**NEAR EAST UNİVERSITY**

**FACULTY OF ARTIFICAL INTELLIGENCE AND INFORMATICS**

**DEPARTMENT OF SOFTWARE ENGINEERING**

**COURSE TIMETABLE**

**SUMMER INTERNSHIP**

**SWE299**

**IŞIK SARIDAŞ 20230045**

**NICOSIA-2025**

**ABSTRACT**

Mobile application development is a process of creating software applications that process in mobile devices such as smartphones and tablets. This field encompasses both native development, which targets specific platforms like Android or iOS using platform-specific languages , and cross-platform development, which allows a single code file to operate across multiple systems using frameworks like React Native. Key principles of the mobile application development includes user interfaces and user experiences, performance optimization, security, scalability and integration with backend services. The development process usually involves stages such as planning, desinging, coding, testing,deployment and maintenance.

When students starts to semester they can be stressed about their courses and students may make mistakes such as missing assignment, coming late to classes or they can entirely miss the classes.

The goal of this project is to helping the students about their course management and assisting them to study more efficiently, for this purpose the application will help the students about editing their course schedules, course hours, how many courses they attended during the week and take notes on calendar so they dont miss the important events for them.

The problem for the students is students dont have a tool for managing courses, taking notes on days and looking at the class attendance for each class in their hands. Different students will use the application differently as they explore the application more but in the base, application will provide course management, allow students to take notes on spesific days and show students their attendance state. The project created in JavaScript language, used React Native for cross platform accesibility and Expo libraries for easier development cycle.

The conclusion of the project points out that this application will help the students for their each semester and be their notebook for tracking courses.

**TABLE OF CONTENT**

ABSTRACT ii

TABLE OF CONTENT iii

INTRODUCTION 1

BACKGROUND 2

CHAPTER ONE: DAYS PAGE 13

CHAPTER TWO: CALENDAR PAGE 22

CHAPTER THREE: STATİSTİCS PAGE 27

CHAPTER FOUR: STORAGE FILE 30

CONCLUSION 31

REFERENCES 32

**INTRODUCTION**

When mobile phones started to manufactured by companies, applications inside the mobile phones were pre-installed by the manufacturers and there was no way for user to install any applications to the phone. After manufacturers started to improve their systems, users started to install applications to their phones and started using mobile phones and mobile applications in daily life.

Today, mobile apps are usually distributed via an official online outlet or marketplace and there is a formalized process by which developers submit their apps for approval and inclusion in those marketplaces. To facilitate developing applications for mobile devices, and the consistency thereof, various approaches have been taken. Historically, however, that was not always the case. The software development packages needed to develop, deploy, and manage mobile applications are made from many components and tools which allow a developer to write, test, and deploy applications for one or more target platforms.React Native is one of the frameworks that supports cross platform interaction for ios and android. Back-end tools pick up where the front-end tools leave off, and provide a set of reusable services that are centrally managed and controlled and provide the following abilities:  
 Integration with back-end systems  
 User authentication-authorization  
 Data services  
 Reusable business logic in mobile apllication porcess

With using React Native the students who uses both ios and android can reach our appliation and track down their classes. Application has three main pages and each page in the application offers a new feature for the user to enjoy.

Chapter one describes the Days page, here students will find the Days and courses in the days for their preference.This page will offer adding days, editing the days information modal and navigating to other pages.

Chapter two presents Calendar page, in here students will find a classic calendar and a modal on the bottom of calendar for days and their description. Students can click the day for editing the day and a information modal will open for that day.

Chapter three is for the Statistics page, it will show the students how many classes they attended through the week and if they attended the courses invidualy in the modal under.

Chapter four is devoted for the Storage file that Days and statistic page shares so they can interact with each other in real time in the application.

**BACKGROUND**

**Mobile application development:** Is the process of developing mobile app for one or more mobile devices. This software applications are spesially designed for mobile devices that contains many hardware constraints, including CPU, RAM and limited data storage capabilities. These applications can be pre-installed on phone during maufacturing or can be delivered as web aplications in a web browser.

As smartphones have become more common, so has the need for apps. Today, almost everyone relies on mobile apps in some way, which is why developing them has become such an important part of the tech industry.  
 With the rapid growth of smartphone usage around the world, mobile apps have become a major part of everyday life. People use apps to communicate, shop, learn, and entertain themselves. Because of this demand, mobile app development has become one of the most important areas in the technology industry.

As one of the part of mobile applications, mobile user interfaces is an important step in creating the application and addresing the user satisfaction. When designing the user interface designer considers different variables like context, screen space, input methods and mobility for design. Designers should alsp cpnsider the challenges in the mobile user interface design, including users attention span to devices size for the user. Mobile user interface context takes signals from when users using the device or where they using it,these factors observes when user interacted with mobile application. For these context signals can be used for customizing the user experience by showing the user related ads or a similar activity.

The users often focuses on the experience with their devices and interfaces contains both hardware and software. User input lets the user manipulate the system and the device outputs shoes the effects of the user manipulations.

Mobile app development is the process of creating software applications that run on mobile devices like smartphones, tablets, and PDAs. These apps can be pre-installed during manufacturing or downloaded later by users. Developers must consider hardware limitations such as CPU speed, RAM, storage capacity, screen size, and input methods when designing these applications.

Developers use different programming languages depending on what they’re building. For example, Android apps are usually made with Java or Kotlin, while iOS apps are often built using Swift. Tools like React Native and Flutter let developers build apps for both platforms at once.

There are different types of mobile apps: native, hybrid, and web based. Native apps are built specifically for one platform (like iOS or Android) and offer high performance and a consistent user experience. Hybrid apps combine elements of both native and web apps, allowing for cross-platform compatibility using a single codebase. Web based apps run in browsers and require an internet connection, making them less resource intensive on devices.

The development process involves several stages:

1. **Planning and Strategy**: Defining the app's purpose, target audience, and key features.
2. **Design**: Creating the user interface (UI) and user experience (UX) designs, considering the constraints of mobile devices.
3. **Development**: Writing the code for the app, which may involve using platform-specific languages or frameworks that support cross-platform development.
4. **Testing**: Ensuring the app functions correctly across different devices and operating systems.
5. **Deployment**: Releasing the app to app stores or distributing it directly to users.

Security is a critical aspect of mobile app development. With the rise of Bring Your Own Device (BYOD) policies in workplaces, developers must implement measures like app wrapping, data encryption, and secure client-server communication to protect user data.

In summary, mobile app development is a multifaceted process that requires careful planning, design, and implementation to create effective and secure applications for various mobile platforms.

**React Native:** React Native is a open sourse UI software framework developed by Meta platforms. It is used to develop applications for Android, AndroidTV, iOs, macOS, tvOS, web, windows ans UWP by offering developers to use the React framework along with native platform capabilities. It is used to develop applications for Android and ios. React Native can also be used for developing virtual reality applications at Oculus.

The way of React Native is similar to React except React Native does not manipulate the DOM from the Virtual DOM. Instead it runs in the background process on the end device and communicates the native platform via serialized data over an asynchronous and batched bridge.

