for-mobile-app-development-what-you-need-to-know/ [Accessed 29.05.22]

[5] Mozilla 2021. About JavaScript. Online: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript [Accessed 31.03.22]

[6] W3Schools 2022. Node.js Raspberry Pi - GPIO Introduction. Online: https://www.w3schools.com/nodejs/nodejs_raspberrypi_gpio_intro.asp [Accessed 31.03.22]

[7] Meta Platforms, Inc. 2022. React Fundamentals. Online: https://reactnative.dev/docs/intro-react [Accessed 29.05.22]

[8] Meta Platforms, Inc. 2022. Learn the Basics. Online: https://reactnative.dev/docs/tutorial [Accessed 23.05.22]

[9] GitHub 2022. Expo README-file. Online: https://github.com/expo/expo [Accessed 23.05.22]

[10] Expo 2022. How Expo Works. Online: https://docs.expo.dev/guides/how-expo-works/ [Accessed 23.05.22]

[11] StackShare, Inc. 2022. Alternatives to React Native. Online: https://stackshare.io/react-native/alternatives [Accessed 23.05.22]

[12] Wexler, J. 2019. Get Programming with Node.js. Shelter Island: Manning Publications Co.

[13] MongoDB Inc. 2022. What is NoSQL? Online: https://www.mongodb.com/nosql-explained [Accessed: 29.05.22]

[14] MongoDB, Inc. 2022. MongoDB Basics. Online: https://www.mongodb.com/basics [Accessed 02.04.22]

[15] Chacon S. & Straub B. 2009. Pro Git. 2nd edition. New York: Apress.

[16] Git 2022. About. Online: https://git-scm.com/about/branching-and-merging [Accessed 01.04.22]

[17] React Native Community 2022. Async Storage. Online: https://react-native-async-storage.github.io/async-storage/ [Accessed 24.05.22]