POLITECNICO DI TORINO

MASTER's Degree in COMPUTER ENGINEERING

MASTER's Degree Thesis

FRONT END DEVELOPMENT AND UX DESIGN

Supervisors

Prof. Giovanni MALNATI

Dr. Waqar HASSAN (Company)

Candidate

Hamza RASHID

JULY 2024

Abstract

In Qatar, there are numerous educational institutes and universities, and finding information about these institutions can be hectic and time-consuming. Daleluk, an organization based in Qatar, addresses this problem faced by the student community by providing a unified and simplified solution. Daleluk is the first platform in Qatar dedicated to this task. The project objective is to develop an application that works seamlessly across all devices and screen sizes, including computers and mobile devices. The app is responsible for displaying data present in CSV files, which contain information about universities and institutes, in the simplest and most user-friendly manner. The UI should be responsive to ensure it displays correctly on devices with different display densities. In addition to development, the project requires analytical data to visualize the app's behavior across all platforms. Insights like Crashlytics reports and app performance metrics were also utilized. React Native addresses this requirement by providing a single codebase for multiple platforms. It offers a front-end framework to display the UI in a simplified and reusable way. Firebase Firestore is used for storing data for all the institutes. Google Analytics captures various analytical metrics of the application, such as demographics, the number of users, and the frequency of screen clicks. The results obtained from the analytical process reveal valuable insights. Specifically, it helps identify which screens users click on the most and which screens they stay on the longest. For example, the "University Fees" screen attracts significant user interest. Additionally, it has been observed that a large number of users from the US are trying to use the application. However, since the app is only available in Arabic, the possibility of supporting multiple languages is proposed to the company.

Keywords: Educational Institutions, Firebase, Front End Development, Google Analytics, HTML Data format, Java script, Qatar, Mobile Application, React Native, User Experience, UI

Acknowledgement

First and foremost, I would like to express my deepest gratitude to my advisor Prof Giovanni Malnati for their guidance, and invaluable insights throughout this research journey. Their expertise and encouragement have been instrumental in shaping this thesis.
A special thanks to my colleagues and friends at Politecnico di Torino for their support, and the stimulating discussions that have often sparked new ideas and perspectives.
I would like to acknowledge Politecnico di Torino for providing the resources and environment conducive to my research.
To my family, my lovely wife, thank you for your unconditional love, patience, and understanding. Your constant support has been a source of strength and motivation throughout this endeavor.
Last but not least, I would like to extend my appreciation to my Daleluk supervisor Waqar Hassan for his support and assistance. Thank you all for your contributions and encouragement.

Contents

List of Figures

Chapter 1

Introduction

1.1 About the Company

Daleluk is an innovative company located in Qatar, devoted to connecting educational institutions with potential students worldwide. Daleluk is a new portal in the region that provides complete essential information about universities and educational institutes. This portal functions as a comprehensive solution for academic guidance, allowing students to enroll in higher education. This application offers an easy and streamlined way of academic guidance to cater to the difficulties faced by students in Qatar. Daleluk enables appropriate access to educational materials by joining all related information in one centralized platform, allowing students to make the best decisions regarding their education. The application is dedicated to giving comprehensive assistance to students during their academic journey, offering in-depth data on university programs, application procedures, and admission criteria.

Daleluk's motive is to rationalize the academic knowledge for students, by efficiently addressing their needs. Organizations are struggling to expand the educational atmosphere in Qatar by offering proper solutions and committed assistance, to make higher education more reasonable and easier to handle for every student.

1.2 Requirement of the project

The requirement involves creating an application that functions well on different smart devices and screen sizes, such as PCs, laptops, tablets, and mobile phones. The key function of this application is to show data in a straightforward and user-friendly manner from CSV files, which consist of extensive info about universities and educational institutions. The user interface (UI) should be designed fully responsive to provide an ideal visual experience for all users.

To accomplish this, the application must include contemporary web development methodologies, including the use of responsive design strategies through CSS media queries, flexible grid layouts, and scalable vector graphics (SVGs). This ensures that the content is easily readable and the interface is easily navigated, regardless of whether the user is opening the application from a computer, laptop, tablet, or mobile phone. The design should give the utmost position to legibility, easy navigation, and automatic interaction, ensuring that users can easily find and use the desired information.

In addition to creating a UI that is both responsive and user-friendly, the application must also include powerful analytics abilities to track and display user action on all platforms. This involves the gathering of data about user interactions with the application, the most often-used features, and the general level of user engagement. By the analysis of this data, developers can get enough information about user preferences and areas of discontent, enabling them to make ongoing improvements to the app's functionality and user experience.

Apart from development there should be some analytical data which is used to visualize the behavior of the app across all the platforms. Other insights like Crashlytics report and Performance of the app were also required. The application should provide features for reporting performance indicators and detecting problems using services like Google Analytics, Firebase Analytics, or comparable systems. Also, one can obtain comprehensive data on user activity, session duration, and sources of traffic. Moreover, the integration of Crashlytics will allow the team to actively track and analyze crashes as they occur, providing valuable data on the app's stability and promptly highlighting any urgent problems that require quick action.

Collectively, the development of this informative application involves developing a responsive, user-friendly interface for displaying university and institute data from CSV files, ensuring compatibility across all smart devices and screen sizes. Simultaneously, integrating analytical tools and performance monitoring systems will provide essential insights into user behavior, app stability, and performance, and enable continuous enhancement and a best user experience.

1.3 Overview and scope of the thesis

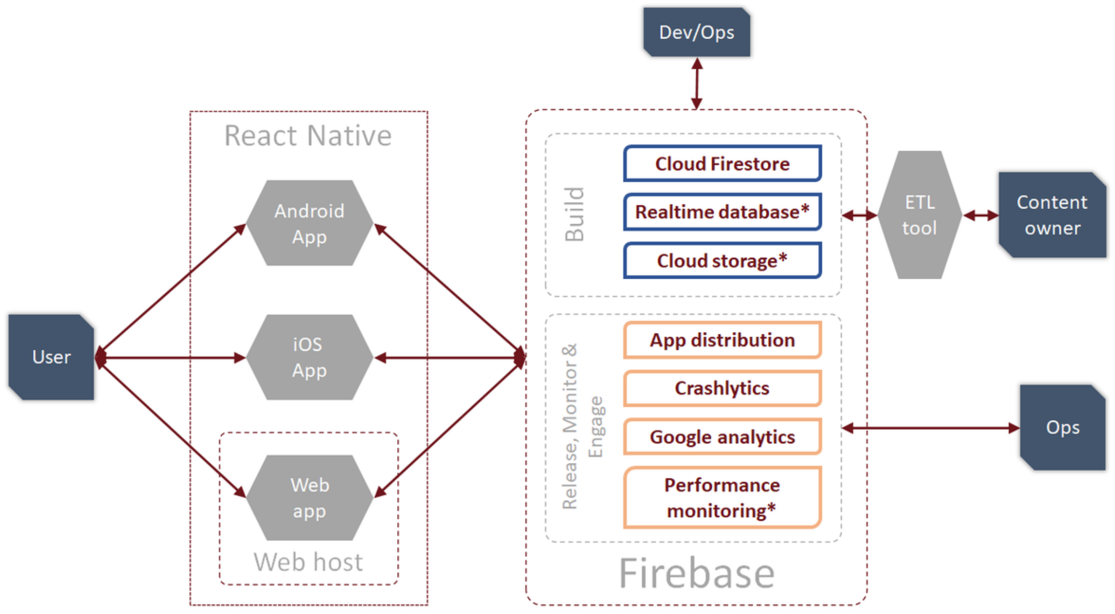


Figure 1.1: High Level Block Diagram

The figure above shows the complete structure of the application. So basically, like any other app/web development, there are 2 sides. Front-end development and back-end development. Back-end development will not be the scope of this thesis, but we will briefly describe the structure of back-end development. The main focus of this thesis will be front-end development and monitoring tools, which we will discuss in detail in the coming chapters.

Front-end development utilizes a unified codebase that is implemented on three different platforms: Android, iOS, and web. The codebase is constructed using React Native, a flexible framework that enables developers to rapidly build applications that can run on several platforms. Users will directly engage with the user interface (UI) developed with React Native, guaranteeing a uniform experience on all devices. The application uses Firebase Firestore, a scalable and versatile NoSQL cloud database, to store the data. Firestore facilitates real-time synchronization and offline functionality, hence improving the overall user experience by guaranteeing data availability and consistency across several platforms. To obtain a deeper understanding of user behavior and the functionality of the application, Google analytics have been used into the app:

1.4 Google Analytics

This is a tool utilized for the collection and examination of user interaction data, which offers significant insights into how users interact with the application. Metrics such as the number of active users, the duration of each session, and the demographic information of users assist in comprehending the audience of the application and their behavioral tendencies.

Another element of the application design is the ETL (Extract, Transform, Load) tool. This tool is responsible for transforming data from CSV files into the database schema utilized by Firestore. The ETL process encompasses the extraction of data from CSV files, followed by its transformation into a structured format that conforms to the database schema, and ultimately loading it into Firestore. This guarantees the precise and efficient integration of the data into the program. However this tool is not in the scope of the project. The subsequent chapters will address the distribution of the Android application. However, the distribution and hosting of the iOS application and online version will not be included in this thesis. Nevertheless, the principles and techniques described for Android can be equally employed for iOS and web platforms.

Overall, the application architecture consists of a cohesive front-end created using React Native, while Firebase Firestore handles the management of data storage. Monitoring tools like Google Analytics, Firebase Performance Monitoring, and Firebase Crashlytics are combined to offer extensive analysis of user behavior, app performance, and stability. The ETL (Extract, Transform, Load) tool is essential for the conversion and importation of data from CSV (Comma-Separated Values) files into the database. Subsequent chapters will explore the intricacies of front-end programming, the integration of monitoring tools, and the delivery procedure for the Android application.

1.5 Challenges

This project encompasses several significant challenges, each of which requires careful consideration and innovative solutions. Addressing these challenges effectively is crucial for the development of a robust and user-friendly application. The techniques employed to address these challenges will be discussed in detail in the subsequent chapters. The key challenges include:

1.5.1 Responsive Application

In the contemporary era, nearly every individual own a smartphone or electronic gadget that exhibits diverse screen sizes and densities. The application must possess the capability to operate flawlessly on a wide range of platforms, including mobile phones, computers, tablets, and other similar gadgets. It is essential to have a responsive design that can adjust to various screen sizes and resolutions to provide a consistent user experience on all platforms. Developing a responsive application necessitates the utilization of a diverse range of web technologies and methodologies. Media queries in CSS enable the program to adjust the layout according to the screen size of the device. Flexible grid layouts and scalable vector graphics (SVGs) guarantee the preservation of visual elements' integrity across various resolutions. In addition, it is necessary to provide touch-friendly interfaces specifically tailored for mobile devices to guarantee seamless engagement. The primary objective is to develop a user-friendly and aesthetically pleasing interface that improves user interaction, irrespective of the device employed.

1.5.2 Data Presentation

It is crucial to present the data of universities and institutes in a streamlined and easily understandable manner. The program should be created with a user-friendly interface that presents information clearly and intuitively for easy comprehension and navigation. This entails arranging data logically and employing UI/UX principles to guarantee clarity and user-friendliness. To achieve effective data presentation, it is crucial to prioritize usability and accessibility. The data should be classified and presented hierarchically, facilitating users' ability to locate their desired information. Interactive features like search capabilities, filters, and sortable tables can improve the user experience by enabling users to easily find and access relevant information. Visual aids such as charts, graphs, and infographics can be utilized to communicate intricate facts in a more easily understandable style. Furthermore, it is crucial to ensure that the application complies with accessibility requirements in order to accommodate people with disabilities.

1.5.3 Multiple University Support

The main goal of the program is to present data from multiple universities and institutes. Nevertheless, several colleges may display their statistics in distinct formats. The program should possess sufficient adaptability to accommodate various data formats and provide a consistent and accurate presentation of all information, irrespective of its origin. To tackle this difficulty, the application architecture needs to incorporate a strong data processing layer that is capable of standardizing different data formats into a cohesive framework. The task at hand entails the creation of an ETL (Extract, Transform, Load) tool. This program will extract data from CSV files, convert it into a standardized format, and then load it into the database. This methodology guarantees the smooth integration and consistent presentation of data from different universities. Furthermore, it is essential to create a versatile database schema that can handle various data kinds and structures to uphold data integrity and consistency. the multiple university support.

1.5.4 Navigation

An essential part of app development is the implementation of efficient and user-friendly navigation. Users should have a seamless and instinctive experience navigating the application, effortlessly locating the desired information without any additional complications. Creating a navigation system that is both efficient and easy to use is crucial for improving the overall user experience. Efficient navigation design entails establishing a coherent and rational framework for the application. This entails creating a navigation bar that is uniform and coherent, using breadcrumbs to facilitate easy retracing of steps, and establishing a distinct visual hierarchy to assist users in navigating around the material. Interactive components like dropdown menus, sidebars, and tabs can effectively arrange information and enhance accessibility by reducing the need for excessive clicking. Furthermore, integrating user feedback and conducting usability testing during the design phase guarantees that the navigation system fulfills user expectations and requirements.