React items covers the existing native code and interact with the Native APIs via Reacts declerative UI paradigm and Javascript/Typescript. React native may look like it has similar syntax with CSS but it doesnt use any CSS or Html instead JavaScript thread are used to manipulate native views. React Native is also available for both Windows and MacOS, which is currently maintained by Microsoft. An example for React Native code for printing hello world on screen: 

**Expo Framework:** Expo is a React Native framework, and the only production-ready one, to date. Expo started around 2016 because building apps with React Native was a bit too hard for many people, especially beginners. You had to install heavy tools like Android Studio or Xcode, and deal with native code, even if you just wanted to test a simple app.

Expo: developer tools. Overview of tools Expo offers as open source, for free. Shipping to production with EAS. Expo generates revenue with its Expo Application Services a freemium remote build and submission service. n the early versions, Expo focused on giving developers tools like:

* Expo Go
* A web based editor,
* And a command line tool to build and publish apps easily.
* By 2017/2018, more people started using Expo because it had a clean workflow and a helpful SDK

By 2017/2018, more people started using Expo because it had a clean workflow and a helpful SDK. They added support for things like camera, push notifications, and location all built in, so you didn’t have to code them from scratch. They included many built-in features like the camera, push notifications, and location. These were ready to use right away, which was great for developers who just wanted to get started quickly.

As time passed, more and more developers started using Expo. Then in 2020, they introduced EAS, which stands for Expo Application Services. This was a big upgrade. It allowed us to build our apps in the cloud, send them to the App Store or Google Play, and even use custom native code if needed.

**JavaScript:** JavaScript often abbreviated as **JS**, is a programming language and core technology of the World Wibe Web alongside HTML and CSS. Ninety-nine percent of websites use JavaScript on the Client side for webpage behavior. Web browser have a dedicated JavaScript Engine that executes the client code. These engines are also utilized in some servers and a variety of apps. The most popular runtime system for non-browser usage is Node.js.

Microsoft first released JScript in 1996, alongside initial support for CSS and extensions to HTML. Each of these implementations was noticeably different from their counterparts in Netspace Navigators These differences made it difficult for developers to make their websites work well in both browsers, leading to widespread use of "best viewed in Netscape" and "best viewed in Internet Explorer" logos for several years

JavaScript is a High level, often just in time compiled language that conforms to the ECMAscript standard. It has dynamic typing, prototype based object orianted , and first class functions. It is multi paradigm, supporting event driven, functional, and imperative programming styles. It has application programming languages (APIs) for working with text, dates, regular extentions, standard data structures, and the Document object model (DOM).

The ECMAScript standard does not include any input/output (I/O), such as networking, storage, or graphics facilities. In practice, the web browser or other runtime system provides JavaScript APIs for I/O.

Although Java and JavaScript are similar in name and syntax, the two languages are distinct and differ greatly in design.

The goal was a "language for the masses", "to help nonprogrammers create dynamic, interactive Web sites.” Netscape management soon decided that the best option was for Eich to devise a new language, with syntax similar to Java and less like Scheme or other extant scripting languages.

During the period of Internet Explorer dominance in the early 2000s, client-side scripting was stagnant. This started to change in 2004, when the successor of Netscape, Mozilla, released the Firefox browser. Firefox was well received by many, taking significant market share from Internet Explorer.

In 2005, Mozilla joined ECMA International, and work started on the ECMAscript for XML  (E4X) standard. This led to Mozilla working jointly with Macromedia, who were implementing E4X in their ActionScript 3 language, which was based on an ECMAScript 4 draft. The goal became standardizing ActionScript 3 as the new ECMAScript 4. To this end, Adobe Systems released the Tamarin implementation as an open source project. However, Tamarin and ActionScript 3 were too different from established client-side scripting, and without cooperation from Microsoft, ECMAScript 4 never reached fruition.

Meanwhile, very important developments were occurring in open-source communities not affiliated with ECMA work. In 2005, Jesse James Garrett released a white paper in which he coined the term Ajax and described a set of technologies, of which JavaScript was the backbone, to create web appliations where data can be loaded in the background, avoiding the need for full page reloads. This sparked a renaissance period of JavaScript, spearheaded by open-source libraries and the communities that formed around them. Many new libraries were created, including jQuerry, Prototype, Dojo toolkit, and MooTools.

The creation of Node.js in 2009 by Ryan Dahl sparked a significant increase in the usage of JavaScript outside of web browsers. Node combines the V8 engine, an event loop, and I/O APIs, thereby providing a stand-alone JavaScript runtime system. As of 2018, Node had been used by millions of developers, and npm had the most modules of any package manager in the world.

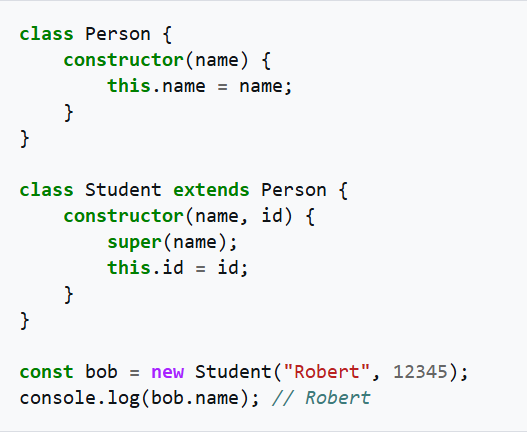
The ECMAScript draft specification is currently maintained openly on GitHub, and editions are produced via regular annual snapshots. Potential revisions to the language are vetted through a comprehensive proposal process. Now, instead of edition numbers, developers check the status of upcoming features individually.

The current JavaScript ecosystem has many libraries and frameworks, established programming practices, and substantial usage of JavaScript outside of web browsers. Plus, with the rise of single page applications and other JavaScript-heavy websites, several transpilars have been created to aid the development process.

The first engines for JavaScript were mere interpreters of the source code, but all relevant modern engines use just in time compilation for improved performance. JavaScript engines are typically developed by web browser vendors, and every major browser has one. In a browser, the JavaScript engine runs in concert with the rendering engine via the Document Object Model and Web IDL  bindings. However, the use of JavaScript engines is not limited to browsers; for example, the V8 engine is a core component of the Node.js runtime. Since ECMAScript is the standardized specification of JavaScript, ECMAScript engine is another name for these implementations. With the advent of WebAssembly, some engines can also execute this code in the same sandbox as regular JavaScript code.

JavaScript is a single threaded language. The runtime processes messages from a queue one at a time, and it calls a funciton associated with each new message, creating a call stack frame with the function's arguments and local variables. The call stack shrinks and grows based on the function's needs. When the call stack is empty upon function completion, JavaScript proceeds to the next message in the queue. This is called the event loop, described as "run to completion" because each message is fully processed before the next message is considered. However, the language's concurrency model describes the event loop as non-blocking: program I/O is performed using events and callback functions. This means, for example, that JavaScript can process a mouse click while waiting for a database query to return information.

In JavaScript, an object is an associative array, augmented with a prototype each key provides the name for an object property, and there are two syntactical ways to specify such a name: dot notation (obj.x = 10) and bracket notation (obj["x"] = 10). A property may be added, rebound, or deleted at run-time. Most properties of an object (and any property that belongs to an object's prototype inheritance chain) can be enumerated using a for...in loop. JavaScript uses prototypes where many other object-oriented languages use classes for inheritance  but it's still possible to simulate most class-based features with the prototype system. Additionally, ECMAScript version 6 (released June 2015) introduced the keywords **class**, **extends** and **super**, which serve as syntactic sugar to abstract the underlying prototypal inheritance system with a more conventional interface. Constructors are declared by specifying a method named **constructor**, and all classes are automatically subclasses of the base class Object, similarly to Java:



Though the underlying object mechanism is still based on prototypes, the newer syntax is similar to other object oriented languages. Private variables are declared by prefixing the field name with a number sign (#), and polymorphism is not directly supported, although it can be emulated by manually calling different functions depending on the number and type of arguments provided.

