Maikel Mirzadegan

WEB530

Cross platfrom app development with react-native

Reflection

Technology:

What are the core concepts of the React Native framework (in general)?

The core concepts of the React Native framework include:

Components: Building blocks of the user interface, which can be reused and combined to create complex interfaces.

JSX: A syntax extension for JavaScript that allows you to write HTML-like code within your JavaScript files.

State: Represents the data that can change over time and is used to manage the dynamic behavior of components.

Props: Short for properties, props are used to pass data from a parent component to a child component.

Virtual DOM: A lightweight copy of the actual DOM that React uses for efficient rendering and updates.

What are the main benefits and drawbacks of implementing apps using React Native?

The main benefits of implementing apps using React Native are:

Cross-platform development: React Native allows you to write code once and deploy it on both iOS and Android platforms.

Native-like performance: React Native apps are compiled into native code, providing near-native performance.

Hot reloading: Changes made to the code can be instantly reflected in the app during development, speeding up the iteration process.

Large community and ecosystem: React Native has a thriving community and a wide range of libraries and tools available for development.

The drawbacks of React Native include:

Limited access to device-specific features: React Native provides a bridge to access native features, but some advanced or niche functionalities may require writing custom native modules.

Debugging challenges: Debugging React Native apps can be more challenging compared to web development due to the bridge between JavaScript and native code.

Performance limitations: Although React Native offers good performance, certain complex animations or heavy computations might require writing native code for optimal performance.

How can components share data via component properties (props)?

Components can share data via component properties (props) by passing data from a parent component to a child component. The parent component can pass data as props when rendering the child component, and the child component can access and use this data. Props are read-only, and changes in props trigger re-rendering of the child component.

The React Context API allows you to control the flow of data by creating a shared state that can be accessed by multiple components without passing props through all the intermediate components. It eliminates the need for prop drilling and simplifies the process of sharing data across the application.

How has your perception of React Native changed after taking this course?

After taking this course, my perception of React Native has evolved significantly. I have gained a deeper understanding of its core concepts and how to build mobile apps using React Native, Expo, and Firebase. I have developed confidence in creating cross-platform apps and utilizing the extensive React Native ecosystem. The course has reinforced my belief in the power and flexibility of React Native for mobile app development.

<u>Team Assignment:</u>

What is the theme for your app? (Briefly share highlights and the main features of your app)

The theme for our app is a fitness tracking application. The main features of our app include tracking workouts, setting goals, providing exercise recommendations, and displaying progress statistics.

Explain how your team worked together to brainstorming ideas and narrowing/choosing a specific concept?

Our team worked together by conducting brainstorming sessions where each member shared their ideas and suggestions for the app concept. We discussed the pros and cons of different ideas and narrowed down the options based on feasibility and user appeal. We considered factors such as market demand, existing competition, and the technical complexity of implementing various features.

What is MoSCoW and how did it help (or confuse) your team to develop your app features?

MoSCoW stands for Must, Should, Could, and Won't. It helped our team in developing app features by categorizing them based on their priority and importance. It provided a clear framework for prioritization and ensured that we focused on the most essential features first. However, at times, it also led to confusion when deciding between features that fell into the "Should" or "Could" category.

How did your team members code the user stories in parallel? Was it an equal split? Why or why not?

Our team members divided the coding of user stories based on our individual strengths and preferences. We aimed for a balanced distribution of workload, but it wasn't always an equal split due to the varying complexity of the user stories as well as experience on the task at hand. Some features required more time and effort to implement, while others were relatively simpler. We focused on effective communication and supporting each other to ensure a smooth development process.

When challenges arose (both technical or disagreements), how did you solve them?

When challenges arose, we approached them by having open and constructive discussions. For technical challenges, we sought help from team members with relevant expertise or conducted research together to find solutions. In the case of disagreements, we encouraged active listening, respect for different opinions, and finding compromises that aligned with the project's goals. Regular team discussions and a supportive team culture fostered by the profs helped us overcome challenges effectively.

<u>Personal Experience:</u>

List your technical contributions to the project and explain your thought-process while you worked on the app implementation?

My technical contributions to the project included implementing the authentication flow using Firebase, integrating Firebase services for data storage and retrieval, and creating reusable components for UI elements such as buttons, forms, and modals that culminated in an events page with mapping functionality and filtering based on distance, location and price. I focused on writing clean and maintainable code, ensuring proper error handling, and optimizing performance where necessary. Throughout the implementation, I constantly sought feedback from team members and incorporated their suggestions to improve the codebase.

Summarize your personal experience of participating in the course. What did you enjoy or dislike?

Participating in this course has been an enriching experience. I particularly enjoyed the hands-on approach of building mobile apps using React Native, Expo, and Firebase. The course provided a structured learning environment with clear explanations, practical examples, and challenging labs. I appreciated the guidance and support from the instructors, who were knowledgeable and responsive to our questions. The opportunity to work in a team allowed me to enhance my collaboration skills and experience the dynamics of real-world app development. Overall, I found the course engaging, valuable, and a significant step in my journey as a softwarer developer to have practical mobile app development experience.