1.5.5 Performance and Optimizations

User satisfaction is directly influenced by the performance of an application. An effective and streamlined application is crucial for retaining users and ensuring a seamless experience. This task entails enhancing the performance of the application, minimizing load times, and guaranteeing efficient operation on all devices. Performance optimization encompasses various measures, such as using efficient coding techniques, reducing resource use, and utilizing caching systems. Lazy loading can be utilized to postpone the loading of non-essential resources, hence decreasing the time it takes for the initial loading process. By optimizing photos and other media assets, their loading speed can be increased while maintaining their quality. Moreover, the utilization of performance monitoring tools facilitates the detection of bottlenecks and areas that require enhancement. Consistently conducting performance testing and continuously optimizing are crucial for upholding a high-performing application. application optimized to have a great performance.

1.5.6 Analytics

The major objective of this thesis is to develop technologies capable of monitoring and documenting user behavior. Analytics are essential for comprehending user interactions with the application and pinpointing areas that can be enhanced. The program should have the capability to obtain comprehensive insights that will be beneficial for making future improvements to the user experience/user interface design. By using analytics solutions like Google Analytics and Firebase Analytics, you may obtain extensive data regarding user activities, session durations, and user demographics. These observations aid in the identification of popular attributes, user navigation patterns, and potential sources of user disengagement. Through the analysis of this data, developers can make well-informed decisions to improve the user experience and resolve any usability problems. Users can establish custom event tracking and conversion targets to quantify particular interactions and outcomes, gaining a more profound understanding of user behavior and app performance.

To tackle these difficulties, a comprehensive strategy is needed, which includes responsive design, effective data presentation, support for various data formats, user-friendly navigation, performance optimization, thorough analytics, performance monitoring, and reliable crash reporting. The upcoming chapters will explore the precise tactics and strategies employed to address these problems and guarantee the triumphant development and implementation of the application.

1.6 Motivation

The ever-growing number of educational institutions and the diverse range of programs they offer have made it increasingly challenging for students to access and understand all the necessary information to make informed decisions about their education. In Qatar, the rapid development of the education sector has created a need for a centralized platform that can simplify this process. Daleluk aims to address this gap by providing a comprehensive, user-friendly application that consolidates data from various universities and educational institutes. This platform not only facilitates easier access to information but also enhances the overall decision-making process for students, thereby supporting their educational and career aspirations.

1.7 Problem Statement

The process of gathering and understanding information about different universities and their programs is often cumbersome and time-consuming for students. The data is typically scattered across multiple sources, presented in various formats, and often lacks clarity. This fragmentation can lead to confusion and suboptimal decision-making. Additionally, the diversity of devices and screen sizes that users employ to access this information further complicates the user experience. Therefore, there is a pressing need for a unified, responsive application that can present university data in a clear, consistent, and accessible manner, while also providing insights into user behavior and app performance to continually improve the user experience.

1.8 Aims

The primary aim of this project is to develop a cross-platform application that simplifies the process of accessing and understanding information about universities and educational institutes. This application will use a single codebase for Android, iOS, and web platforms, ensuring a consistent user experience across all devices. The project also aims to integrate robust monitoring tools to gather analytical data, track app performance, and report crashes, enabling continuous improvement of the application.

1.9 Objectives

Below are the objectives of the project, which are set to provide a valuable tool for students in Qatar, simplifying their search for educational opportunities and enhancing their overall experience with a user-friendly, high-performance application:

- Create a single codebase using React Native that works seamlessly across Android, iOS, and web platforms, ensuring the application adapts to different screen sizes and densities.
- Design an intuitive user interface that presents university and institute data in a clear and accessible manner, allowing users to easily understand and navigate the information.
- Implement a flexible data processing system that can handle diverse data formats from various universities, ensuring consistent and accurate data presentation.
- Develop a user-friendly navigation system that allows users to efficiently find and access the information they need, enhancing the overall user experience.

- Ensure the application is optimized for performance, reducing load times and resource consumption to provide a smooth and responsive user experience.
- Integrate Google Analytics and Firebase Analytics to gather detailed insights into user behavior, helping to identify popular features and areas for improvement.
- Detail the process of distributing the Android application, ensuring it is accessible to the target audience.

Chapter 2

Related Work

Front End Application Development and User Experience Design are essential parts of creating a successful mobile application. A famous framework named “React Native” introduced by Facebook, has grown significantly in the last decade because of its extraordinary capacity capability to build cross-platform applications [1]. Author (Desmurs, 2023) [2] shares the knowledge about the popularity of React Native among the famous companies that have adopted React Native for their applications. Facebook, Instagram, Walmart, Bloomberg, Tesla, and Shopify are prominent cases. These companies have provided data about multiple benefits, like rapid development times, shared codebases, and enhanced performance. For example, Facebook recorded a twice increase by shifting to React Native, and Instagram emphasized its capacity to speed up feature releases for both iOS and Android [2]. React Native boasts a huge and active community of software developers. This community facet is crucial as it offers a support network where developers can share knowledge, ask questions, and solve problems collaboratively [3]. The open-source nature of React Native boosts constant improvement and novelty within the community. This background study will explore the connection of Front-End Application Development, User Experience Design, HTML Data format, Google Analytics, and React Native.

2.1 Front End Development

Front End Development includes creating the user interface and user experience of a website or mobile application. It focuses on the visual features of the application that users interact with. Front End Developers use technologies like HTML, CSS, and JavaScript to build responsive and interactive interfaces. For mobile applications, frameworks like React Native have enabled the easy process for Front-End Developers by allowing the developers to create code once and use it across multiple platforms [4].

2.2 User Interface and User Experience(UI and UX)

User Experience Design is the systematic approach to increasing user contentment by enhancing the ease of use, availability, and enjoyment experienced during the interaction between the user and the product. An effectively crafted user experience can greatly influence the success of a mobile application. Design concepts, such as easy-to-use navigation, adaptable layouts, and cohesive branding, are essential for ensuring a favorable user experience. In the realm of React Native development, User Experience Designers work closely with Front End Developers to ensure that the application fulfills the requirements and anticipations of its users [5].

2.3 HTML Data Format

HTML, short for HyperText Markup Language, is the fundamental language used in web development to organize and structure content on the Internet. The text describes various components such as titles, paragraphs, hyperlinks, and multimedia. HTML has undergone evolution and now includes HTML5, which provides additional components and APIs to facilitate the development of intricate online applications. Introducing semantic elements like as, and enhances accessibility and search engine optimization (SEO). The research conducted by Berners-Lee et al. in 1994 established the foundation for HTML, which has subsequently undergone continual enhancements to cater to the requirements of contemporary web development, such as responsive design and multimedia integration [6].

Data extraction from HTML-encoded tables has been a subject of ongoing research, highlighting the significance of transforming data formats that are easy for humans to understand into ones that can be read by machines for different purposes. From 2000 to 2018, the literature presents various proposals that try to tackle this topic, with each offering distinct methodology and approaches. Initial endeavors in this field were on creating rule-based algorithms and heuristics to detect and extract tabular data from online pages. These solutions usually entailed analyzing the HTML structure to identify tables and then implementing particular algorithms to identify headers, rows, and cells. Nevertheless, the diversity in HTML coding standards and the intricacy of nested tables frequently restricted the efficiency of these methods (Roldán et al., 2020) [7]. Comparative studies identify multiple unresolved difficulties in extracting data from HTML tables. For example, it is still challenging to manage intricate tables that have headers at multiple levels, cells that stretch across multiple columns or rows, and layouts that are not regular. Furthermore, the absence of agreement on standardized datasets and evaluation measures adds complexity to the comparison of experimental findings across various investigations. The presence of this inconsistency highlights the necessity of a cohesive benchmarking framework to propel the field forward.

The article "Extraction and integration information in HTML tables" [8] emphasizes that the process of extracting and integrating information from HTML tables is a complicated task. This is mostly because HTML tables are designed for presentation purposes and are not well-suited for database applications. This article suggests a new method to tackle this problem by capturing the semantic hierarchies of HTML tables and incorporating pertinent information. The process entails standardizing tables to identify attribute-value pairs, use eigenvalues to determine headings, and manually creating global concepts and schemas. It utilizes lexical semantic sets and contexts to assign labels to attributes and combine data, eliminating conflicts and non-deterministic issues when mapping source schemas to global schemas [8].

2.4 Google Analytics

Google Analytics, introduced in 2005, is an essential tool for digital marketers to monitor and analyze website traffic. It offers a detailed analysis of user activity using indicators like as page views, bounce rates, session duration, and conversion rates. The platform enables the establishment of objectives, monitoring of events, and generation of tailored reports, hence facilitating decision-making based on data. According to Clifton (2012) [9], it has been emphasized that it is useful in developing marketing strategies and increasing user engagement. Nevertheless, the utilization of Google Analytics has sparked apprehensions over privacy and data security, particularly about rules such as the General Data Protection Regulation (GDPR). The aforementioned considerations highlight the necessity of adhering to ethical data usage and complying with privacy legislation [10].

Clifton (2012) explores the process of establishing goals and events, as well as generating personalized reports, emphasizing the adaptability and comprehensiveness of GA in facilitating data-driven decision-making. A substantial amount of the book focuses on the pragmatic use of GA, encompassing case studies and optimal strategies for utilizing the instrument to amplify marketing return on investment. Clifton tackles prevalent difficulties such as privacy issues and adherence to legislation such as GDPR, offering advice on the ethical utilization of data. The book highlights the significance of GA in contemporary digital marketing tactics and its function in promoting user engagement and business expansion [9].

2.5 React Native

React Native is a JavaScript framework that enables developers to create mobile applications that can run on several platforms using a single codebase. Developers can utilize JavaScript to construct applications while preserving the inherent performance and functionalities. Developers may utilize React Native to construct high-performance applications that possess the appearance and functionality of native apps on both iOS and Android platforms. The framework offers a comprehensive range of components and APIs that simplify the development process and facilitate quick prototyping. React Native's quick reloading functionality enables developers to instantly view and incorporate real-time modifications in the application, facilitating iterative improvements in design and user experience (Eisenman, 2015) [11]. A prominent advantage of React Native is its ability to facilitate cross-platform development. Developers can utilize a unified codebase for iOS and Android platforms, resulting in a notable decrease in development time and expenses. Utilizing pre-existing components from the open-source library, together with the flexibility to reuse code, improves efficiency [12].

React Native, a framework developed by Facebook, has become a prominent tool for developing mobile applications. It enables developers to create applications using JavaScript while still keeping the performance and features of native applications. One of the most prominent advantages of this technology is its ability to facilitate cross-platform development, allowing developers to use a single codebase for both iOS and Android. This greatly decreases the time and expenses associated with development. Code reusability, which involves utilizing pre-developed components from open-source libraries, improves efficiency (Mandapuram, 2016) [13]. Moreover, React Native possesses a substantial and dynamic community, offering a support system for exchanging expertise and resolving issues collectively, which is essential for ongoing enhancement and creativity.

The cost-effectiveness of the framework is apparent, as it allows a single team to work on both platforms, hence reducing the requirement for separate development teams and resulting in substantial cost savings. React Native optimizes application performance and preserves the native user interface by leveraging the platform's view rendering techniques (Dekkati et al., 2019) [14]. The fact that large firms such as Facebook, Instagram, Walmart, Bloomberg, Tesla, and Shopify have adopted it emphasizes the numerous advantages of this technology, such as reduced development time and enhanced performance. Nevertheless, React Native does have several limitations, including regions that require enhancement and difficulties in debugging caused by the introduction of an extra layer [15].

React Native outperforms competing cross-platform frameworks by leveraging the native rendering APIs of the platform, leading to a more authentic user experience (Hitz et al., 2017) [16]. The future of React Native seems auspicious given its swift market acceptance and ongoing enhancements, indicating that it will continue to be a significant asset for developing cross-platform mobile applications. Based on its capacity to streamline development difficulties and the increasing backing from its community, the framework is expected to continue being a vital resource for cross-platform mobile application development [17].

2.6 Responsive Interfaces

Responsive Web Design (RWD) has undergone substantial development since it was first introduced by Ethan Marcotte in 2010. The key principles of RWD include media queries, fluid grids, and fluid pictures. According to MOHAMED (2015) [18], the fluid grid concept enables websites to effortlessly adjust to different screen sizes, resulting in an improved user experience on different devices. Conventional fixed grid layouts frequently necessitate significant customization to adapt to fluid grids, which hinders the acceptance of such layouts among web designers. In response to this issue, Mohamed suggests a cutting-edge algorithm called Liquidizer.js, which is built using the jQuery Framework. The study utilizes an experimental research approach and purposive sampling. It employs tools such as Matt Kersley's RWD Tool and Bersoft Image Measurement (BIM) tool for testing and validation. The usability evaluation is conducted using the USE questionnaire using SPSS. The findings indicate that Liquidizer.js surpasses BlockIt.js in terms of image quality, ensuring non-distorted images and improved responsiveness. This optimized method for responsive web design (RWD) provides a more streamlined and standardized solution that can be easily implemented with just one line of code. This simplifies the process and encourages wider usage, thereby enhancing web design processes [18].