Unlike in many object-oriented languages, in JavaScript there is no distinction between a function definition and a method definition. Rather, the distinction occurs during function calling. When a function is called as a method of an object, the function's local this keyword is bound to that object for that invocation.

JSON is a data format derived from JavaScript; hence the name JavaScript Object Notation. It is a widely used format supported by many other programming languages.

**CHAPTER ONE: DAYS PAGE**

First page and the default page of the application is days page before explaining its key features and providing a code example, I would like to begin by showing the purpose of the libraries it imports:

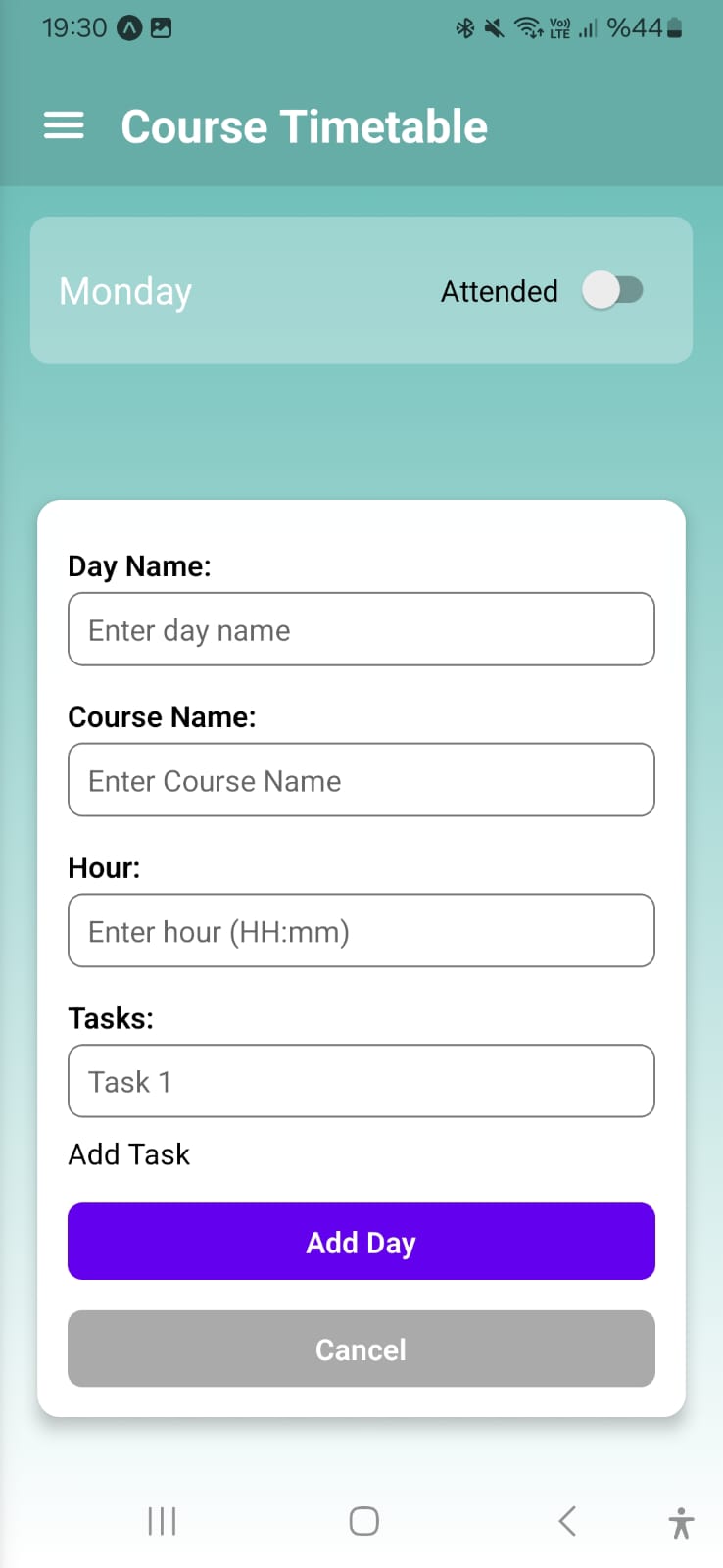


**Figure 1.1**

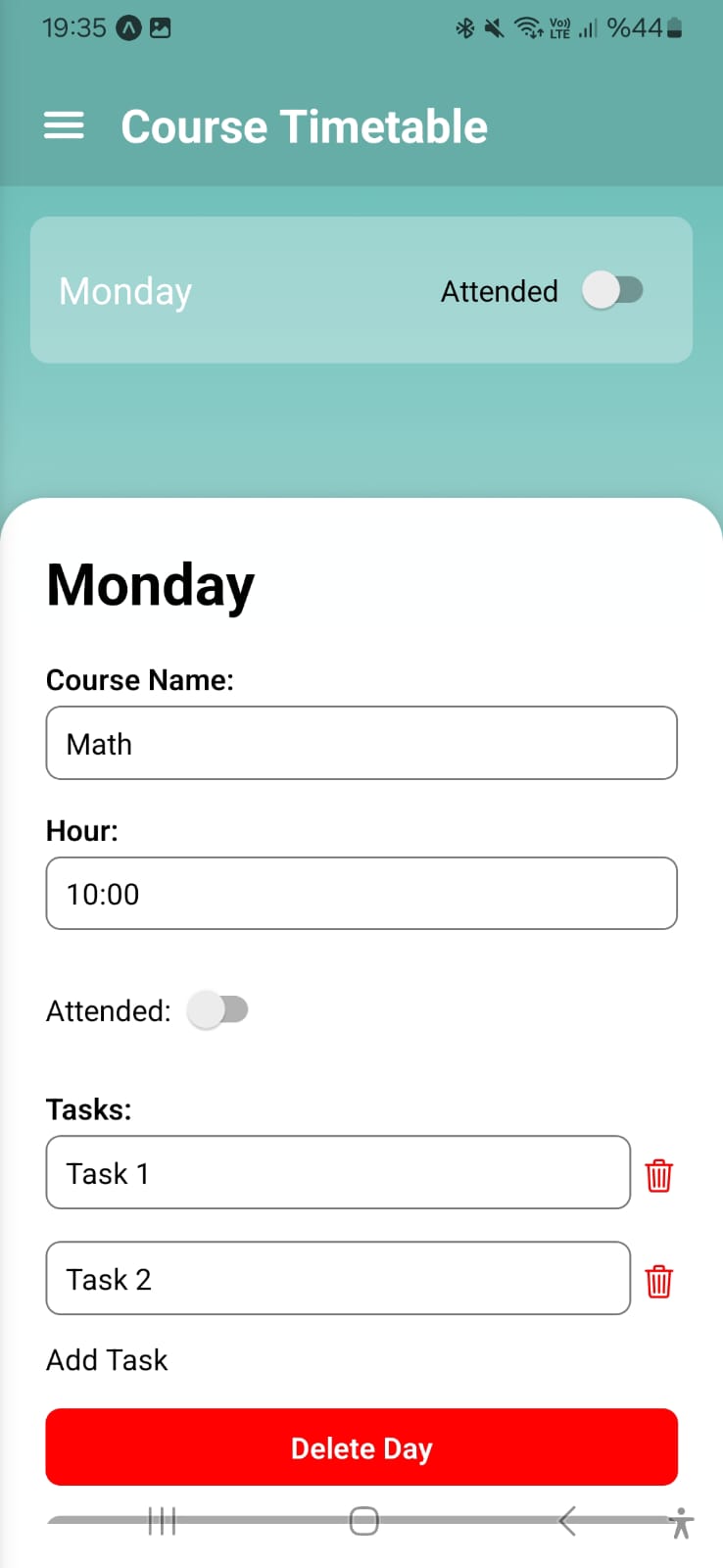
* import { Ionicons } from "@expo/vector-icons";
* This Librairy is a collection of free, ready-to-use icons. This library imported to the code to display visually intuitive icons that improve the user interface. These icons can be used within buttons or next to text labels to make the app more user-friendly and visually informative. The icons are vector-based, meaning they scale without losing quality, which is ideal for devices with different screen sizes and resolutions.
* import { LinearGradient } from "expo-linear-gradient";
* This library lets you create smooth, linear gradient backgrounds or overlays in your UI. It is usefull for improving visual appeal and highlighting certain elements. A gradient is a gradual transition between two or more colors. With Linear gradient you can define two points in the as the start of the gradient and the end of the gradient, and assign colors to them for color transmition between the two points. It works by rendering a custom native view that blends colors in the specified direction. In this application this library used as background color tool.
* import { useNavigation } from "expo-router";
* This is part of Expo Router, a file-based navigation system for React Native apps. useNavigation provides acces for navigation functions to move between other pages, going back or updating screen options. It works by integrating React Navigation’s core features but simplifies route management by linking it directly to your file structure. Instead of manually configuring the routes opening files for the navigation is enough and the expo router handles the navigation between them.
* import { useEffect, useState } from "react";
* This library is built in hooks from React used for managing component behvaiour as:
  + UseState lets you create and update local state variables within functional components.
  + UseEffect lets you side effect like setting timers, triggering notifications, fetching data when a component updates. This effects trigger after every render or when specific dependencies change.

