**Summary  
- Components  
- Data Binding  
- Style Binding  
- Class Binding  
- Event Binding  
- Forms                                  
- Form Validations  
  
                      Component Life Cycle Hooks**

**- High Level Architecture for React Application**

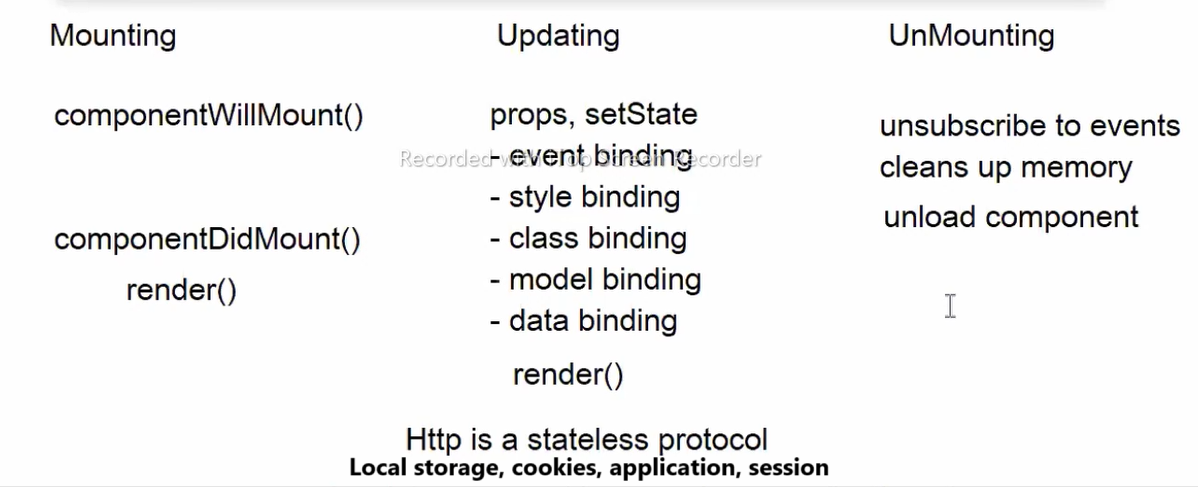
**- In Every frontend technology have these functionality like model binding, routing, cashing, State management.**

**- Every web developer should know for executing the application we have,**

**Compiler   
Routing Engine : Mapping to path to corresponding component. Browser Engine View Engine : Converts server side to HTML. - All these are managed by one compiler babel.**

**- Component Life Cycle Starts when the component is created and loaded by application.  
- Component life cycle comprises of 3 phases  
        a) Mounting  
        b) Updating  
        c) UnMounting  
  
- Mounting configures the actions to perform at the time of loading component.  
        a) componentWillMount()        allocates memory  
        b) componentDidMount()         renders output  
  
- Updating configure the actions to perform on "Change Detection", which include event, style, data, class binding techniques.  
  
        a) setState()  
        b) forceUpdate() : It means event binding. If we will not trigger**