2.7 Overview

This literature analysis has provided insight into the important relationship between front-end development, user experience design, data concerns, and framework selection in creating effective mobile applications. React Native has gained popularity due to its capability to utilize JavaScript for developing applications that can run on several platforms. However, achieving a smooth user experience necessitates a careful equilibrium between capitalizing on the framework's advantages and recognizing its constraints. React Native provides a compelling value proposition. The capability to create code once and distribute it on both iOS and Android platforms results in substantial time and cost reductions. Moreover, the framework features a diverse range of pre-existing elements and a dynamic community of developers, which promotes the quick creation of prototypes and encourages creativity. This combination enables developers to create high-performance applications that have a native feel on both main mobile operating systems.

2.8 Research Gaps

Despite React Native's advantages, the review has identified key areas demanding further exploration. One critical research gap lies in extracting data from complex HTML tables. Current methods struggle with intricate layouts and lack standardized evaluation metrics, hindering effective data collection and analysis. Addressing this gap is crucial for applications that rely heavily on structured data from websites. Another area requiring scrutiny is the tension between user privacy and the power of analytics tools like Google Analytics. While GA offers invaluable insights into user behavior, concerns remain regarding user data security, particularly in light of regulations like GDPR. Research in this domain should explore methods that promote ethical data usage and ensure compliance with evolving privacy laws.

Finally, even with its proven benefits, React Native has limitations. The additional layer of abstraction can introduce debugging challenges that necessitate further investigation. Exploring ways to streamline debugging processes within the React Native framework would significantly enhance developer productivity. By addressing these research gaps, developers can refine their use of React Native and related technologies. This, in turn, will lead to the creation of even more efficient, user-centric, and secure mobile applications. The future of mobile app development hinges on continuous improvement, and these identified research areas offer valuable starting points for further exploration.

Chapter 3

Design and Methodology

Before we dive into the front-end development, we have to briefly understand what is happening at the back-end side. So that we can better understand how the data is being displayed on the front-end side. What does it mean to understand the back-end side? It means that we have to understand how the data is obtained. How are we storing it? Where are we storing it? What is the database schema etc.?

3.1 Back-end side

To answer all the above questions, let's observe the figure below. This figure shows how the content that we want to display on the front end will be obtained and stored in the database.

Let's start with the Content Owner. The content Owner is someone who has the raw data of all the universities/institutes. This raw data is stored in CSV format. The format of all the CSV files is defined by the content owner and the back-end team. The Content Owner will upload this data to the ETL tool.

ETL tool is a python script that converts the CSV file format to the database schema. Content Owner will simply upload the data into this script and this script will convert this data according to the pre-defined database schema. We will see what the database schema is later in this chapter.

The database we are using for the application is Firebase Firestore. The data in the Firestore will be available through the ETL tool. Now this data is ready to be fetched by the application. The application will fetch the data when the user is online and it will simply store this data in the device. Once the user is offline it simply retrieves the stored data. When the user becomes online again it will sync the data again. By doing this we will attain offline persistence.

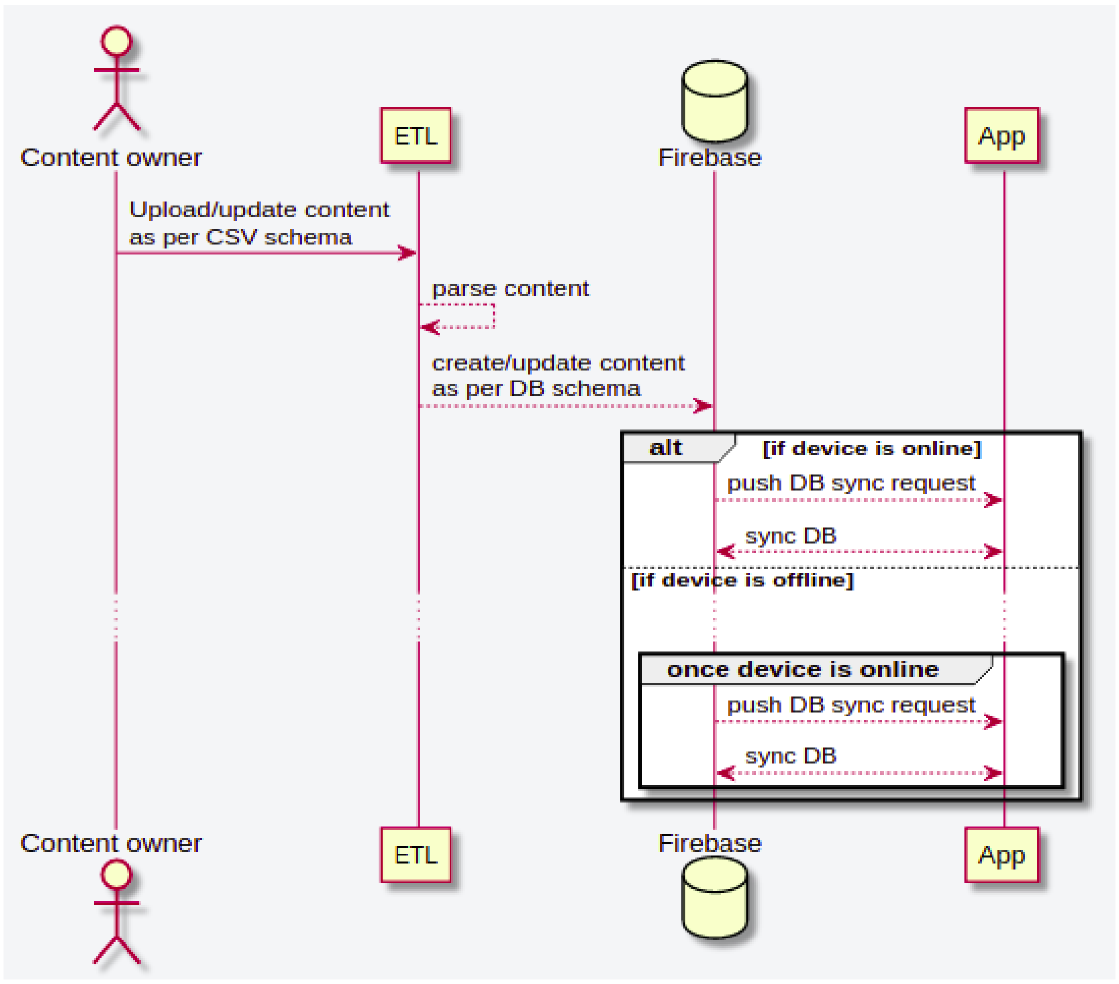
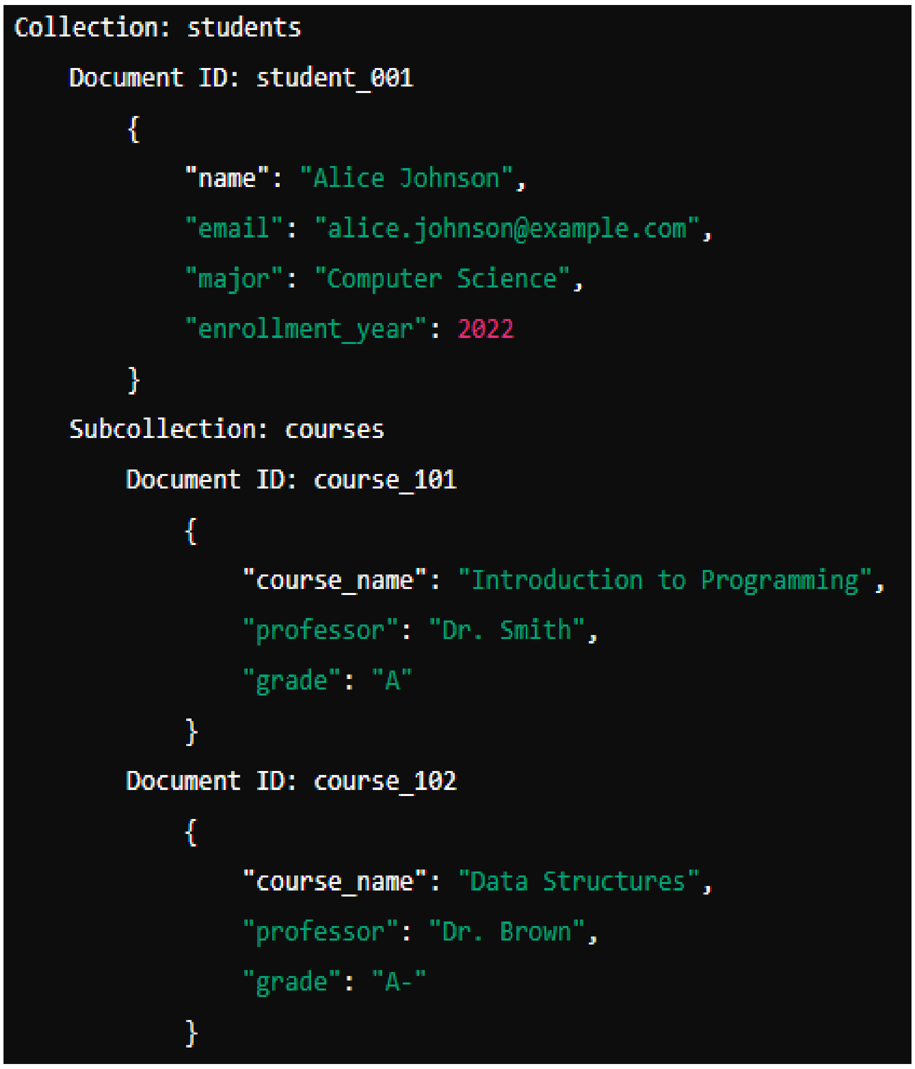


Figure 3.1: Content Flow Diagram

3.1.1 Database schema

Firebase Firestore also known as Cloud Firestore, is a scalable and flexible NoSQL database for mobile and web development from Google Firebase and Google Cloud Platform. It is designed to store, sync, and query data for your applications on a global scale. Firestore provides a powerful, efficient, easy-to-use, and integrated solution that allows developers to build rich, collaborative applications that enable secure and real-time data synchronization across multiple devices.

Firestore's data model is hierarchical. The data is stored in documents organized into collections. Each document contains key-value pairs. It can also be nested to create complex and structured data. Unlike traditional SQL databases, Firestore's schema-less design allows developers to evolve their applications without needing to manage and migrate schemas. This flexibility is particularly useful for rapid development and iteration, as it supports both real-time and offline use cases. It allows apps to remain responsive regardless of network latency or internet connectivity. e.g



```
Collection: students
    Document ID: student_001
        {
            "name": "Alice Johnson",
            "email": "alice.johnson@example.com",
            "major": "Computer Science",
            "enrollment_year": 2022
        }
    Subcollection: courses
        Document ID: course_101
            {
                "course_name": "Introduction to Programming",
                "professor": "Dr. Smith",
                "grade": "A"
            }
        Document ID: course_102
            {
                "course_name": "Data Structures",
                "professor": "Dr. Brown",
                "grade": "A-"
            }
```

Figure 3.2: Firebase Data Heirarchy

1. **Collection students:** As seen in the figure each student document contains key-value pairs for the student's basic information. Each student document

has a subcollection course containing documents for each course the student is enrolled in, including the course name, professor, and grade.

2. **Collection courses:** As seen in the figure each course document contains key-value pairs for the course's basic information just like the student document. Each course document has a subcollection of students containing documents for each student enrolled in the course, including the student's name and grade.

One of the key features of Firestore is its real-time synchronization capability. Changes to data are instantly synchronized with all clients connected to the database, enabling a seamless and responsive user experience. This is particularly useful for applications requiring real-time collaboration, such as chat apps, multiplayer games, and shared documents. So in our case, it is helpful too so that when a Content Owner adds or modifies something, the changes will be instantly visible to the users across all the devices. Additionally, Firestore supports robust querying and indexing, allowing developers to perform complex queries that are both fast and efficient, even as their data scales.

Security in Firestore is managed through Firebase Authentication and Firestore Security Rules, which provide a powerful and effective way to control access to data. These rules are written in a flexible, expression-based language that can enforce complex security policies. This ensures that data remains protected and compliant with privacy requirements. Firestore also integrates seamlessly with other Firebase services, such as Firebase Analytics, Cloud Functions, and Firebase Hosting, creating a cohesive ecosystem that simplifies the development process.

In conclusion, Firebase Firestore is a versatile and powerful database solution tailored for modern app development. Its real-time synchronization, flexible data model, robust querying capabilities, and strong security features make it an ideal choice for developers looking to build responsive, scalable, and secure applications. Whether you're developing a simple mobile app or a complex, multi-platform system as our application, Firestore provides the tools and infrastructure needed to support the project's growth and success.

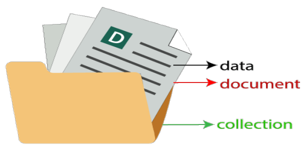


Figure 3.3: Firestore Data Model

The below figure shows a brief schema of the application's database. This figure does not show the actual data inside the schema, instead, it shows only the structure of our database. So there are multiple Collections and in each collection, we have documents that consist of the data (key-value pairs) and the Subcollections. In the Subcollections, there are documents, similar to Collections, which consist of data (key-value pairs).

