React Native Auth & SDLC

SD550 – Web Application Development 2

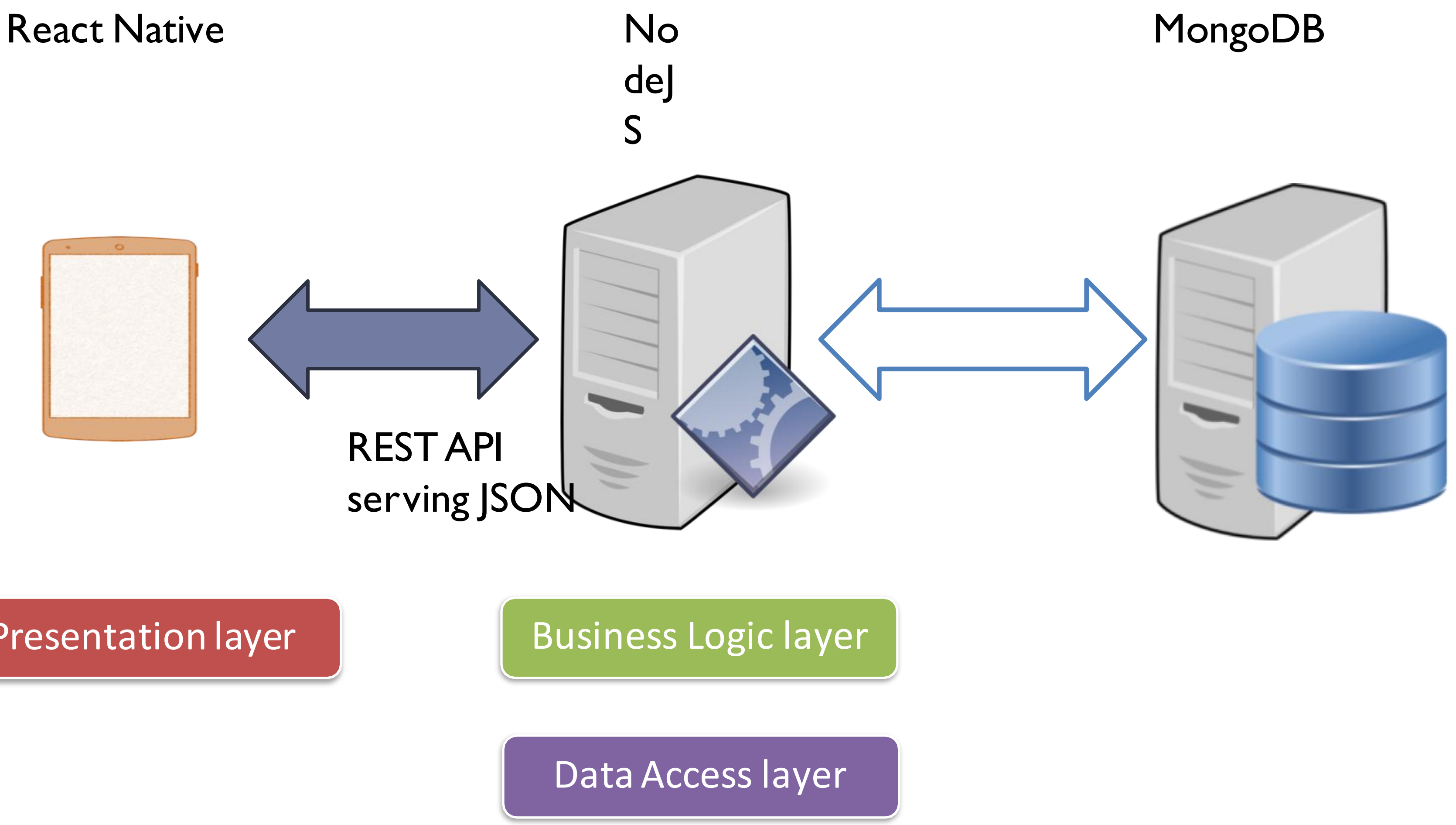
Maharishi International University

Department of Computer Science

M.S. Thao Huy Vu

Maharishi International University - Fairfield, Iowa

Modern JavaScript Architecture



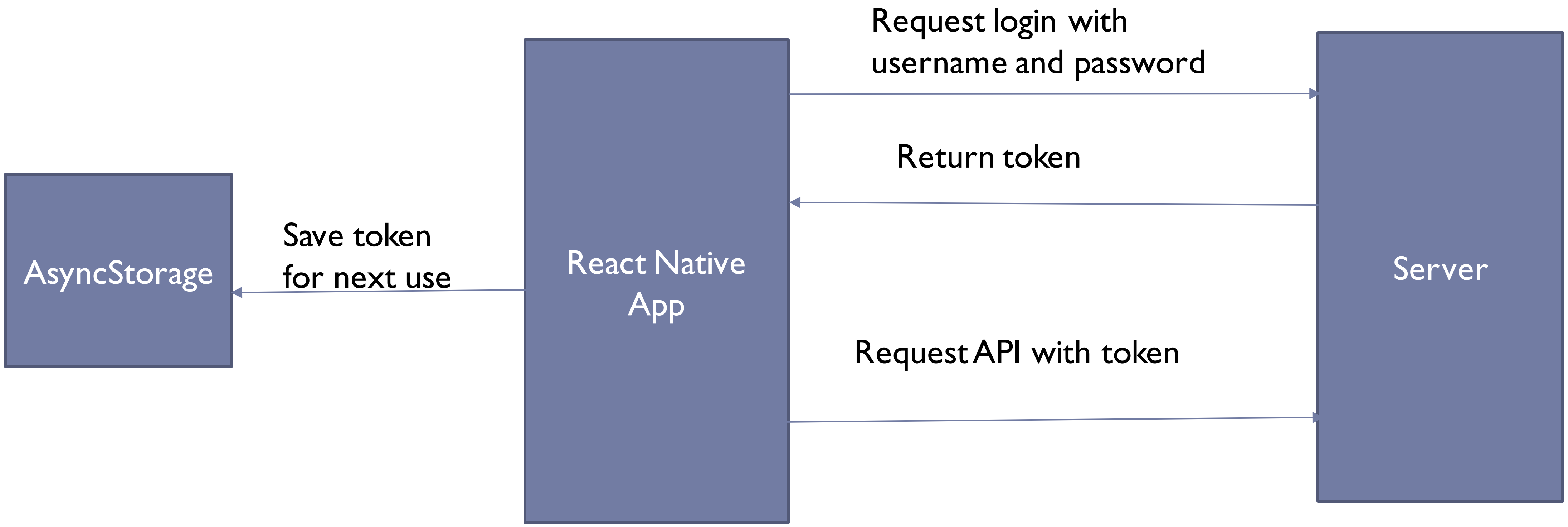
Authentication Flows

Most apps require that a user authenticate in some way to have access to data associated with a user or other private content.

- The user opens the app.
- The app loads some authentication state from persistent storage (*AsyncStorage*).
- When the state has loaded, the user is presented with either authentication screens or the main app, depending on whether valid authentication state was loaded.
- When the user signs out, we clear the authentication state and send them back to authentication screens.

https://reactnavigation.org/docs/auth-flow/

Auth for React Native apps



React Native Networking

```
fetch('https://mywebsite.com/endpoint/',  {
  method: 'POST',//PUT, DELETE, PATCH
  headers: {
    Accept: 'application/json',
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({
    firstParam: 'yourValue',
    secondParam:  'yourOtherValue',
  }),
});
```

Reference: https://reactnative.dev/docs/network

React Native Networking

Login function

```
axios.defaults.baseURL  = "http://localhost:5000";
export async function login(email: string){
  try {
    const res = await axios.post(`/users/login`,  {email});
    if(res.status === 200 && res.data.success  === true){
      return res.data.data;
    }
  } catch (error) {

  }
  return null;
}
```