Together, these hooks allow you to create interactive components that responds to user interaction without needing class components.

* import {
* Animated,
* Dimensions,
* FlatList,
* ScrollView,
* StyleSheet,
* Switch,
* Text,
* TextInput,
* TouchableOpacity,
* View
* } from "react-native";
* React Native provides a set of core components that are the foundation for building any mobile application. These components behave like a building blocks for creating user interfaces that supports both iOS and Android platforms. Each of these components serves a different purpose in the code with relatively little code. These components are:
  + **Animated:** Animated is used to create visually appealing, interactive animations like fades, scaling and slides. Allows you to animate values like opacity and position over time. In the code this used as animator for navbar openin animation as sliding.
  + **Dimensions:** Dimensions gives you information about the screen size of the device as width and height in pixels. This is used for creating responsive layouts to adapt different phone sizes.
  + **FlatList:** Flatlist is a way that rendering large list efficiently. It only renders what’s visible on the screen and loads more as the user opens new areas inside the application. It accepts the data and a render item as function to describe how each item should appear.
  + **ScrollView:** ScrollView is a container that enables vertical (or horizontal) scrolling. It is useful when your content doesn’t fit the screen. All children inside it become scrollable. Unlike FlatList, it renders all its child components at once, which is fine for small lists.
  + **StyleSheet:** Stylesheet is a tool to define and manage styles in a clean and performance efficient way. Instead of writing inline styles you can write all of the style properties inside of one object and apply them using style names inside the object.
  + **Switch:** Switch is a simple on/off feature that commonly used for boolean options for if something is on or off. In the applicaon this feature used as attendance toogle button.
  + **Text:** This component is used to display any textual content in the application. As an example text in the day information modal or adding day modal texts. React Native requires all of the text to be in this component otherwise it wont render properly.
  + **TextInput:** TextInput is an input space that users can interact with and enter text. In the appliation this feature used as costumizing the days information or when adding the day. You can also control it using the useState hook to dynamically update the value the user types.
  + **TouchableOppacity:** this component can wrap around any element and make it tappable. It gives visual feedback with reducing opacity when hovered, indicating to user that its clickable. You define what happens on press using the onPress prop.
  + **View:** This component is one of the most fundamental parts of React Native. It acts like a container that holds other components such as text, button or images. You can apply styling to a View to control things like size, background color, alignment, and margins. In a general layout view components are nested to form of sections, rows or collums in user interface.
* import { FAB, Provider } from “react-native-paper”;
* React-native-paper is a user interface library offering pre-styled features like buttons, text inputs and floating action buttons (FAB). This library is used for adding a floating action button on the application as adding a day on the page. Provider is a wrapper component that allows these Paper components to access theme settings.  
  This library ensures that your app looks consistent, professional, and follows modern design practices without writing a lot of custom styles.
* import { StoreForDay } from "../../Store/Store";
* This is not a library that i imported from React Native or expo. This is a file in the application that contains various days components inside for beign the brigde for days page and statistics page. While both pages takes data from this file they are synchronizing and working on real time with each other as they take the same data in the same time from the same file. In days page this feature used for days data to share with statistics page. When a day opens in days page statistics page saws the changes in the page in real time and adapts to it by adding the day to itself.
* import \* as Notifications from "expo-notifications";
* This Expo module allows appliation to schedule, display, and manage notification sending both from local(from the device) and push(from a server). In the application this feature used for sending notifications to students device 2 hours earlier from the stored data from days information modal. Expo-notifications handles scheduling with scheduleNotificationAsync(), which lets you specify content and trigger time. It works with native APIs on background means it handles different feedbacks from iOS and Android platforms. This library also supports to requesting permission for notifications and canceling notifications.

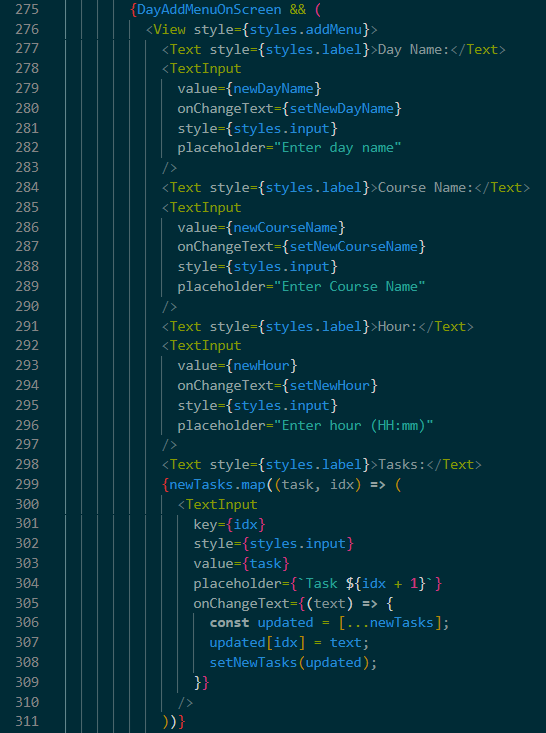
All these libraries handles a different feature of the application. The first feature of in the page is Adding days. As i stated before on the react-native-paper library it is done by a floating action button (FAB). This button is on the right bottom on the screen and when pressed a modal for adding day opens.