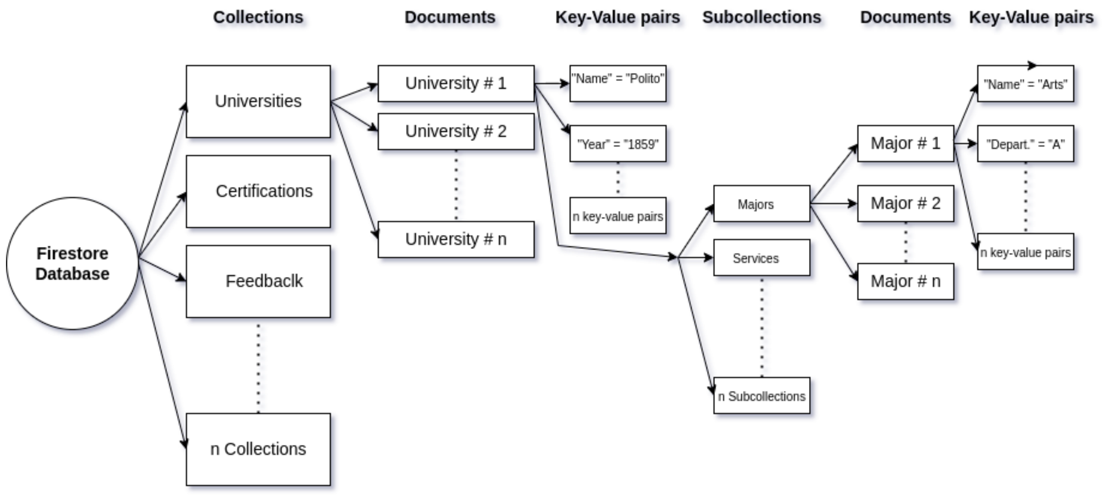


Figure 3.4: Database Schema

3.2 Front-End side

3.2.1 Why React Native

Before any development, one has to realize the project requirements, based on which we choose the best development environment, the best frameworks, and languages. In this section, we are going to discuss a bit about the different language and framework choices we had and why we chose React Native.

There were multiple options of languages/frameworks. Some of them are as follows:

1. Java/Kotlin for Android development.
2. Swift for IOS development.

3. React-JS for web application development.
4. React Native for native application development.

Cross Platform

React Native is capable of creating cross-platform apps i.e. Android and IOS. It creates apps using JavaScript as React is a framework built on it. So for any web developer, it is easier to learn and code in React Native instead of coding in JAVA/Kotlin/Swift.

Web Support

React Native for the Web is a compatibility layer between React DOM and React Native. It can be used in new and existing apps, web-only and multi-platform apps [19].

Single Code base

By choosing React Native. It gave us the ability to code in only one coding environment. All the native (Android and IOS) and web code be written in React naive framework. So at the end of the project, we will have a single code base that will be able to run on these 3 platforms.

Efficiency

Since our application does not require any complex APIs, in other words, we do not need to write the native code. So React Native will be a faster and more efficient approach for us.

Development and Maintenance cost

Since there is one code base, the development and maintenance cost is reduced because we don't have to write 3 codes for 3 different platforms.

Unified Development Language

React Native is built on JavaScript, a globally used programming language, and React, a popular library for building user interfaces. Many web developers are already familiar with JavaScript and React, which makes it easier for them to transition to mobile app development using React Native. This reduces the learning curve and accelerates the development process compared to learning platform-specific languages such as Java or Kotlin for Android and Swift for iOS.

Consistent User Experience

React Native ensures a consistent user experience across both Android and iOS platforms. And react native also supports web development bu using the libraries. The framework provides a set of common UI components that render natively on both platforms, maintaining the look and feel of a native application while sharing the same underlying code. This consistency is beneficial for users who switch between different devices and expect a uniform experience.

Code Reusability

React Native enables developers to write code once and deploy it across multiple platforms. The code is written using Javascript. This is achieved through the use of shared components and modules that work seamlessly on both Android and iOS. By reusing code, developers can significantly reduce development time and avoid the redundancy of writing and maintaining separate codebases for each platform.

Hot Reloading

React Native supports hot reloading. It is a feature that allows developers to see the results of their code changes in real time without restarting or refreshing the whole application. This feature enhances productivity by providing immediate feedback and reducing the time spent on testing and debugging. It also increases the efficiency of the development process.

Strong Documentation and Tutorials

The React Native documentation is comprehensive and well-maintained, providing clear guidelines and best practices for building applications. Additionally, there are numerous tutorials, courses, and community resources available that can help developers quickly get up to speed with React Native development. For developers having a strong community is a plus point and very helpful.

Growth

React Native is growing very fast. And there is currently no sign of it stopping. It's getting popular day by day. Facebook, Instagram, Skype, and many other famous apps are built on react native.

Third party plugins

React native allows us to easily implement third-party plugins and APIs within our application.

3.2.2 React Native Web Limitations

1. For layout, the classes and styling required for the web app will be quite different from the classes and styling required for the mobile app.

2. React-native-web does not have complete compatibility with React Native's JavaScript API. [20]

3. Web APIs cannot be directly leveraged in a React native project.

4. The web app pages have a different navigation flow than the mobile app in certain scenarios.

5. Although React Native functions well on mobile devices, intricate animations, and interactions may not exhibit the same level of smoothness on the web when compared to native web technologies. The abstraction layer can occasionally cause a decrease in the responsiveness of UI interactions due to the performance overhead it introduces.

6. React Native for Web may not achieve the same rendering efficiency as highly optimized web frameworks such as React DOM, particularly for apps that include frequent dynamic updates and intricate layouts.

7. React Native for Web aims to achieve extensive browser compatibility, while inconsistencies or problems may still arise in older or less frequently used browsers. To ensure complete compatibility, it may be necessary to conduct extra testing and utilize polyfills.

8. Certain native mobile APIs lack direct counterparts in online contexts. Functionalities that significantly depend on APIs specifically designed for mobile devices may require substantial modifications or may not be possible to implement in a web environment.

9. Not all React Native third-party libraries are compatible with the React Native for Web framework. Developers may be required to locate or generate substitute libraries that function flawlessly on the internet, a task that can extend the duration of the development process.

10. Modules that depend on native code, such as Java/Kotlin for Android or Swift/Objective-C for iOS, cannot be directly utilized in a web context. Due to this constraint, it is necessary to develop alternative implementations or completely distinct solutions to support the web.

11. React Native is specifically designed to prioritize touch interactions, which might vary greatly from the mouse-based interactions commonly found in online applications. Modifying the user experience to accommodate both touch and mouse interactions can provide difficulties.
12. Integrating React Native components into a web application might lead to a larger bundle size, which may have a negative influence on loading times and performance. Efficiently reducing the size of the build may necessitate meticulous configuration and implementation of code-splitting techniques.
13. React Native for Web applications may encounter difficulties with Search Engine Optimization (SEO), unlike conventional web applications. Implementing server-side rendering (SSR) using React Native for the Web can be more intricate than with standard React.

To deal with these limitations, we are going to have one repository for the native apps and the web app, but we'll either end up having some separate files for native and web anyway, or we'll have to check Platform.OS repeatedly to separate platform-specific code.

3.2.3 Expo

Expo is a framework for the React applications. It provides us with the tools and services through which we can easily develop, build, and deploy on IOS, Android, and Web apps from a single codebase.

Pros

Expo is free and open source which lets the developer write pure JavaScript code without writing any native code. All the native code is handed by the expo under the hood. So we do not have to worry about the native code. This makes the development fast. As there is no native code, it means the developer does not have to use Android Studio or Xcode for Android and IOS development respectively.

Expo also provides a feature called Publish Over The Air (OTA). With this feature, the developer does not have to upload every version of the application on the Play Store / App Store. Instead, they can publish their application on the expo servers very easily. This feature is great for testing the application.

With the expo, we can access native APIs for Android and IOS. Expo provides these native APIs out of the box which makes it easier for the developer to use the native features in the application.

Cons

Expo has a limited number of Android and IOS APIs. And apart from that out of the 102 modules available in the Expo SDK, 49 are not compatible with Web. Out of the 939 libraries listed on `https://reactnative.directory/`, 134 libraries support the web. These numbers are just to illustrate that React native tooling is not necessarily developed with the web in mind, and we'll most likely have to use React tooling either way. Expo adds another layer of abstraction, which means that he developer cannot directly access the native features. A lot of things are happening under the hood. Another limitation of the expo is the minimum supported versions. The minimum supported version for Android and IOS is 5+ and 10+ respectively.

But the good news is that we can always eject our application from the expo. Which means that we can always access the native code. In the coming sections, we will see that we have to eject the application because we are going to implement some features directly into the native code.

3.2.4 Navigation

For any web or mobile application, navigation is one of the most important aspects. It is the path that a user can follow to reach a certain goal. In any application, there could be multiple goals a user can achieve. For example, To place an order on Amazon. To do this user has to go through a certain path/screen to act. Home Screen -> Login Screen -> Search Screen -> Summary Screen -> Confirmation Screen. These are the screens a user has to go through to place an order on Amazon.

Similarly, in our application, there are multiple screens for multiple goals. For example, the user wants to submit the feedback or the user wants to book an appointment. To attain all these goals, we are using the React native navigation library. The complete navigation path of the entire application is given below.

As we can see there are two ways to navigate inside the app. The first one is the fab, which acts like a header in any web application. This fab will be replaced as a header and footer in the web application. So in the web app, all the navigation that we can do with the fab will be done using headers and footers.

From the FAB we can navigate to Home Screen, Our Services Screen, University Selection Screen, Major Selection Screen, Favorites Screen, and Feedback Screen

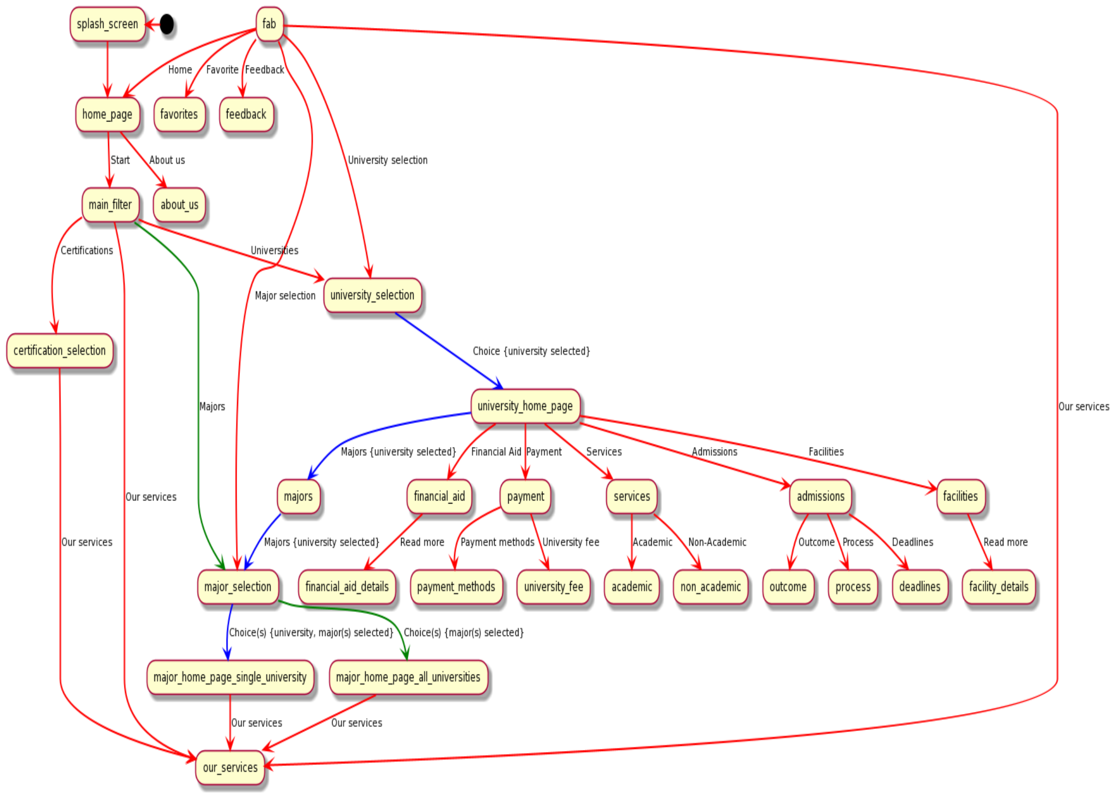


Figure 3.5: Navigation Flow of the Native Application

anywhere from the native application. But this isn't the case for the web application. As for the web, we are navigating through the header.

Apart from FAB navigation can be done using the normal buttons as shown in the figure. The buttons are represented as the words written along with the lines. For example, on the home page, we have 2 buttons named Main-Filter and About-Us. These buttons will navigate to these screens.

Navigating to Major Selection Screen

The purpose of the Major Selection Screen is to choose the majors along with the gender and the school name. For example, a user can select a major "History" based on the gender "MALE" and the school "ARTS". The screenshot of the Major Selection Screen is shown in the figure below. The green words in the figure are the translated text of the screen.