React Native Networking

Get Products

```
export async function getProducts(token: string){
  try {
    axios.defaults.headers.common['Authorization'] = `Bearer ${token}`;
    const res = await axios.get(`/products`);
    if(res.status === 200 && res.data.success === true){
      return res.data.data;
    }
    return [];
  } catch (error) {
    return [];
  }
}
```

Define App State

```
const [state, dispatch] = React.useReducer((prevState: any, action: any) => {
switch (action.type) {
   case 'RESTORE_TOKEN': return {...prevState, userToken: action.token, isLoading: false};
   case 'SIGN_IN': return {...prevState, userToken: action.token, isLoading: false};
   case 'SIGN_OUT': return {...prevState, userToken: null};
}
}, {
      isLoading: true,
      userToken: null,
});

// Same as

const [state, setState] = React.useState({isLoading: true, userToken: null});
```

Conditionally Define Screens

```
state.userToken ? (
  <>
    <Stack.Screen name="Home" component={HomeScreen} />
    <Stack.Screen name="Profile" component={ProfileScreen} />
  </>
) : (
  <>
    <Stack.Screen name="SignIn" component={SignInScreen} />
    <Stack.Screen name="SignUp" component={SignUpScreen} />
    <Stack.Screen name="ResetPassword" component={ResetPassword} />
</>
);
```

Restore the Token

```
React.useEffect(~~async~~() => {
     (async () => {
             // 2. try to read userToken from storage
             let userToken = await AsyncStorage.getItem('userToken');
             // 3. change state
             dispatch({ type: 'RESTORE_TOKEN', token: userToken });
             // OR setState(prev=>({…prev, token: userToken}));
     })();
}, []); // 1. in App.js on mount.
```

Provide Helpers

```
import React, { createContext } from "react";
const GlobalContext = createContext();
export default GlobalContext;
const ACTIONS = {
  SIGN_IN: "SIGN IN",
  SIGN_OUT: "SIGN OUT"
}
function reducer(state: any, action: any){
  switch(action.type){
  case ACTIONS.SIGN_IN:
    return {...state, token: action.payload};
  case ACTIONS.SIGN_OUT:
    return {...state, token: null};
  default:
    return state;
 }
};
export {reducer, ACTIONS};
```