User can costumize the current variables shown in the picture Day name, Course name, Course hour (case sensitive for notifications) and Tasks.

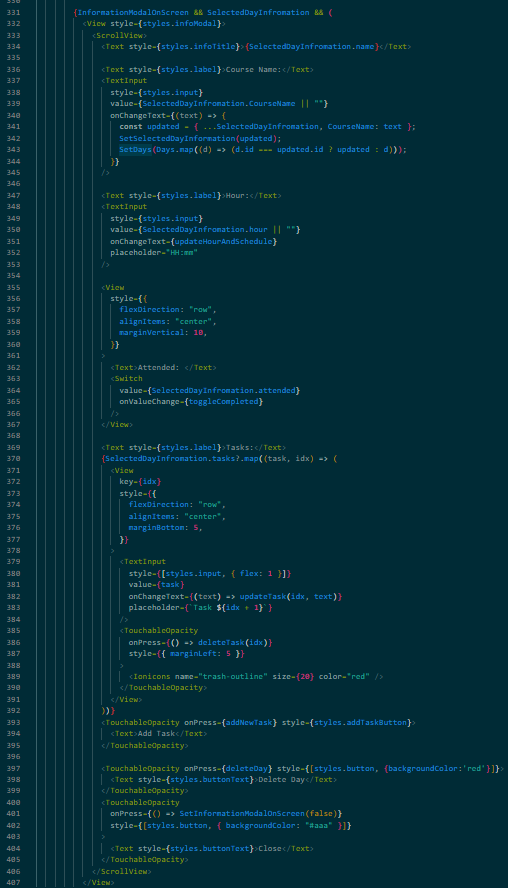
User can change these variables Whenever they want when they add the day. When they add the day a day modal will show on the screen as shown in the figure. When user clickes this modal an days information modal opens as in the second figure and in this page all the variables in the FAB. Respectively Day name on top of the modal, Course name, Course hour( case sensitive for notifications), Class attention toggle button, Tasks the user can add as they want and take notes for class and on their right a trash bin for deleting task, lastly Delete Day deletes day and Close button closes the day information modal, in the figure you cant see the close button because the days information modal can be slided to bottom for both compactibility of the feature and When there is a lot of tasks and prevents the modal from overflowing from the screen. The attendance toggle button is interactable on both from days information modal and from the main menu day component.

**Figure 1.2**

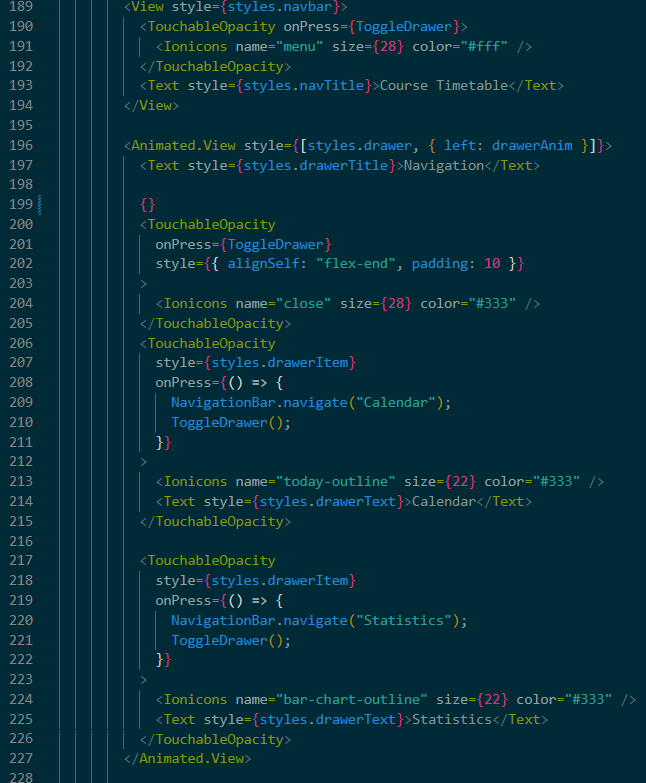
**Figure 1.3**

 For better understanding i am going to show codes from the project and explain with visual support from images. The first Feature i will explain will be adding days to the main page. In this figure DayAddMenuOnScreen components shows this code is the background of the add day menu on screen as shown in the previous figures. When FAB button is pressed and this modal opens. View is the container component for all add day features and it takes styles from addMenu style in the stylesheet object. Then for user to interact with this menu TextInput is used under the features name it takes the datas as newDayName and setNwDayName and stores it. Under the Day name input is respectively Course name and Course hour. Corse name has the same properties with the Days name but the Course hour is case sensitive because when notifications checks the time in the days if it doesnt recieve the exact information from it it wont send a notification. Under the Course hour input feature is tasks feature. There is one task input panel for default in this modal for user to take notes but if user wants they can add more task from here.