This screen has 2 major objectives:

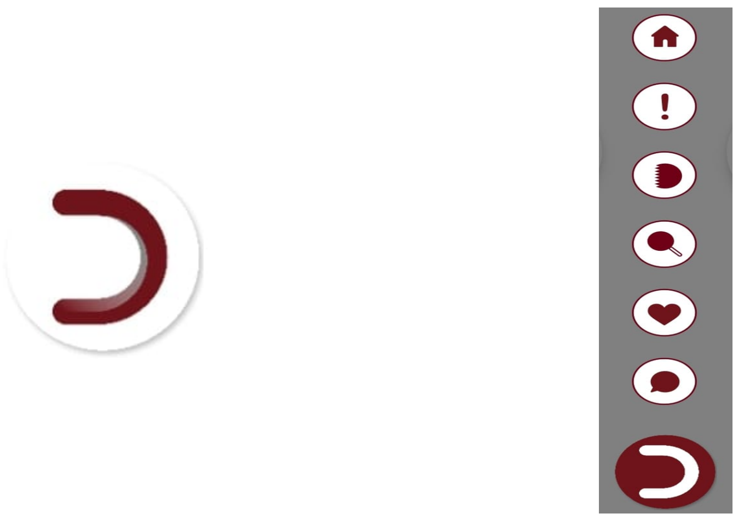

(a) FAB closed **(b)** FAB opened

Figure 3.6: FAB

1. Displaying majors for a single university
2. Displaying majors for all universities.

To accomplish these, there were 2 possible solutions. The first one is to have 2 separate screens. One screen displays the majors for a single university and another screen displays the majors for all the universities. The Second solution was to have a single screen. So the data will be displayed based on which screen the user navigates from. So if a user is navigating from the University Selection Screen after selecting a particular university and then navigates to the Major Selection Screen then it will display the majors of that selected university. If the the user is navigating from the FAB or Main Filter Screen it will display the majors of all the universities.

The second option was chosen to reduce the redundant screen. To implement this option we are passing the additional parameters along with the navigation prop. This is called the route parameters. The route parameters passed are University ID and University Name. Based on these 2 parameters, we are deciding if the university is selected or not. If it is selected then it means that the user is coming from the University Selection Screen. So we have to display the data of only that particular university. Otherwise, we will display the data of all the universities.

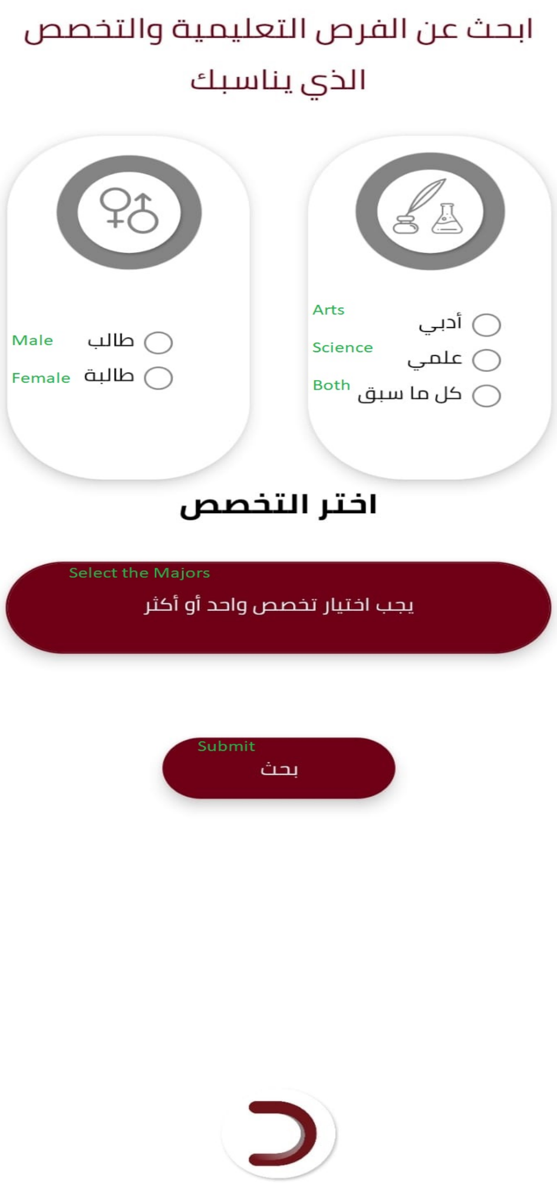
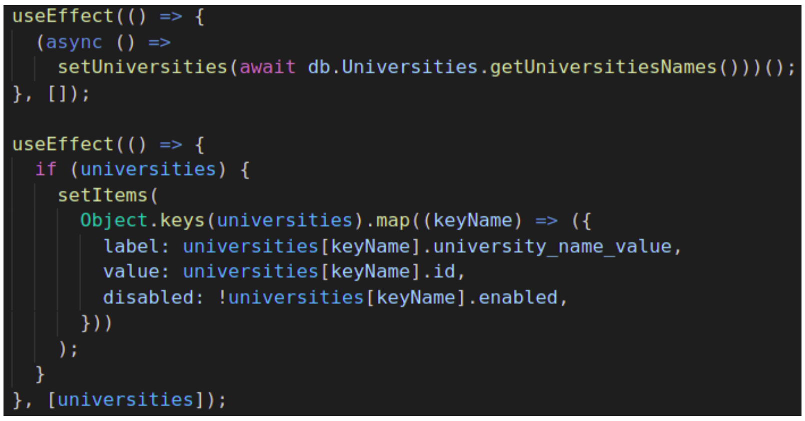


Figure 3.7: Major Selection Screen

3.2.5 Data Presentation

The main goal of the app is to display the data to the user. The data is the university data. In the previous sections, we have discussed how the is obtained. Initially, the Content Owner (CO) has the data in the CSV format. CO inputs this data into an ETL tool. the purpose of the ETL tool is to convert the CSV format to the database format. ETL tool is written in Python. After the data is converted into the database schema, it is ready to be imported into our React Native codebase. In React Native, we have created different classes and methods to call different types of data. For example, if we want to import the names of all the universities present in the database, we have to call a method called getUniversitiesNames(). This method will return all the names of the universities. And then we can map those names according to our needs. The figure below shows how we are getting all the universities by calling the method. And then how we are mapping those return values.

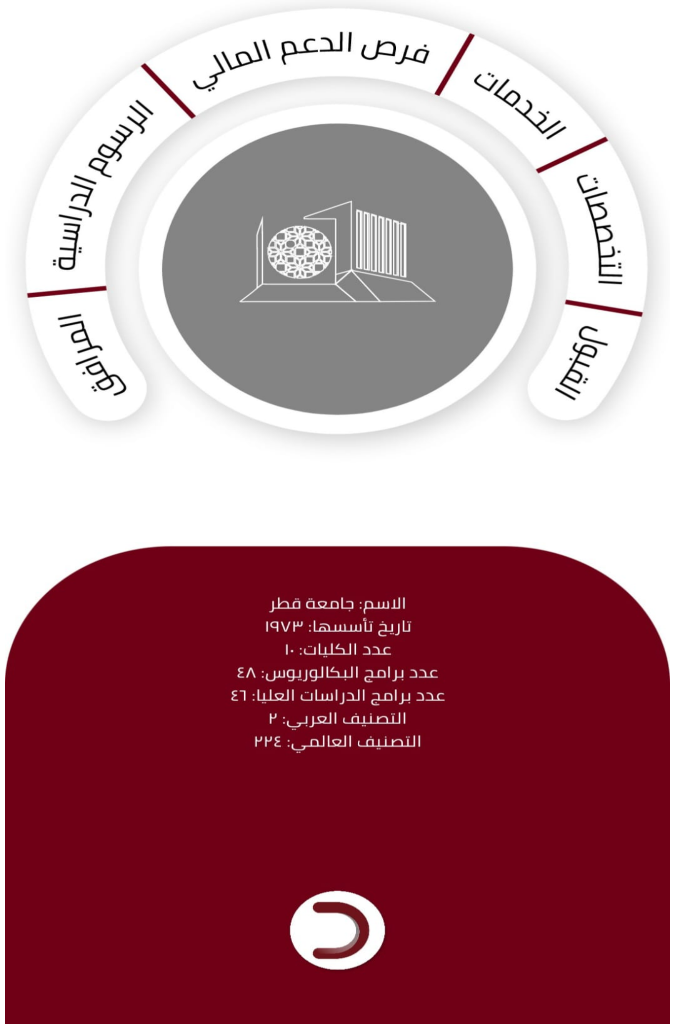


```
useEffect(() => {
  (async () =>
    setUniversities(await db.Universities.getUniversitiesNames()))();
}, []);

useEffect(() => {
  if (universities) {
    setItems(
      Object.keys(universities).map((keyName) => ({
        label: universities[keyName].university_name_value,
        value: universities[keyName].id,
        disabled: !universities[keyName].enabled,
      }))
    );
  }
}, [universities]);
```

Figure 3.8: getUniversitiesNames() Method

Each university's data is displayed on the University Home Screen. This is the screen where the user can see the university logo, university information (foundation year, name, etc), services that the selected university provides (sports, academic, non-academic), etc. From this screen, the user can navigate to the screen according to the selected service as shown in the navigation flow diagram. The University Home Screen is shown below.

As we can see there are two sections of this screen. In the top section, there is a circular navigator through which the user can click to see the information about the selected tab in the bottom section. From the bottom section, the user can navigate to the screens based on what he clicks on. The logo of the university is in



الاسم: جامعة قطر
تاريخ تأسسها: ١٩٧٣
عدد الكليات: ١٠
عدد برامج البكالوريوس: ٤٨
عدد برامج الدراسات العليا: ٤٦
التصنيف العربي: ٢
التصنيف العالمي: ٢٢٤

Figure 3.9: University Home Screen

the center of the navigator. This circular navigator is made up of SVG(Scalable Vector Graphics) instead of PNG or JPEG format. The reasons to use SVG are the following.

1. Scalability: SVG are screen resolution independent. It means that no matter what the size or the screen resolution is, the quality of the image won't be changed. Or we can say that the SVGs are independent of the screen's pixel density.

2. Performance: These are extremely beneficial for the web because it does not need to make an HTTP request to load an image. So they are fast as

compared to the PNG or JPEG file formats.

3. Customization: SVG images have a great styling option. You can easily customize your images using the properties fill color, stroke color, sizing, and more through CSS.

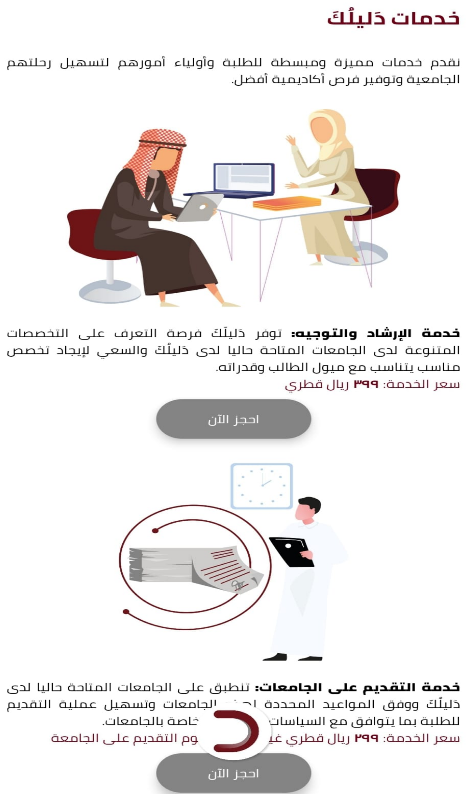
4. File Size: SVG images are best for screens with high resolutions. As they are independent of the resolution their size doesn't change concerning the resolution. Without optimizations, PNG images are up to 70 percent larger in size compared to SVG [21].

3.2.6 Multiple University Support

One of the challenges of the project was to make the application compatible to support multiple universities. What does it mean? Every university can have a different type of data. For example, a university can have a sporting facility but another university doesn't have this facility. Or one university has a special program while other universities don't. The data which we are going to present is dynamic. This means that in some scenarios there can be a screen that is going to present the data which has 5 headings and 3 subheadings for a single university, while for the different universities, the headings can be 7 and subheadings can be 6. This will create a problem. The problem is that we have to let the JSX know dynamically which data is a heading which one is a subheading and which one is normal text so that different formatting can be used to style these different types of data.

Let's see an example from the application to better understand the problem. There is a screen called Our Services which provides the user with information on all the services provided by DALELUK as shown in the figure below.

In the above figure, the bold text represents the headings which is the name of the service provided by Daleluk. The normal text is the description of that service. There are multiple services. The number of services is dynamic which means that it can be changed in the future. But how do we let the JSX know to differentiate the styling properties for both of these texts (bold and normal)? If the number of headings and their descriptions are fixed, then we can easily hard-code the styling attributes for each text type. But since they can be changed in the future we cannot do that.



خدمات دَليلُكَ

نقدم خدمات مميزة ومبسطة للطلبة وأولياء أمورهم لتسهيل رحلتهم الجامعية وتوفير فرص أكاديمية أفضل.