Provide State Helpers with Context

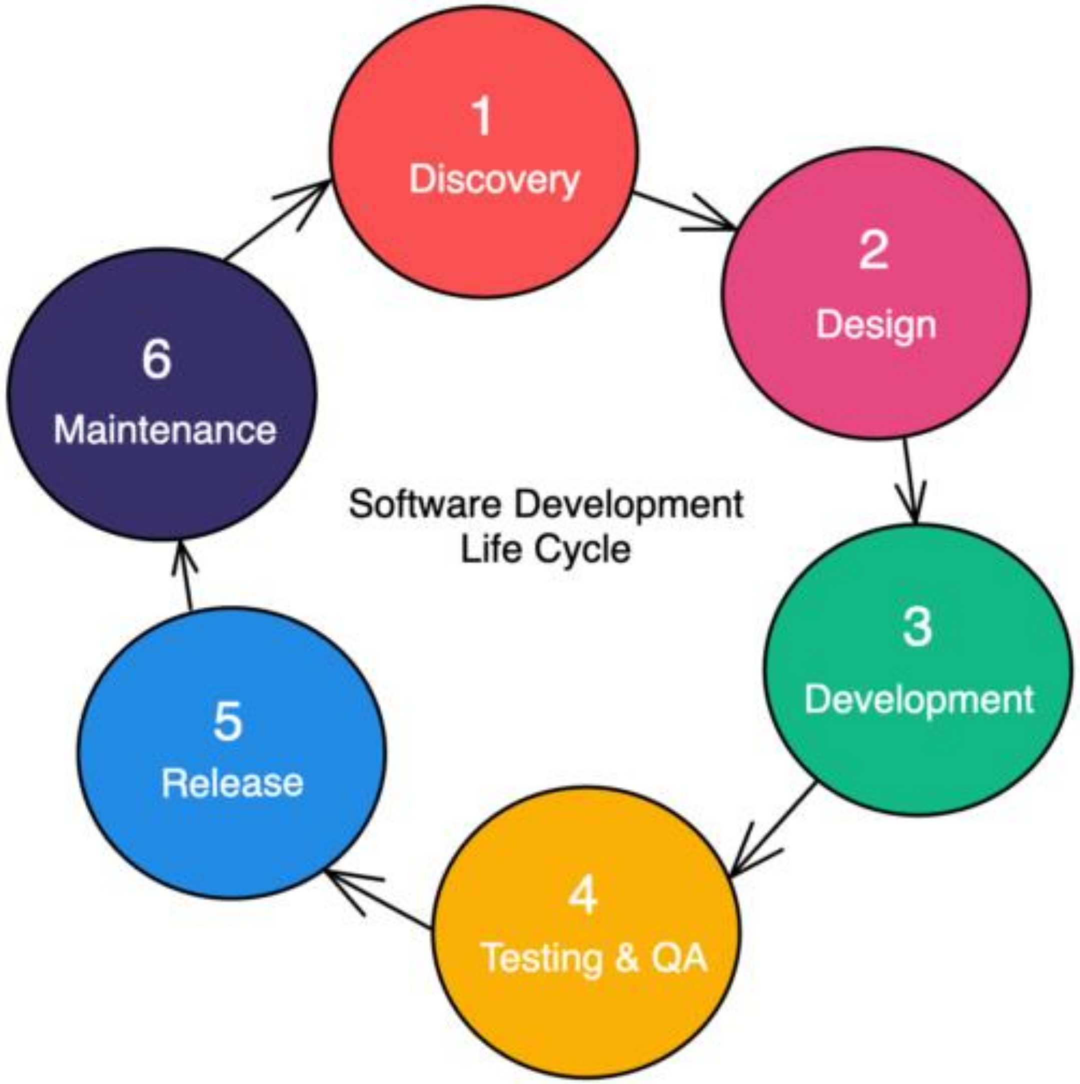
```
import GlobalContext, {reducer} from "./Helper";
import {useReducer, useState} from 'react';

export default function App() {
const [state, dispatch] = useReducer(reducer, {})
return (
  <GlobalContext.Provider value={{ state, dispatch }}>
  <View style={styles.container}>
  </View>
  </GlobalContext.Provider>
);
}
```

Software Development Process

Requirements –> Analysis –> Design –> Implementation –> Testing –> Deployment, Operation & Maintenance

Software Development Life-cycle

- Typically, a Software Development project will go through following activities in its lifecycle:



Software Development Methodology

- Software Development methodology is the process that a project team follows in creating software product. The methodology a team adopts/practices is what determines how the various software development life-cycle activities will be performed/executed.
- The following methodologies can be used:
  - **Agile: Scrum**, Extreme Programming (XP) etc.
  - Rational Unified Process (RUP)
  - Waterfall
  - Kanban, Lean

Software Requirements Discovery

Fairfield is a small and beautiful town in Iowa, home to a library that has been managed manually in many years. To embrace the new technology and streamline the management, the library has decided to invest in the development of a software system. This system is used by the librarians and managers and is designed to maintain information of all books, authors, publishers, catalogs, and members. Each book entry contains title, genre, and category. Each book may have multiple authors. Information for authors includes name, phone, and email. Each book is produced by a publisher that is recorded with name, phone, email, and address. To manage books, this library uses catalogs. Each catalog contains the number of copies of a book, and how many are available for borrowing. The system will enable searches by book title, author’s name, or publisher’s name. It also maintains the information of all members, who must be residents of this town. Member records include the first name, last name, ID, address, phone, and email. Librarians can use this system to facilitate book borrowing and returns for members.