**Figure 1.4**

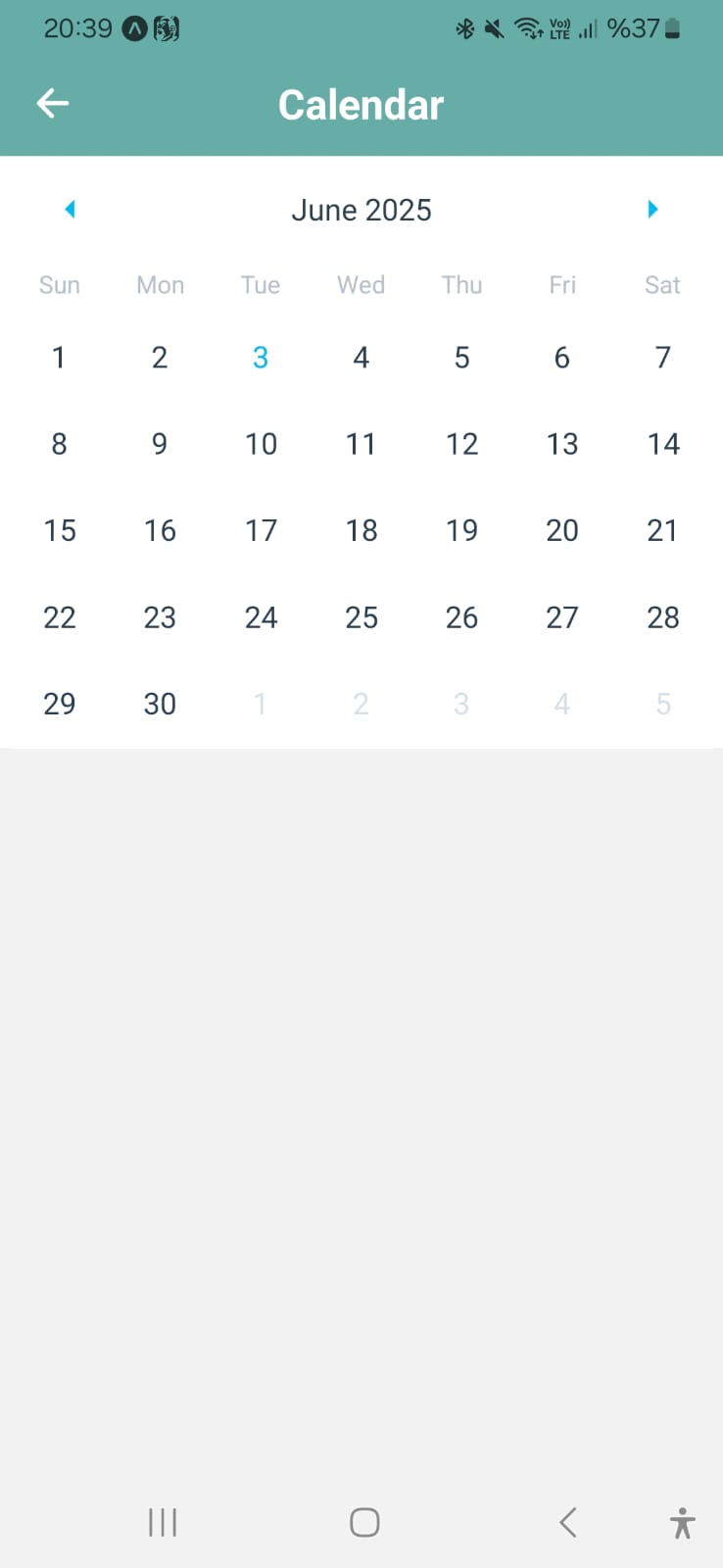
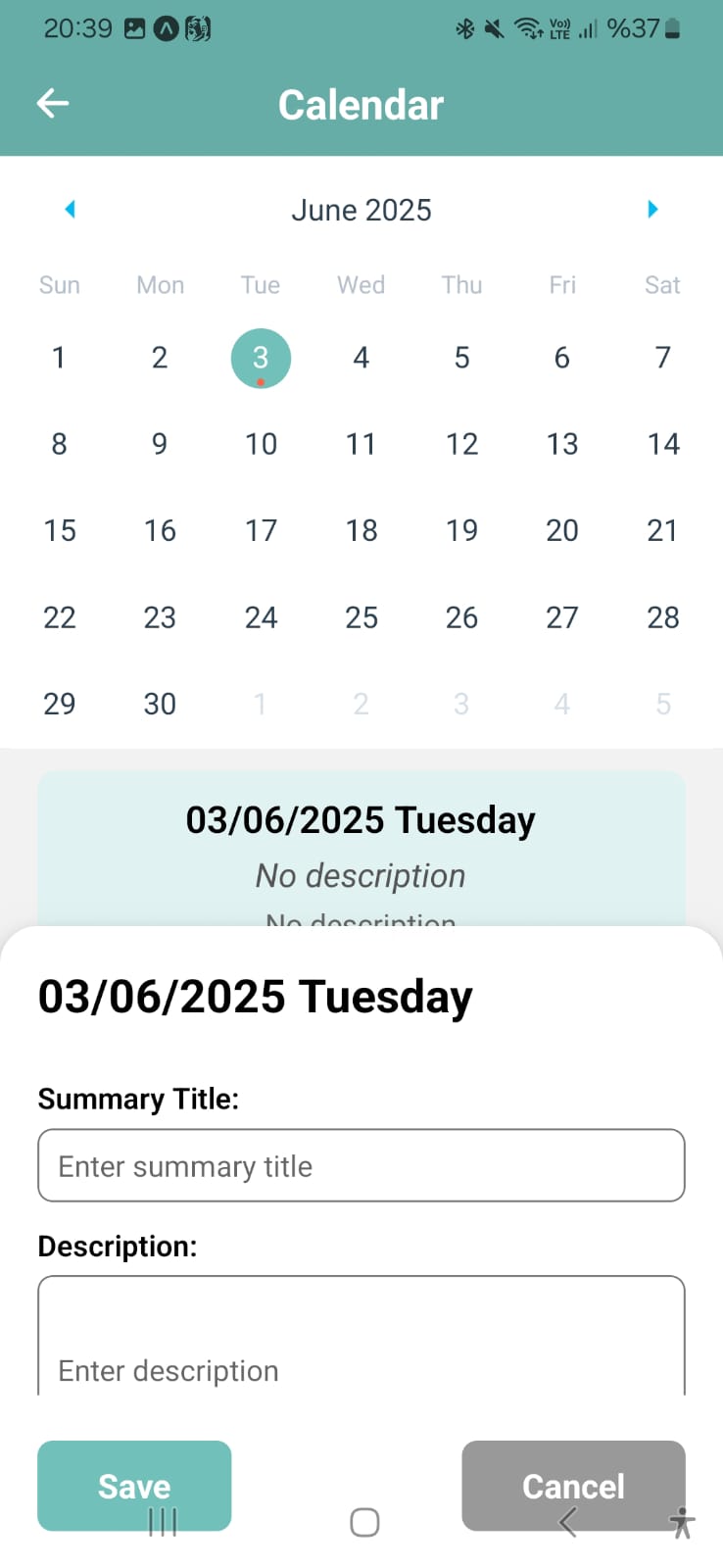
 Second feature is the days information modal. This code is the background of the days information modal on the screen. This code starts with taking all the components inside a container with view command and takes it style from the infoModal from the stylesheet objective. On top of the days information modal is the Day name this is the only variable user cant change when they opened a new day. Under the day name there starts the interactable variables for users like the Course name and Course hour. These features uses the same method as adding day and with TextInput component. Then there is a gap between Hour and the Task features because in this modal between them there is a Toggle button for attendance checking. This button is uses the boolean checking system and default for this button is false, when toggled this buttons value changes and this value change reflects to the statistics page in real time. After this button there is Task input area on that right its the trash bin for deleting the task area and under that an button for adding new tasks.

**Figure 1.5**

 Third and Last feature for this page is the navbar/Drawer feature the default navbar is always on top of the main page with the title of Course Timetable, drawer menu is the left of this title as a three lines. When this button clicked the drawer for navigation between the other pages opens up with a little animation from popping up from left of the screen. When drawer is open user can see the title of Navigation title at top of the screen and under the title is the buttons for navigating to other pages respectively, Calendar and Statistics. These buttons have their own icons for them and use the TouchableOpacity library, this means when user hovers over these buttons these buttons will reduce their opacity to indicate they can be pressable.

**Figure 1.6**

**CHAPTER TWO: CALENDAR PAGE**

 When navigated from the main page to second page/Calendar page the first thing the user will see is a basic calendar with days, months and years. The navbar for this page is simply top of the screen with a title of Calendar and a backtrack button for going back to the Days page. When user inspects the calendar they can see the current day with light blue color on the calendar. The user simply can click any day to take notes for that day. User will see a red dot under the selected day indicating the day they will take notes to. When a day selected there will be information panel right under the calendar and this panel will contain the exact date at the dd/mm/yyyy day format as the title, under the title there is a main description input area and a detailed input area like shown in the figure. User can see the no description text when they first opened a day information modal under the calendar but also a modal for user to interact will be opened when user clicked a day. This modal will contain the same structure as the information panel under the calendar as the dd/mm/yyyy day format title summary title as the input area for short description for day and a more detailed description input area for all the notes that user want to take on that day. This modal is slidable, the reason for this is when user enters a lot of notes to the description part modal may stay compact and wont overflow from the screen. Under the description, there save and cancel buttons. Save button saves the information user entered and cancel button will close the modal without saving the changes from the modal.