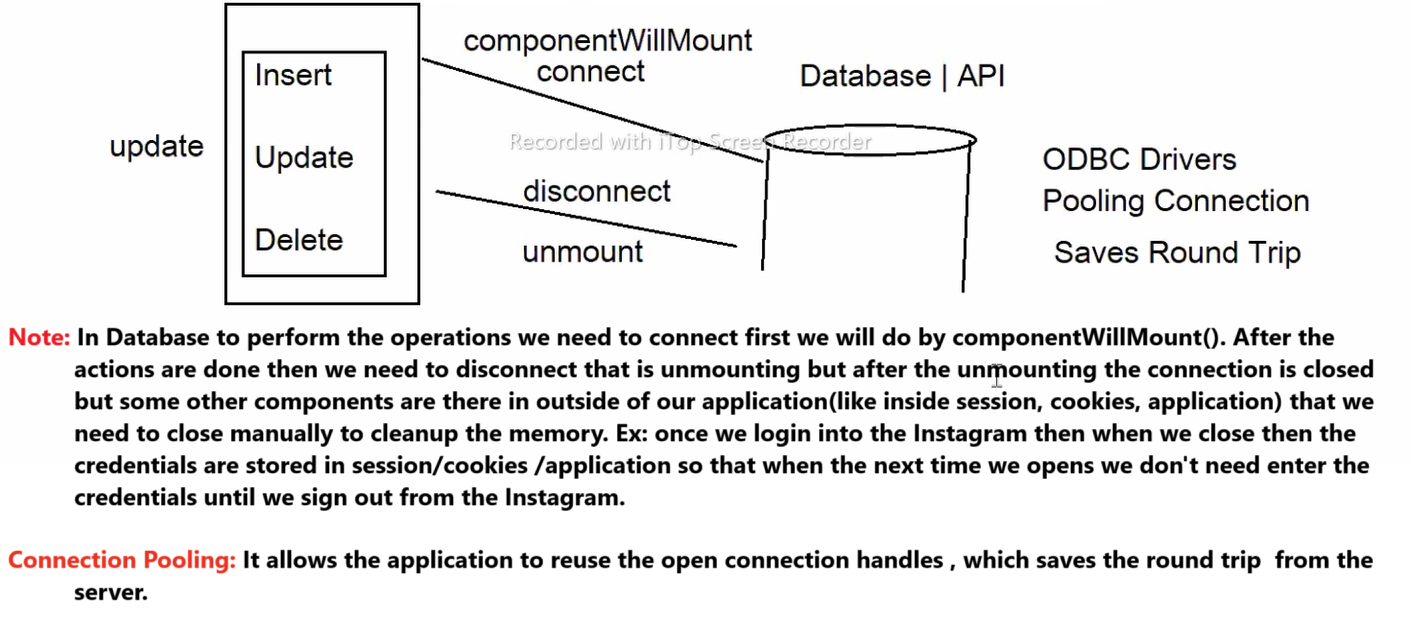
**the event then also it will work in background i.e Fromik Yup.  
        c) getProps()  
                ...fomik.getFieldProps("Name")  
        d) componentDidMount()  
  
- Unmounting configures the actions to perform when a component is unloaded and another component is requested. It includes  
        - unsubscribe to events      
        - clean up the memory  
        - unload, loads the next component.  
  
        a) componentWillUnMount()**

 **- In React upto 16 version these hooks are provided with class component.  
- React 16+ version provides life cycle hooks for components.**

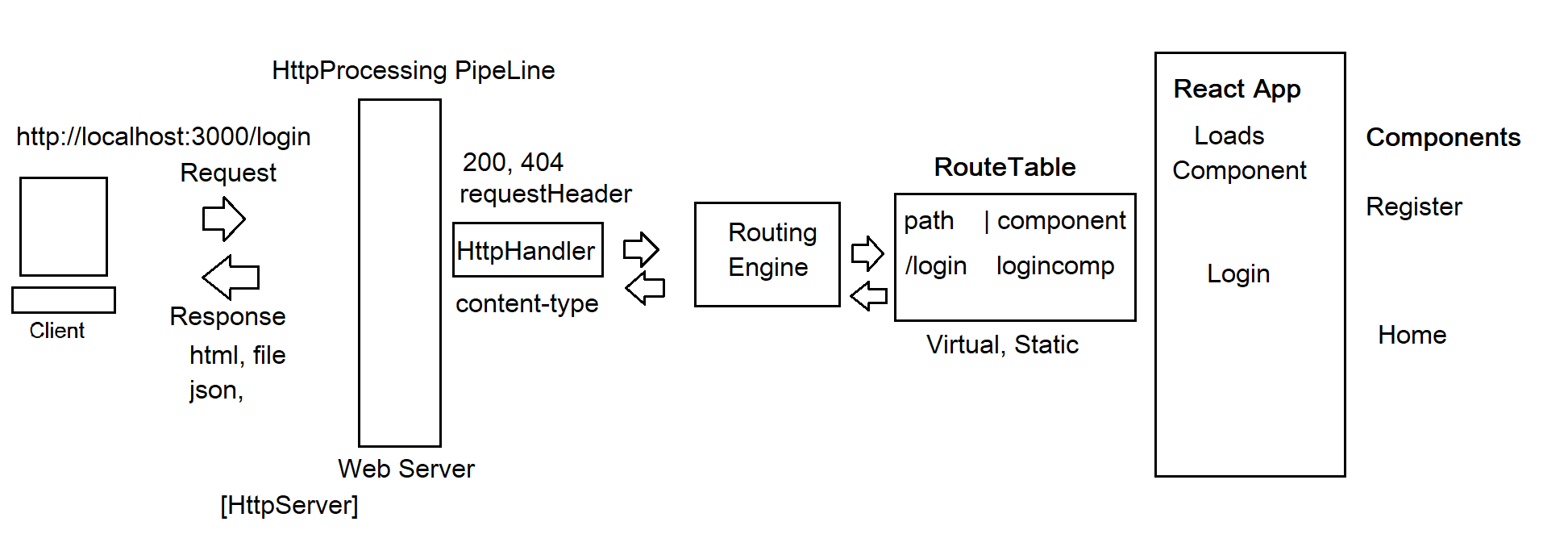
**FAQ: How component unmount and mount are defined in function component?  
Ans:   useEffect()  
  