Discovery

- What are the entities here?
    - Book
    - Author
    - Publisher
    - Catalog
    - Catalog
- What are the functionalities?
    - CRUD for entities
    - Search
    - Borrow a book
    - Return a book

Design

- Book: bookId, title, genre, category, authorIDs, publisherId
- Author: authorId, name, phone, email
- Publisher: publisherId, name, phone, email, address
- Catalog: catalogId, bookId, numberOfCopies, availableCopies
- Member: memberId, ID, firstname, lastname, address, phone, email
- Transaction: bookId, memberId, borrowedDate, returnedDate

Implementation

- Developers will develop the application
    - Backend
    - Database
    - Frontend

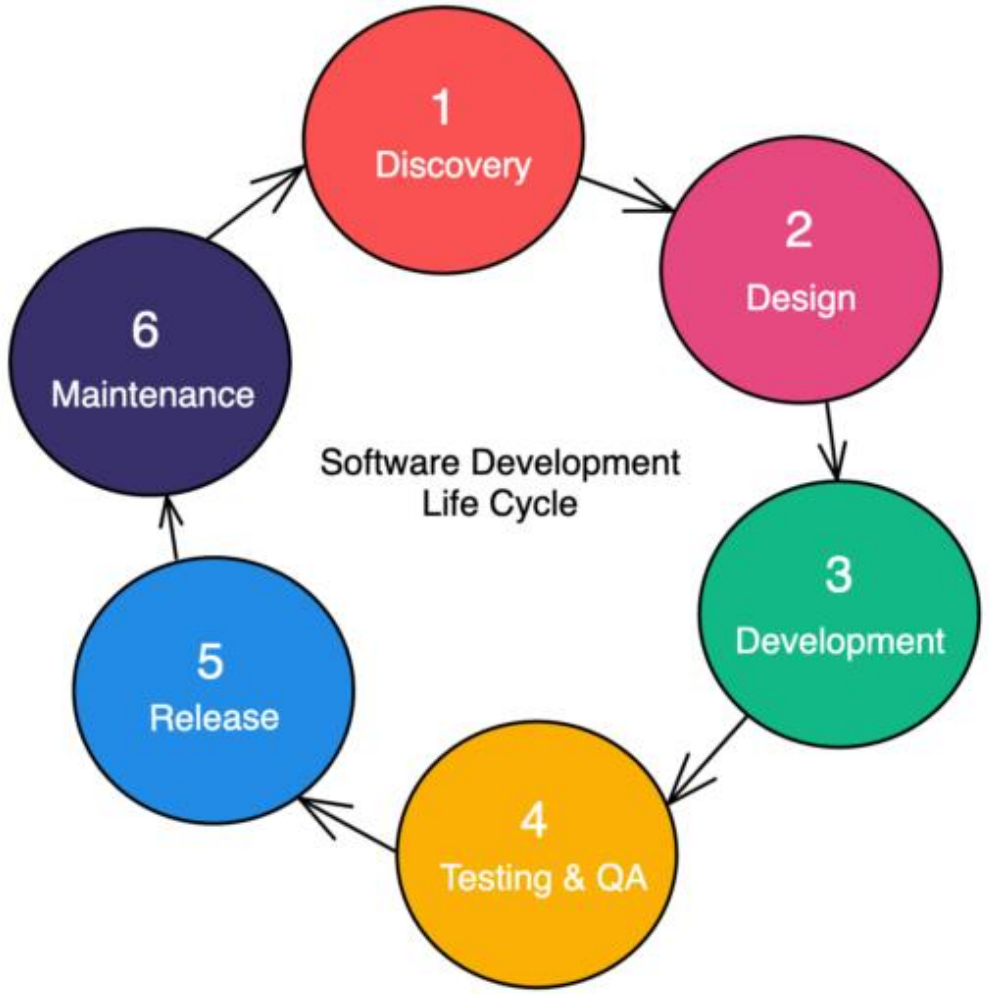
Testing

- QA will test the software before releasing

Release

- After testing successfully, software will be released to end users

Software Development Process: Loop


1
Discovery
2
Design
3
Development
4
Testing & QA
5
Release
6
Maintenance
Software Development
Life Cycle