خدمة الإرشاد والتوجيه: توفر دَليلّكَ فرصة التعرف على التخصصات المتنوعة لدى الجامعات المتاحة حاليا لدى دَليلُكَ والسعي لإيجاد تخصص مناسب يتناسب مع ميول الطالب وقدراته.

سعر الخدمة: **٣٩٩** ريال قطري

خدمة التقديم على الجامعات: تنطبق على الجامعات المتاحة حاليا لدى دَليلُكَ ووفق المواعيد المحددة [illegible] الجامعات وتسهيل عملية التقديم للطلبة بما يتوافق مع السياسات [illegible] خاصة بالجامعات.

سعر الخدمة: **٢٩٩** ريال قطري غ[illegible]وم التقديم على الجامعة

Figure 3.10: Our Services Screen

Markdown Data format

To solve the above problem we can use the Markdown data format inside our Firebase noSQL. Using the Markdown data format inside the Firebase is quite beneficial. It is helpful to create dynamic content and real-time updates. The Content owner only needs to update the data in Firebase which will reflect on the application. There is no need to re-build the application and redo the delivery on the App Store and Play Store

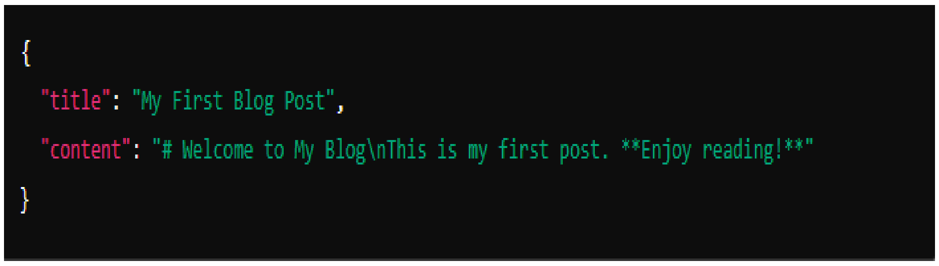
1. Simplicity and Readability: Markdown is easy to write and understand. It uses plain text formatting which is better than HTML and other markup languages. Due to these reasons, it is one of the best approaches for the content owners.

2. Portability: One of the big advantages of using the Markdown format is that it is easily transferred, versioned, and backed up. They can be opened and edited in any tool. They can be used in different operating systems and environments without any compatibility issues.

3. Flexibility and Versatility They are easily convertible to other data formats like HTML, PDF, etc.

4. Maintenance and Version Control: Since they are plain text it works best with version control like Git. This format makes it easier to see the differences and resolve the merge conflicts as compared to other formats such as binary.

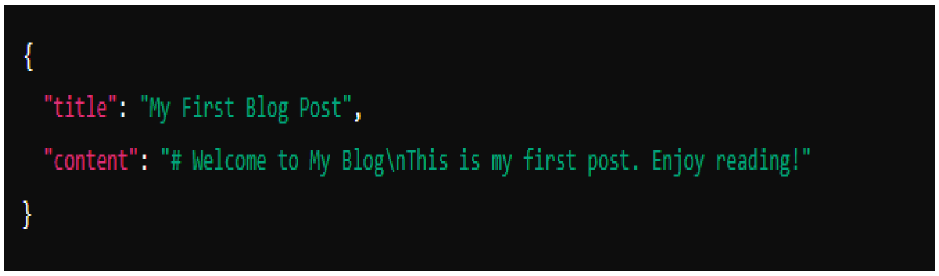
5. Dynamic content: One of the most important requirements of the project is to make the content dynamic. Markdown solves that problem. The content owner can change the data and type of data format very easily using Markdown. e.g.



```
{
  "title": "My First Blog Post",
  "content": "# Welcome to My Blog\nThis is my first post. **Enjoy reading!**"
}
```

Figure 3.11: Markdown data

In the figure, it is shown a JSON object which we can assume is stored in the database. Look inside the object. It's a plain text. As we can see there are multiple markdown tags. Now imagine this scenario from a content content owner's point of view. The CO will use this plain text to format their data accordingly. For example, if the CO wants to use normal text instead of bold text, he/she just needs to remove the "*" from the "content" like this

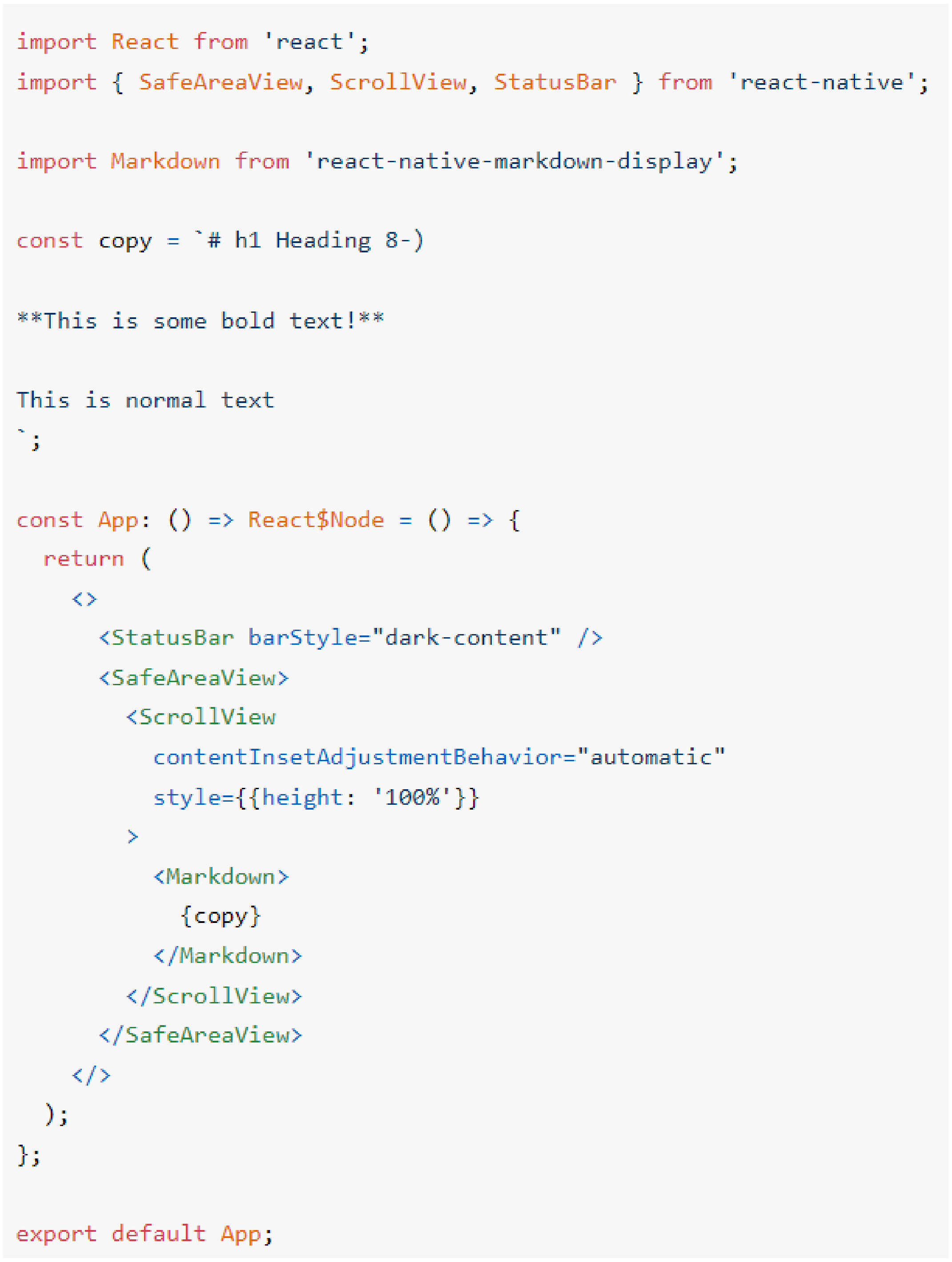


```
{
  "title": "My First Blog Post",
  "content": "# Welcome to My Blog\nThis is my first post. Enjoy reading!"
}
```

Figure 3.12: Markdown updated

So instead of changing the JSX inside the react the CO just changed one minor thing from the database. This change will be reflected to the user in real time thanks to the firestorm.

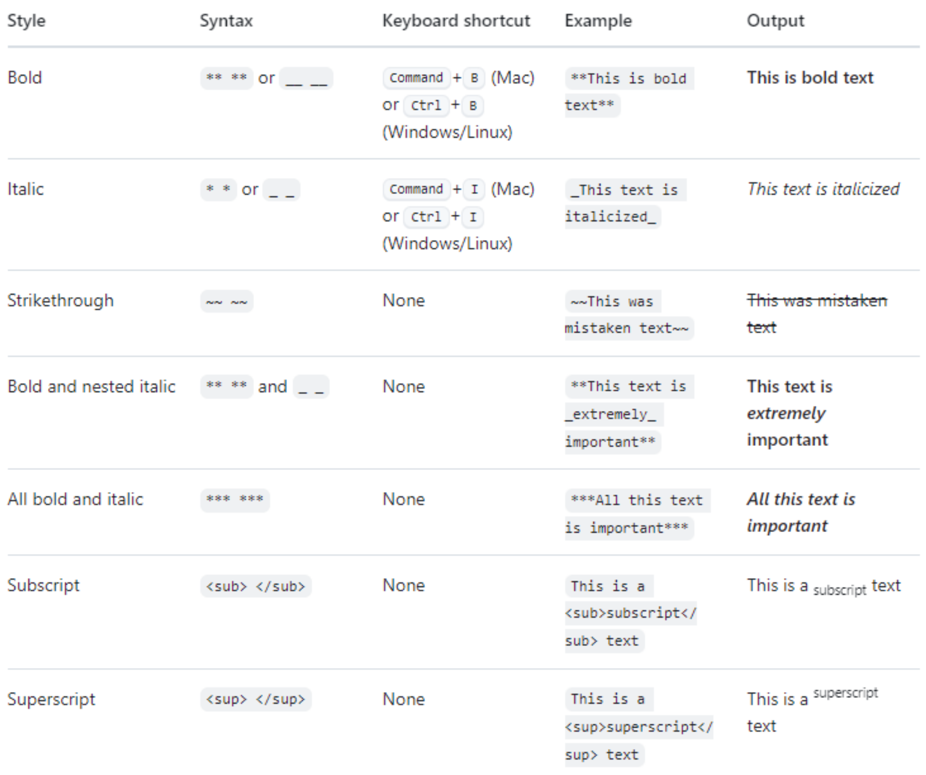
In our application, we are using a third-party library to render the markdown which is called "react-native-markdown-display". it can be easily installed using node package manager by this command **npm install -S react-native-markdown-display**. Here is an example of how to use it inside a React native project.



```
import React from 'react';
import { SafeAreaView, ScrollView, StatusBar } from 'react-native';

import Markdown from 'react-native-markdown-display';

const copy = `# h1 Heading 8-)

**This is some bold text!**

This is normal text
`;

const App: () => React$Node = () => {
  return (
    <>
      <StatusBar barStyle="dark-content" />
      <SafeAreaView>
        <ScrollView
          contentInsetAdjustmentBehavior="automatic"
          style={{height: '100%'}}
        >
          <Markdown>
            {copy}
          </Markdown>
        </ScrollView>
      </SafeAreaView>
    </>
  );
};

export default App;
```

Figure 3.13: Markdown usage

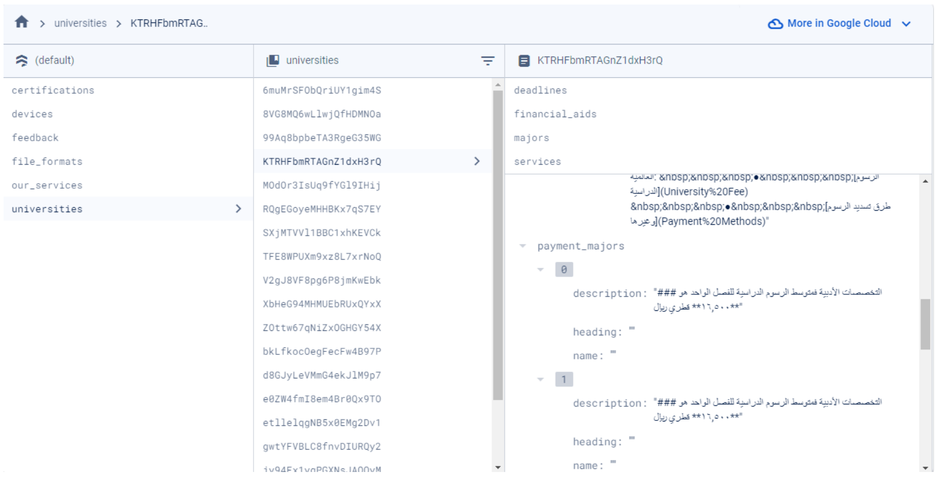
Markdown offers so many formatting options some of the most commonly used are listed below:



Style	Syntax	Keyboard shortcut	Example	Output
Bold	`** **` or `__ __`	`Command` + `B` (Mac) or `Ctrl` + `B` (Windows/Linux)	`**This is bold text**`	**This is bold text**
Italic	`* *` or `_ _`	`Command` + `I` (Mac) or `Ctrl` + `I` (Windows/Linux)	`_This text is italicized_`	*This text is italicized*
Strikethrough	`~~ ~~`	None	`~~This was mistaken text~~`	~~This was mistaken text~~
Bold and nested italic	`** **` and `_ _`	None	`**This text is _extremely_ important**`	**This text is *extremely* important**
All bold and italic	`*** ***`	None	`***All this text is important***`	***All this text is important***
Subscript	`<sub> </sub>`	None	`This is a <sub>subscript</sub> text`	This is a $_{\text{subscript}}$ text
Superscript	`<sup> </sup>`	None	`This is a <sup>superscript</sup> text`	This is a $^{\text{superscript}}$ text

Figure 3.14: Markdown Syntax

Here is an example of a markdown used inside our Firestore. As seen the "description" of the "paymentmajors" inside the "services" contains a text which is a heading of level 3. and the text at the end which is using "**" is the bold text.