        useEffect(()=> {  
            // defines actions on mount  
             return(()=>{  
                   // define action on unmount  
            })  
        },[])**

**FAQ: (For Experienced Level)**

**In React component lifecycle hooks mounting and unmounting is there and they are performing their actions but what the developers do for that. Explain one Scenario.**

**Ans:** 

**Ex:  
*import React from "react";  
  
class SuccessComponent extends React.Component  
{  
    componentDidMount(){  
        alert('Success Component will render');  
    }  
    componentWillUnmount(){  
        alert('Success Component will unmount');  
    }  
    render(){  
        return(  
            <div>  
                <h2>Login Success..</h2>  
            </div>  
        )  
    }  
}  
class ErrorComponent extends React.Component  
{  
    componentDidMount(){  
        alert('Error Component will render');  
    }  
    componentWillUnmount(){  
        alert('Error Component will unmount');  
    }  
    render(){  
        return(  
            <div>  
                <h2>Invalid Login</h2>  
            </div>  
        )  
    }  
}  
  
export default class LifeCycleDemo extends React.Component  
{  
    constructor(props){  
        super(props);  
        this.state = {  
            userDetails: {  
                UserName: 'john\_nit',  
                Password: 'john@11'  
            },  
            formDetails: {  
                UserName: '',  
                Password: ''  
            },  
            result: ''  
        }  
        this.handleUserName = this.handleUserName.bind(this);  
        this.handlePassword = this.handlePassword.bind(this);  
        this.handleLoginClick = this.handleLoginClick.bind(this);  
    }  
  
    handleUserName(event){  
        this.setState({  
            formDetails: {  
                UserName: event.target.value,  
                Password: this.state.formDetails.Password  
            }  
        })  
    }  
    handlePassword(event){  
        this.setState({  
            formDetails: {  
                UserName: this.state.formDetails.UserName,  
                Password: event.target.value  
            }  
        })  
    }  
    handleLoginClick(){  
        if(this.state.formDetails.UserName==this.state.userDetails.UserName && this.state.formDetails.Password==this.state.userDetails.Password) {  
            this.setState({  
                result: <SuccessComponent />  
            })  
        } else {  
            this.setState({  
                result: <ErrorComponent />  
            })  
        }  
    }  
  
    render(){  
        return(  
            <div className="container">  
                <dl>  
                    <dt>User Name</dt>  
                    <dd><input onChange={this.handleUserName} type="text"/></dd>  
                    <dt>Password</dt>  
                    <dd><input onChange={this.handlePassword} type="password"/></dd>  
                </dl>  
                <button onClick={this.handleLoginClick}>Login</button>  
                <div>  
                    {this.state.result}  
                </div>  
            </div>  
        )  
    }  
}***

****