**Figure 2.2**

**Figure 2.1**

The background of this page contains same libraries as days page and a few extra libraries these libraries are:

**Figure 2.3**

* import { SafeAreaView,…} from “react-native”;
* Safeareaview is a warpper like view but it ensures the content inside of it renders within the safe areas of the device screen, especially important for iPhones that have the notches or android devices with curved edges. When wrapped to a content React Native adjustes automatically the position of content to avoid any overlaps with the status bar , notch or home gesture bar.
* import { …, KeyboardAvoidingView, … } from “react-native”;
* Prevents the keyboard from overlapping input fields by shifting the content upward when the keyboard appears. On iOS and Android, when the user focuses on a TextInput, the keyboard may cover it. KeyboardAvoidingView automatically adjusts the layout using padding or height so users can still see what they’re typing.
* import { …, Platform, …} from “react-native”;
* Used to detect which platform the app is running on—either Android or iOS. You can conditionally apply logic or styling based on the platform. This library helps with platform specific adjustments.
* import { …, ScrollView, …} from “react-native”;
* Provides vertical or horizontal scrolling for a container when its content exceeds the screen size.  
  Place your form or list inside a ScrollView so users can scroll through content—useful when using the on-screen keyboard or displaying a long list of input fields, especially on smaller screens.
* import { …, StatusBar, …} from “react-native”;
* This library allows control over the phone’s status bar appearance like background color or text color. You can set the status bar to have matching styles to your theme so it ensures your app looks integrated with the user interface. In the application this library used for adjusting the height for iOS platform.
* import { Calendar } from "react-native-calendars";
* The Calendar component is part of the react-native-calendars library. It renders a customizable and interactive calendar user interface inside your app. Users can click on the dates that triggers the functions like onDayPress to perform actions such as showing or saving notes for the day they clicked. In the project calendar has the features like highlighting the selected date, display a red dot when there is a stored information in that day and lets users easily manage summaries per day.
* import { useNavigation } from "@react-navigation/native";
* useNavigation is a React Navigation hook that provides access to the navigation object in functional components. This library allows developer to programmatically control screen transitions like going back or navigating to other pages. In this project this feature used by making a functional arrow for going back to the days page

**Figure 2.4**

* import { Ionicons } from "@expo/vector-icons";
* The Ionicons library, provides access to a large collection of vector icons from the popular Ionicons icon set.This library specially maded for React Native applications that built with Expo, making it more easy to include scalability and high quality icons for user interface. Each icon is a React component that can be styled with props like name, size, and color. In this project this library used for displaying a back arrow for navigating to days page.

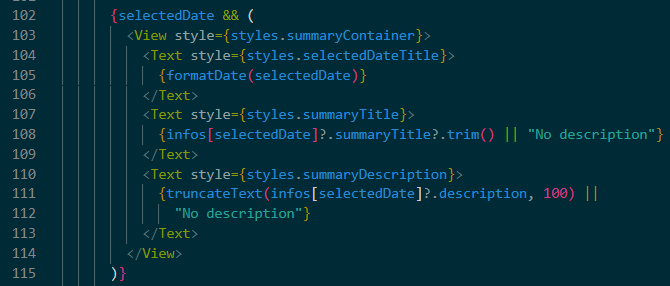
 There is main three features in this page these are calendar, description modal under the calendar and the day description modal when clicked to a day. In the first feature, Calendars code is as follows:

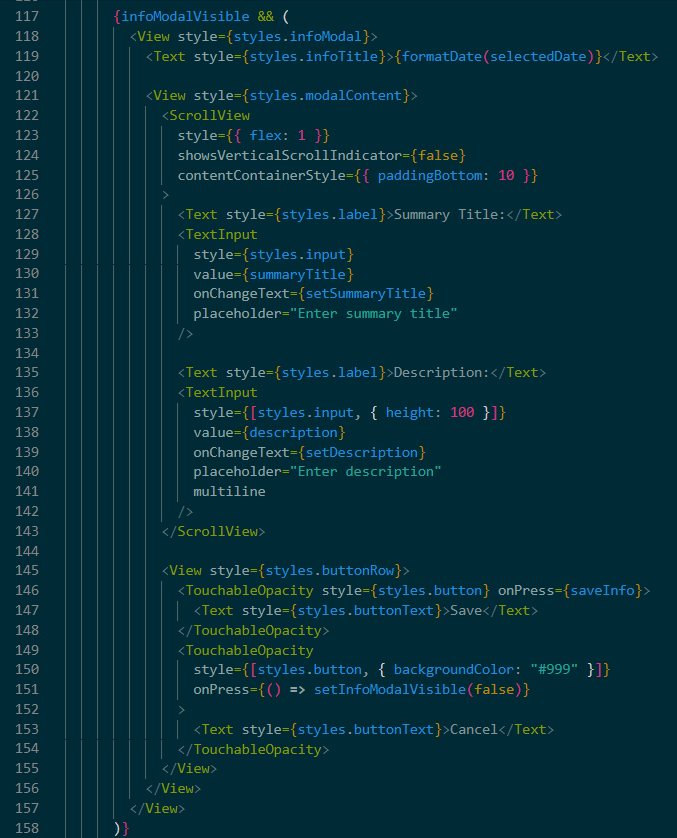
This is a simple calendar with functionality so the users can use this calendar easily and find the feature they want quickly. Code starts with integrating a calendar to the page and when user presses a day it will outline the day on calendar with #70c0ba color and if it has a description will put a dot under the day.

Second feature of this page is description modal under the calendar. This modal will only opens when the user selects a day for the first time when they enter the application so its default state is false.

This code starts with selecting a day, when a day is selected by theuser for the first time they entered the application the table will be shown to user. The structure of this table is on top of the modal the title with dd/mm/yyyy day name format and under it summary description and detailed description for the selected day. This table is just a description for the selected day and user can not interract with this modal. If the day has no description the summary description or the detailed description area will show the no description text invidually.

