

Cross Platform App Development Lab Experiment No.9

Aim: Exploration of life cycle functions for components, hook functions, and XAML page.

Objectives:

1. Understand the life cycle functions of components in software development.
2. Explore hook functions and their role in component-based architectures.
3. Gain insights into the life cycle of XAML pages in the context of UI development.

Theory:

- **Life Cycle Functions for Components:** Components go through various stages during their life cycle, such as initialization, mounting, updating, and unmounting. Understanding these stages is crucial for efficient development and optimization.

- **Hook Functions:** Hook functions are special functions that developers can use to perform actions at specific points in a component's life cycle. Common hooks include `'componentDidMount'`, `'componentDidUpdate'`, and `'componentWillUnmount'` in React.

- **XAML Pages Life Cycle:** XAML (eXtensible Application Markup Language) pages, commonly used in UI development, have their own life cycle. This involves loading, initialization, rendering, and disposal of the page.

- There are four main components and hooks in React Native:

useEffect: This hook allows you to perform side effects in functional components, such as fetching data or updating the DOM.

useContext: This hook allows you to access context values in functional components.

useReducer: This hook allows you to manage state in a more complex way than using `useState`.

useState: This hook allows you to add state to functional components.

Requirements:

1. Development environment for the chosen framework (e.g., React for components, Xamarin for XAML).
2. Code editor (e.g., Visual Studio Code, Visual Studio).
3. Basic understanding of the chosen framework and language.

Tools:

1. React for component-based development.
2. Xamarin for XAML pages.
3. Code editor of choice.

Implementation/Code:


- Using useEffect hook to implement authentication on our application


useEffect: This hook allows you to perform side effects in functional components, such as fetching data or updating the DOM.


```
useEffect(() => {  
  setLoading(true);  
  const unsubscribe = auth.onAuthStateChanged((authUser) => {  
    if(!authUser){  
      setLoading(false);  
      if(authUser){  
        navigation.replace("Home");  
      }  
    });  
    return unsubscribe;  
  },[])
```

Register

Create a new Account





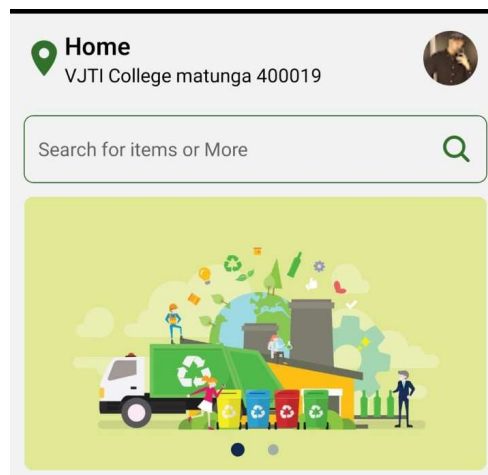


Register

Already have a account? Sign in

- Using useState hook to display **Current Location**.
- **useState**: This hook allows you to add state to functional components.

```
const [displayCurrentAddress, setdisplayCurrentAddress] = useState(  
  "VJTI College matunga 400019"  
);  
const [locationServicesEnabled, setLocationServicesEnabled] =  
useState(false);  
useEffect(() => {  
  checkIfLocationEnabled();  
  getCurrentLocation();  
}, []);
```



Conclusion:

We learned about components, hook functions, and XAML pages is essential for building efficient applications. Proper utilization of life cycle events and hooks can lead to optimized resource management and improved user experiences.

References:

1. React Documentation:

<https://reactjs.org/docs/react-component.html>(<https://reactjs.org/docs/react-component.html>)

2. Xamarin.Forms Page Life Cycle:

<https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/navigation/page-lifecycle>(<https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/navigation/page-lifecycle>)