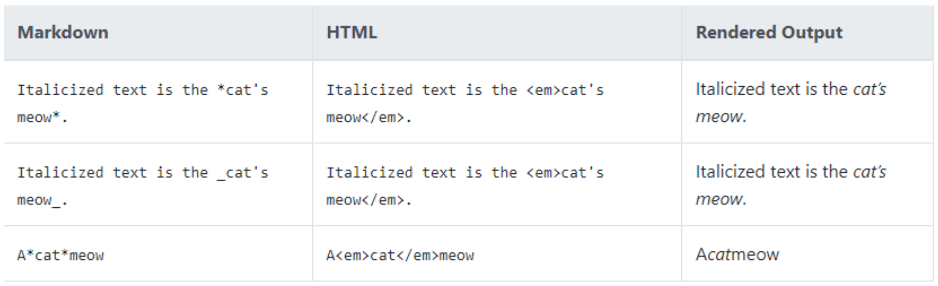
```
universities > KTRHFbmRTAG..                                   More in Google Cloud

(default)            universities               KTRHFbmRTAGnZ1dxH3rQ
certifications       6muMrSFObQriUY1gim4S       deadlines
devices              8VG8MQ6wLlwjQfHDMNOa       financial_aids
feedback             99Aq8bpbeTA3RgeG35WG       majors
file_formats         KTRHFbmRTAGnZ1dxH3rQ       services
our_services         MOdOr3IsUq9fYGl9IHij         العالمية: &nbsp;&nbsp;&nbsp;•&nbsp;&nbsp;&nbsp;[الرسوم
universities         RQgEGoyeMHHBKx7qS7EY         الدراسية](University%20Fee)
                     SXjMTVVl1BBC1xhKEVCk         &nbsp;&nbsp;&nbsp;•&nbsp;&nbsp;&nbsp;[طرق تسديد الرسوم
                     TFE8WPUXm9xz8L7xrNoQ         وغيرها](Payment%20Methods)"
                     V2gJ8VF8pg6P8jmKwEbk       payment_majors
                     XbHeG94MHMUEbRUxQYxX         0
                     ZOttw67qNiZxOGHGY54X           description: "### التخصصات الأدبية فمتوسط الرسوم الدراسية للفصل الواحد هو
                     bkLfkocOegFecFw4B97P           **١٦,٥٠٠** قطري ريال"
                     d8GJyLeVMmG4ekJlM9p7           heading: ""
                     e0ZW4fmI8em4Br0Qx9TO           name: ""
                     etllelqgNB5x0EMg2Dv1         1
                     gwtYFVBLC8fnvDIURQy2           description: "### التخصصات الأدبية فمتوسط الرسوم الدراسية للفصل الواحد هو
                     iv94Fx1vqPGXNsJAQQvM           **١٦,٥٠٠** قطري ريال"
                                                    heading: ""
                                                    name: ""
```

Figure 3.15: Markdown Example inside firestore

Markdown vs HTML format

The main reason for using the Mardown format is its easier to use for content owners. Since they are no developers so choosing the plain text format is the best choice. However, other formats could be used and have some advantages over markdown. HTML is the most popular and widely used format.

Markdown is good for its simplicity and readability but it has several disadvantages compared to HTML. Markdown offers limited formatting options, supporting only basic text styles like headers, lists, and emphasis. This makes it unsuitable for creating complex layouts or interactive elements that HTML can easily handle. Additionally, Markdown documents require conversion to HTML or another format for final rendering, adding an extra step to the workflow. This dependency on external tools can complicate the content management process. Lastly, the lack of native support for semantic tags in Markdown can hinder accessibility and SEO, areas where HTML excels by providing detailed context to search engines and screen readers. So here we can see that the web application will have a low SEO because of the markdown. HTML's extensive set of tags and attributes allows the user for better control over document structure and presentation. It ensures a richer and more accessible user experience. Finally, Markdown's plain text nature means it cannot natively incorporate styles, scripts, or other advanced elements, making it less versatile for web development compared to HTML.



Markdown	HTML	Rendered Output
`Italicized text is the *cat's meow*.`	`Italicized text is the <em>cat's meow</em>.`	Italicized text is the *cat's meow.*
`Italicized text is the _cat's meow_.`	`Italicized text is the <em>cat's meow</em>.`	Italicized text is the *cat's meow.*
`A*cat*meow`	`A<em>cat</em>meow`	A*cat*meow

Figure 3.16: HTML vs Markdown syntax

This image shows a short comparison of the syntax of markdown and HTML. As shown, it is understood that markdown was the best choice in our case. It is more suitable for content owners than HTML. It is less error-prone compared to HTML.

Chapter 4

Analytical Results

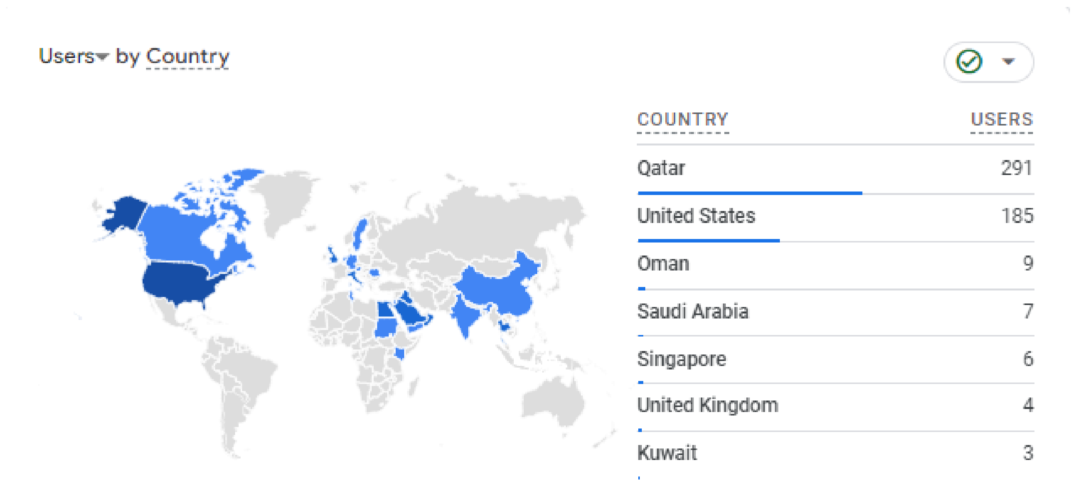
Google Analytics is a powerful web analytics service offered by Google that enables website owners and marketers to track and analyze website traffic and user behavior. By implementing a small piece of code on their application, users can collect a wealth of data regarding how visitors interact with their application. This includes information such as the number of visitors, their geographic locations, the devices and browsers they use, the pages they visit, the duration of their visits, and the paths they take through the site.

The insights gained from Google Analytics help users understand their audience, evaluate the effectiveness of their online marketing strategies, optimize their website's performance, and make data-driven decisions to improve user experience and achieve business goals. Google Analytics provides various tools and reports to visualize data, create custom dashboards, and set up goals to measure specific actions like form submissions or product purchases. Additionally, it offers advanced features such as segmentation, audience insights, and integration with other Google services like Google Ads, providing a comprehensive view of digital marketing efforts.

4.1 Demographical insights

Demographical insights in Google Analytics provide information about the age, gender, and interests of your website visitors. These insights help you understand the characteristics of your audience, enabling you to tailor your content, products, and marketing strategies to better meet their needs. By analyzing demographical data, you can identify key segments of your audience, discover new growth opportunities, and create more targeted and effective campaigns.

4.1.1 Location and language



COUNTRY	USERS
Qatar	291
United States	185
Oman	9
Saudi Arabia	7
Singapore	6
United Kingdom	4
Kuwait	3

Figure 4.1: Users by country

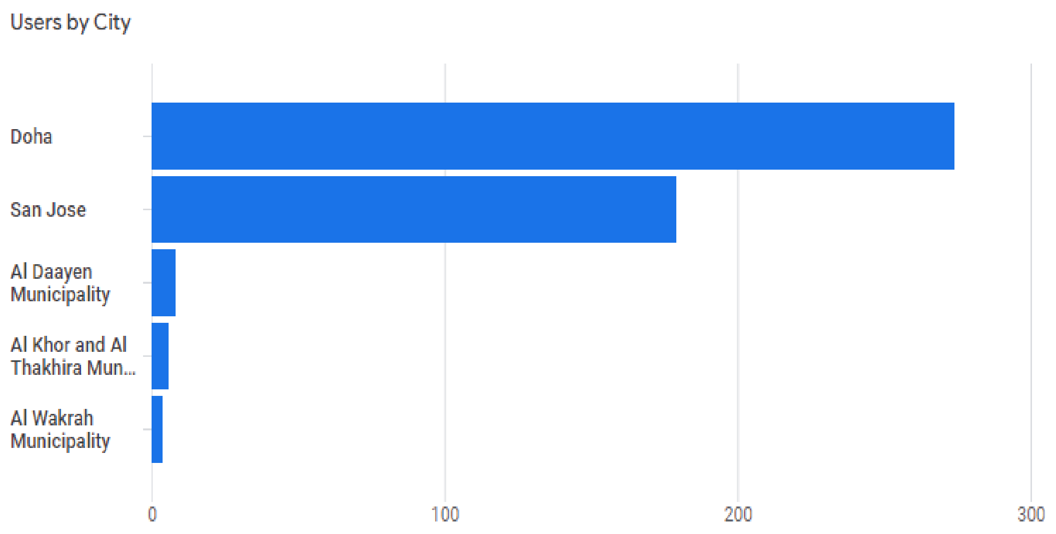


Figure 4.2: Users by city

Conclusion: This is one of the analytical data that shows the location and languages of all the users using the application. We can deduce from this that most numbers of users are from Qatar as expected but there are unexpectedly large

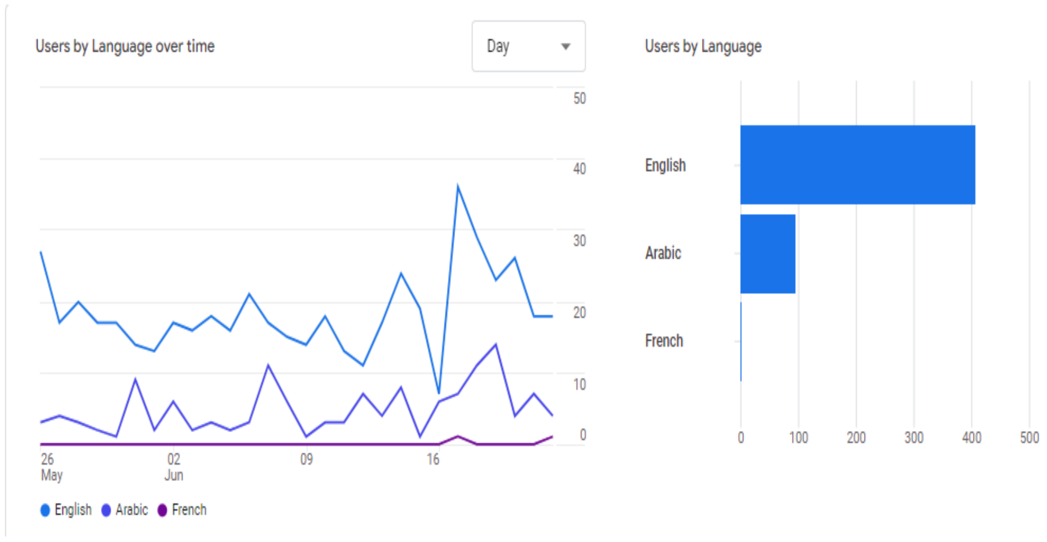


Figure 4.3: Users by languages

number of users coming from the United States. However, since the application is only in Arabic a large number of the audience who speak English cannot use this application. Apart from this, a large number of users are using the application from the capital city of Doha. So the possibility of multi-language support is proposed to Daleluk and more universities from Doha should be added.