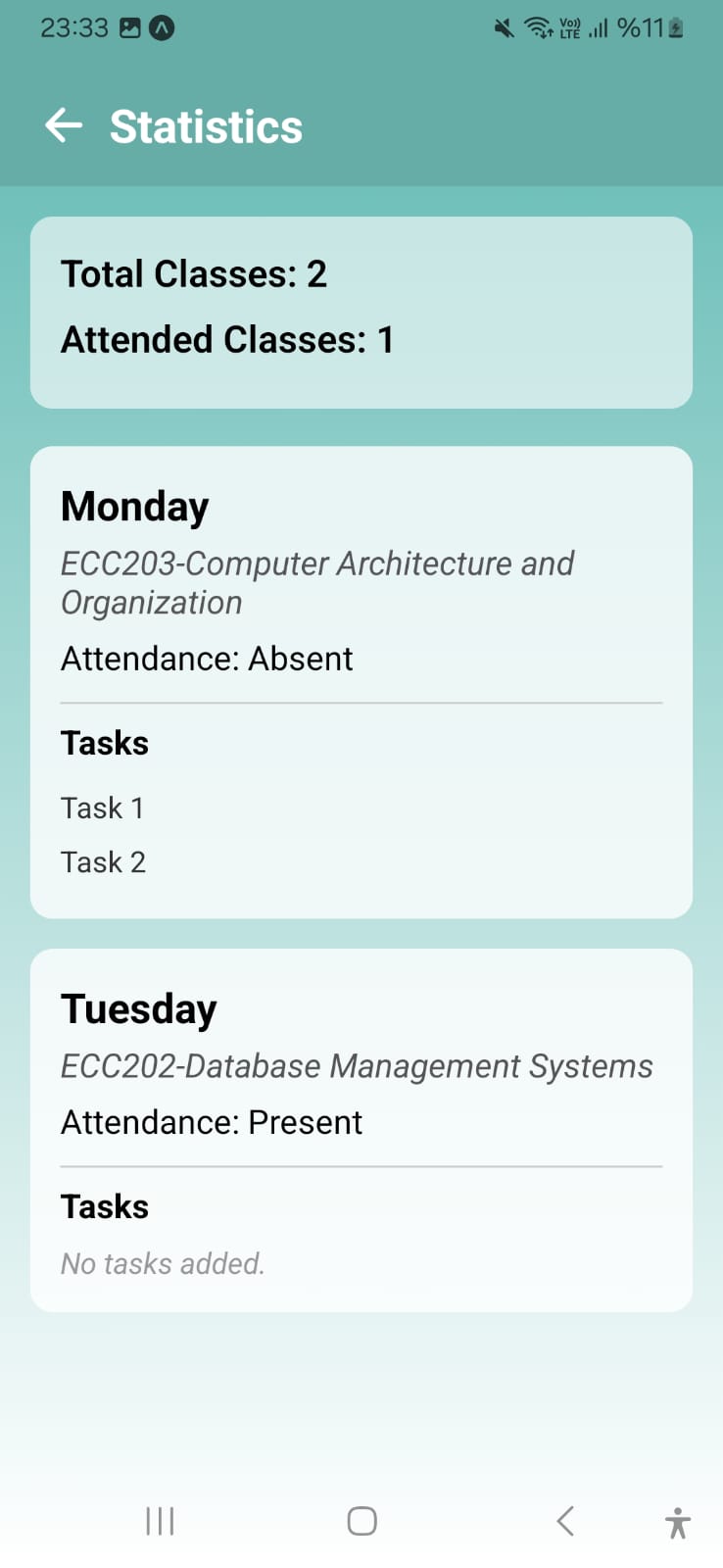
Third Feature is the day description modal that opens whenever a day is selected by the user. This day description modal has the same structure as the description modal under the calendar but user can interract with this description area.

Its code starts with the taking all components into a container so the components will be in order. At the top of the day description modal is the title of the selected day as the dd/mm/yyyy day name format, then follows the summary description area interactable by user for taking notes for the day and a detailed description area that users can write all the details they want. The users can scroll this description are because of pereventing the overflow when description the user enters will be long. After these description areas at the bottom there is two butttons as Save and cancel. The save button will save the data that user writed on the summary description and detailed description areas but if user presses the cancel button the modal will close without saving the any changes that user writed.



**Figure 2.5**

**CHAPTER THREE: STATİSTİCS PAGE**

Another page user can navigate through the days page is statistics page. In this page Students can see total number of classes they have in days page and how many classes they attended between those classes as a table. Under that table will show the all the day informations days page currently has. Days page and Statistics page works together wile sharing data to one another. Where Days page takes the data stores it somewhere else and statistics page takes the data from that storage and creates the table and current days.

**Figure 3.1**

The structure of the page is as shown in the figure, a Statistics title on top of the screen a go back button as an icon. A table for users to see their total number of classes at the days page at the current time. Under this table each day has its area for details, these areas has the structure of title top on top of the area course name, if the student is absent or present this is the data from toggle attendance button and lastly all tasks for that class. If there is a task area but user didnt write anything in it this table will show the text no tasks added. Users cant interact with this page if uders want to change any information on this screen they need to do it on the days page and save that changes, statistics page immediatly will apply the changes.

There is two main features of this page they are Total classes table and invidual class modal. Firstly if we will look into the Total classes table in the code, we can see it is inside the navigation bar so it is a part of the navigation bar. The table has its own space inside the view component and has two parameters as:

1. Total Classes
2. Attended Classes

 These parameters come from the store.js file, calculating in the statistics page and then wirtten on this parameter. Second parameter is for the attendance toggle button. When the user toggled the attendance button on the Days page, days page sends data to store file and statistics page takes that data from the storage as the second parameter in this table.

**Figure 3.2**

In this part of the code we can see a function for creating invidual table for the each day on the days page. Firstly code opens a space for the invidual day table and then starts by taking the informations from the store file as respectively, Day name, Course name, attendance check if student toggled the button or didnt and lastly all the tasks. If the task in the days information is empty it will show the text of No tasks added, if the days information is not empty it will map through the task array and display each task inside its own styled row. Although it currently only displays data, this component could be easily extended to support interactivity.

**Figure 3.3**

**CHAPTER FOUR: STORAGE FILE**

This is the file that is the brigde between Days page and Statistics page. Days page takes data from the user and stores that data in here, then the statistics page takes this data and creates the corresponding tables for the stored data. This project uses a global state store for a course timetable app using Zustand, a lightweight state management library for React. Zustand is a state management tool for React that allows for easy creation of global state stores. The create() function from Zustand is used to define the store. The function takes a callback that receives set and get functions:

* set: Updates the state.
* get: Retrieves the current state.

**Figure 4.1**

**CONCLUSION**

In the chapter one the outcomes are three functional features as Adding day, looking and editing the days information and a navigation bar for navigating through other pages, the app sends notification to users device 2 hours before the classes but it is not stable right now the project must be transferred to an development build for it must be stable. Chapter two’s results gave the project the features, an interactable calendar, an information panel for the selected day on the calendar and a description modal for editing the description of the selected day on calendar. Chapter three has the part of the project that provided the information about the days and attendance, student can see their total classes and every day seperately. In the Chapter four the storage file connected the days page and statistics page. Days page took the data from the user stored here and statistics page tooke that tada from store than created tables with the data. This project can be improved with a lot of little and big features for example a better user interface in calendar page or days page, transfering project to development build so notifications run without problem and with this project transfers to the development build it can be developed further without any risks.

**REFERENCES**

<https://en.wikipedia.org/wiki/React_Native>

<https://docs.expo.dev/get-started/introduction/>

<https://en.wikipedia.org/wiki/Mobile_app_development>

<https://en.wikipedia.org/wiki/JavaScript>

<https://zustand.docs.pmnd.rs/>

<https://reactnative.dev/docs/getting-started>

<https://reactnavigation.org/docs/getting-started/>

<https://docs.expo.dev/guides/icons/>

<https://docs.expo.dev/versions/latest/sdk/linear-gradient/>

<https://docs.expo.dev/router/introduction/>

<https://callstack.github.io/react-native-paper/>

<https://docs.expo.dev/versions/latest/sdk/notifications/>

<https://reactnative.dev/docs/getting-started>

<https://docs.expo.dev/>

<https://docs.expo.dev/router/introduction/>

<https://callstack.github.io/react-native-paper/>

<https://wix.github.io/react-native-calendars/docs/intro>

<https://docs.expo.dev/guides/icons/>

<https://docs.expo.dev/versions/latest/sdk/linear-gradient/>

<https://reactnavigation.org/docs/getting-started>

<https://code.visualstudio.com/>

<https://git-scm.com/>

<https://nodejs.org/>

<https://www.npmjs.com/>

<https://reactjs.org/docs/context.html>

<https://reactjs.org/docs/hooks-intro.html>

<https://docs.expo.dev/troubleshooting/clear-cache/>