4.1.2 Age

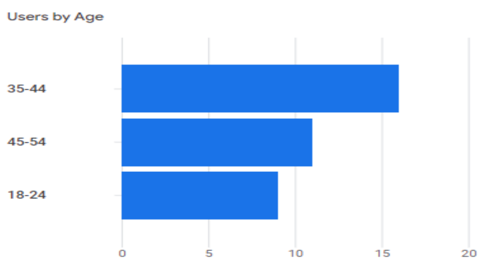


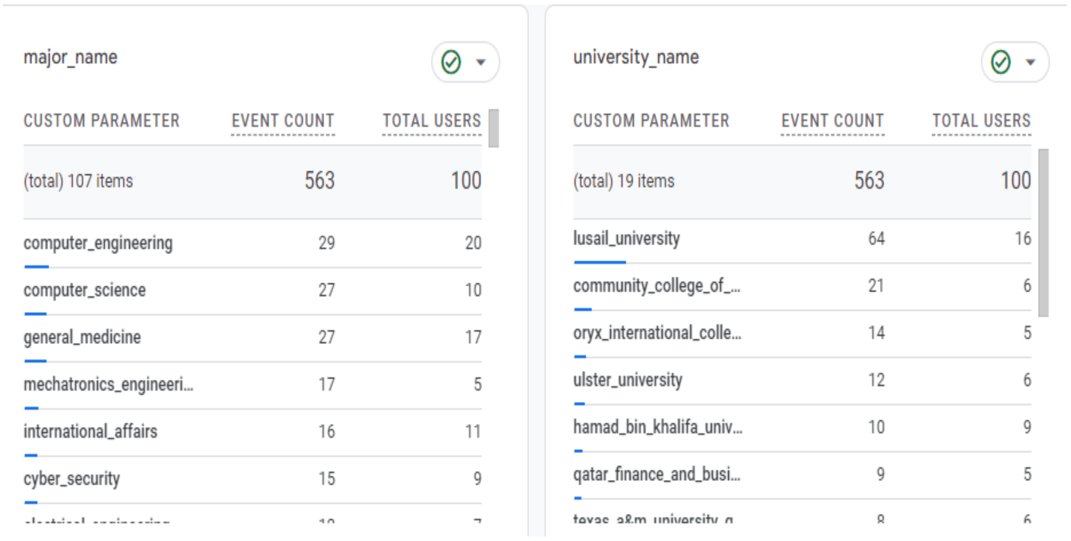
Figure 4.4: Users by languages

Conclusion: As shown, a large amount of the audience is from the age group 35-44. As there isn't any option inside the application to change the font, this could be a problem for users to read the information related to the institutes. So the possibility to change the size of the font is proposed to Daleluk.

4.2 Events

Event logs in Google Analytics track specific interactions or actions that users take on the application, such as clicking a button, downloading a file, or watching a video. These events go beyond standard page views and provide detailed insights into how users engage with your application's elements. By setting up event tracking, we can measure and analyze user behavior more precisely, helping to understand which features are most engaging, identify potential issues, and optimize the application for better performance and user experience.

Some logs were captured to understand which buttons or options a user is clicking. This will give us insight into the application. It's a good way to monitor which part of the application a user is most interested in and which part is redundant.



major_name

CUSTOM PARAMETER	EVENT COUNT	TOTAL USERS
(total) 107 items	563	100
computer_engineering	29	20
computer_science	27	10
general_medicine	27	17
mechatronics_engineeri...	17	5
international_affairs	16	11
cyber_security	15	9
electrical_engineering	10	7

university_name

CUSTOM PARAMETER	EVENT COUNT	TOTAL USERS
(total) 19 items	563	100
lusail_university	64	16
community_college_of_...	21	6
oryx_international_colle...	14	5
ulster_university	12	6
hamad_bin_khalifa_univ...	10	9
qatar_finance_and_busi...	9	5
texas_a&m_university_q	8	6

Figure 4.5: Events of university majors selection

The figure shown above shows the user selection of majors type and universities from the screen "Majors Selection" as shown in majorSelection. This log is helpful for the universities and institutes to find out which majors the applicants are more interested in. For example: As shown the users are more interested in engineering-related subjects.

Conclusion: Daleluk also offers consultation services to its users. To make it happen the users have to go to the "Our Services" screen and then they have to click on "Book Now" to book their appointment with them. But as shown in the

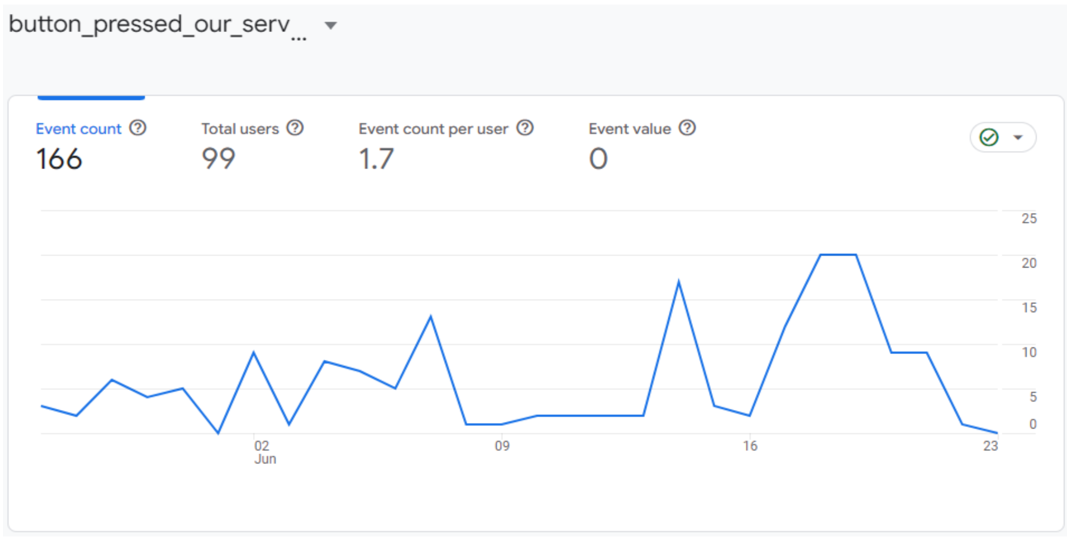


Figure 4.6: Event of Our services

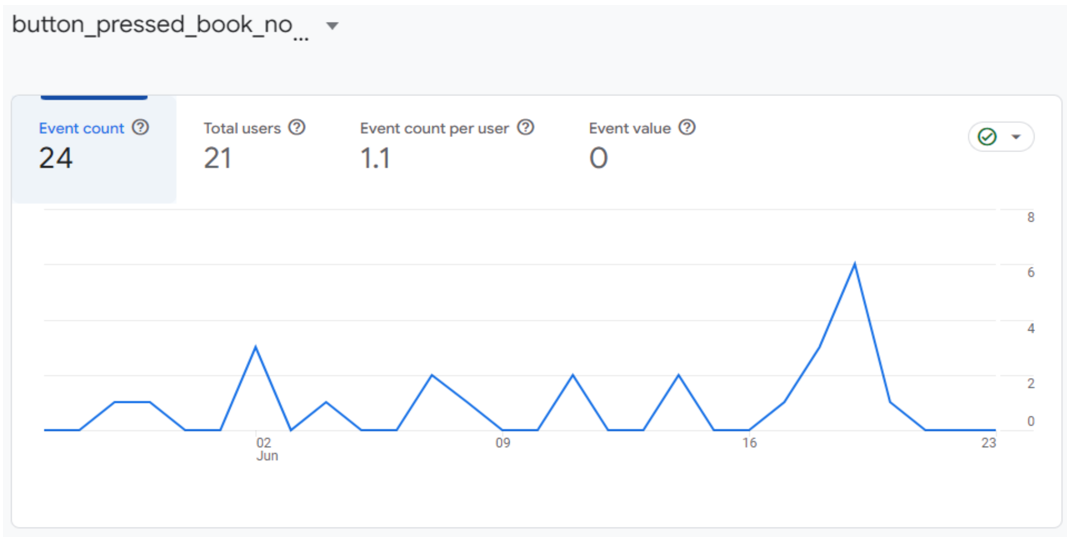


Figure 4.7: Events of Book Now

figures it is found that only 24 users clicked on book now but in total 166 users opened the Our Services screen.

Chapter 5

Conclusion

In conclusion, the development of a mobile and web application using React Native, along with the integration of Google Analytics, has proven to be a robust solution for delivering a seamless user experience. React Native's cross-platform capabilities have enabled the creation of a consistent and efficient user interface across both mobile and web platforms, ensuring that users enjoy a uniform experience regardless of their chosen device.

The use of Google Analytics has been instrumental in enhancing the user experience by providing valuable insights into user behavior and interactions with the application. This data-driven approach has allowed for the identification of key areas for improvement, leading to more informed decision-making and targeted enhancements that directly address user needs and preferences.

By adding the markdown data format for dynamic content storage in Firestore, the application benefits from a flexible and lightweight data structure that simplifies content management and rendering. This approach has facilitated efficient data retrieval and display, contributing to a smoother and more responsive user experience. They are suitable for Content owners.

Overall, the integration of these technologies has resulted in a highly functional and user-centric application. The combination of React Native's versatile development framework, Google Analytics' powerful analytics capabilities, and Markdown's dynamic data handling has enabled the creation of an application that not only meets but exceeds company's expectations.

Chapter 6

Future Work

In the future, there are multiple possibilities for advancing and improving the application. Although the application functions effectively, there is still potential for enhancing its performance, particularly as the number of users increases. The program will be optimized to maintain quick and responsive performance by implementing techniques such as lazy loading, code splitting, and optimizing database queries. Additional features will be taken into account based on user feedback to improve the functionality of the program. Possible enhancements could involve the integration of additional third-party services, the provision of offline functionalities, and the broadening of the range of supported content. It will be essential to continuously improve the user interface by analyzing data and incorporating user feedback. This encompasses not just changes in visual design but also improvements in navigation, accessibility, and overall usefulness. Although React Native provides a robust solution for cross-platform development, there are some advantages to having a distinct codebase for the web application. This method provides enhanced precision in managing web-specific elements such as SEO, superior performance optimizations for web browsers, and the capability to incorporate web-specific functionality without the limitations of a mobile-first framework. Having a specialized web codebase guarantees that both the mobile and online versions of the application can be fully optimized, resulting in an exceptional user experience on each device.

Bibliography

[1] Mehlhorn, N. (2016). Modern Cross-Platform Development for Mobile Applications.

[2] Desmurs-Linczewska, A. (2023). Simplifying State Management in React Native: Master State Management from Hooks and Context through to Redux, MobX, XState, Jotai and React Query. Packt Publishing Ltd.

[3] Paul, A., & Nalwaya, A. (2019). React Native for Mobile Development: Harness the Power of React Native to Create Stunning iOS and Android Applications. Apress.

[4] Ballard, B. (2007). Designing the mobile user experience. John Wiley Sons.

[5] Hansson, N., Vidhall, T. (2016). Effects on performance and usability for cross-platform application development using React Native.

[6] Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H. F., Secret, A. (1994). The world-wide web. Communications of the ACM, 37(8), 76-82.

[7] Roldán, J. C., Jiménez, P., Corchuelo, R. (2020). On extracting data from tables that are encoded using HTML. Knowledge-Based Systems, 190, 105157.

[8] Li, S., Peng, Z., Liu, M. (2004, September). Extraction and integration information in HTML tables. In The Fourth International Conference onComputer and Information Technology, 2004. CIT'04. (pp. 315-320). IEEE.

[9] Clifton, B. (2012). Advanced web metrics with Google Analytics. John Wiley Sons.

[10] Chaffey, D., Ellis-Chadwick, F. (2019). Digital marketing. Pearson uk.

[11] Eisenman, B. (2015). Learning react native: Building native mobile apps with JavaScript. " O'Reilly Media, Inc.".

[12] Danielsson, W. (2016). React Native application development. Linköpings universitet, Swedia, 10(4), 10.

[13] Mandapuram, M. (2016). Applications of Blockchain and Distributed Ledger Technology (DLT) in Commercial Settings. Asian Accounting and Auditing Advancement, 7(1), 50-57.

[14] Dekkati, S., Lal, K., Desamsetti, H. (2019). React Native for Android: Cross-Platform Mobile Application Development. Global Disclosure of Economics and Business, 8(2), 153-164.

[15] Sengupta, D., Singhal, M., Corvalan, D. (2016). Getting Started with React. Packt Publishing Ltd.

[16] Hitz, B. C., Rowe, L. D., Podduturi, N. R., Glick, D. I., Baymuradov, U. K., Malladi, V. S., ... Cherry, J. M. (2017). SnoVault and encodeD: A novel object-based storage system and applications to ENCODE metadata. PloS one, 12(4), e0175310.

[17] Zhou, C. (2024). Challenges and solutions in cross-platform mobile development: a qualitative study of Flutter and React Native.

[18] MOHAMED, A. A. (2015). An Enhanced Approach to Responsive Web Design Influid Grid Concept (Doctoral dissertation, JKUAT).

[19] Web Support, www.necolas.github.io/react-native-web/docs/, accessed on 25th June,2024

[20] React Native Web Limitations, www.https://necolas.github.io/react-native-web/docs/react-native-compatibility/, accessed on 27th June,2024

[21] Data Presentation, www.https://vecta.io/blog/comparing-svg-and-png-file-sizes: :text=Before%20optimization%2C%20PNG%20images%20are,unzipped%2C%206.38KB%20zipped, accessed on 28th June,2024