

Full Stack Application Development

Application Development- Introduction

Akshaya Ganesan





Introduction

- This course aims to provide a comprehensive introduction to modern Web application architecture approaches, frontend and backend technologies, and suitable web application frameworks required for developing modern web apps.

Learning Outcomes

Understand the underlying architecture used for Web applications and identify the various components of the Web Application

Demonstrate the creation of API to accomplish various backend functionalities of an application like database interaction, handling user requests

Design and develop user-friendly and interactive web frontends.

Implement a functional end-to-end web application using client-side and server-side web technologies.



Modular Structure

Module 1: Application Development

Introduction to the Various Application Landscape

Module 2: Understanding the Web Basics

Structure of web applications, Client – Server Communication

Module 3: Web Protocols

HTTP, HTTPS, WebSockets

Module 4: Server Side: Implementing Web Services

REST, gRPC, GraphQL

Module 5: Securing Application

JWT, OAuth





Modular Structure

Module 6: Understanding Frontend Development

Implementing Single Page Applications using JavaScript Tech stack

Module 7: Testing

API Testing, Unit Testing

Module 8: Accessibility and Performance

Inclusive Design , Assistive Technologies, Tools and Metrics for Measuring Performance

Module 9: Latest Advancements

Progressive Web Apps

Web Assembly





Development Environment

- Tools/Technologies
 - Visual Studio Code
 - Browser(Chrome)
 - Frontend: ReactJS, Node Ecosystem
 - Backend: NodeJS, Express, Django
 - Database: MongoDB/Postgresql
 - Postman for Testing
 - GitHub and GIT

Module 1.1: Application Introduction





Activity 1

- Open Makemytrip in a browser
- Open the BITS Website in a browser
- **Observe the difference in the address bar!!**
- Repeat in a different browser
- Install the app from one browser and observe



Website

- A group of interlinked web pages having a single domain name
 - Hosted on a web server
 - Accessible over the web with an internet connection
 - Easily accessible through browsers
 - Can be developed and maintained by individuals/teams for personal or business usage
- Use Cases for Static Websites
 - portfolios
 - personal blogs
 - informational websites and
 - small business websites with minimal content updates.
- They are also suitable for landing pages or temporary promotions where the content doesn't change frequently.



Web Application

- A web application is an application program stored on a remote server and delivered over the internet.
- Users can access a web application through a web browser, such as Google Chrome, Mozilla Firefox or Safari.
- Common web applications include e-commerce shops, webmail, and social networking sites.



Application Types

- There are three basic types of mobile apps based on the technology used to develop the:
- **Native apps** are created for one specific platform or operating system.
- **Web apps** are responsive versions of websites that can work on any mobile device or OS because they're delivered using a browser.
- **Hybrid apps** are combinations of both native and web apps, but wrapped within a native app, giving it the ability to have its icon or be downloaded from an app store.



Native Apps

- Developed specifically for a particular mobile device
- Installed directly onto the device itself
- Needs to be downloaded via app stores such as
 - Apple App Store
 - Google Play store, etc.
- Built for specific mobile operating system such as
 - Apple iOS
 - Android OS
- An app made for Apple iOS will not work on Android OS or Windows OS
- Need to target all major mobile operating systems
 - require more money and more effort



Native Apps

- Pros
 - Can be Used offline - faster to open and access anytime
 - Allow direct access to device hardware that is either more difficult or impossible with web apps
 - Allow the user to use device-specific hand gestures
 - Full access to all device features and APIs.
 - Gets the approval of the app store they are intended for
 - User can be assured of improved safety and security of the app



Native Apps

- Cons
 - More expensive to develop - separate app for each target platform
 - The cost of app maintenance is higher - especially if this app supports more than one mobile platform
 - Getting the app approved for the various app stores can prove to be a long and tedious process
 - Needs to download and install the updates to the apps on user's mobile device



Native Apps

- Native apps are built specifically for one platform using its native programming language and tools.
- **iOS:**
 - **Language:** Swift, Objective-C
 - **Frameworks:** UIKit, SwiftUI
 - **Tools:** Xcode
 - **Example App:** Instagram (iOS)
- **Android:**
 - **Language:** Java, Kotlin
 - **Frameworks:** Android SDK
 - **Tools:** Android Studio
 - **Example App:** WhatsApp (Android)



Web Apps

- Internet-enabled applications
 - Accessible via the device's Web browser
- Don't need to be downloaded and installed onto a mobile device
- Written as web pages in HTML and CSS with interactive parts in JQuery, JavaScript, etc.
- Single web app can be used on most devices capable of surfing the web irrespective of the operating system



Web Apps

- Pros
 - Instantly accessible to users via a browser
 - Easier to update or maintain
 - Easily discoverable through search engines
 - Development is considerably more time and cost-effective than development of a native app
 - common programming languages and technologies
 - It has a much larger developer base.



Web Apps

- Cons
 - Only have limited scope as far as accessing a mobile device's features is concerned such as device-specific hand gestures, sensors, etc.
 - Many variations between web browsers and browser versions and phones
 - Challenging to develop a stable web app that runs on all devices without any issues
 - Not listed in 'App Stores'
 - Unavailable when offline, even as a basic version



Web Apps

- Web apps run in web browsers and are built using standard web technologies.
- **Languages:** HTML, CSS, JavaScript
- **Frameworks:** React, Angular, Vue.js
- **Tools:** Web browsers, developer tools
- **Example App:** Google Docs



Progressive Web Application

- A **progressive web app** (PWA) is an app that's built using web platform technologies
- Like a platform-specific app, it can be installed on the device.
- Offline and background operation
- Progressive web apps combine the best features of traditional websites and platform-specific apps



Progressive Web Application

- PWAs offer a native app-like experience on the web, including offline capabilities and installation.
- **Languages:** HTML, CSS, JavaScript
- **Frameworks:** Workbox, Lighthouse (for auditing)
- **Tools:** Service Workers, Web App Manifests



Hybrid Apps

- Hybrid apps combine elements of both native and web applications. They are built using web technologies but run inside a native container.
- **Languages:** HTML, CSS, JavaScript
- **Frameworks:** Ionic, Apache Cordova, PhoneGap
- **Tools:** Web technologies wrapped in native code



Hybrid Apps

- Part native apps, part web apps
- Like native apps,
 - available in an app store
 - can take advantage of some device features available
- Like web apps,
 - Rely on HTML, CSS , JS for browser rendering
- The heart of a hybrid mobile application is an application that is written with HTML, CSS, and JavaScript!
- Run from within a native application and its own embedded browser, which is essentially invisible to the user
 - iOS application would use the WKWebView to display the application
 - Android app would use the WebView element to do the same function.



Hybrid Apps

- Pros
 - Don't need a web browser like web apps
 - Can access to a device's internal APIs and device hardware
 - Only one codebase is needed for hybrid apps
- Cons
 - Much slower than native apps
 - With hybrid app development, dependent on a third-party platform to deploy the app's wrapper
 - Customization support is limited.



Cross-platform Application

- Cross-platform app development refers to developing software that can run on multiple devices.
- Multi-platform compatibility is a pervasively desirable trait.
- Product to be available to as many consumers as possible.



Cross-platform Application

- Cross-platform native apps use frameworks that allow development for multiple platforms from a single codebase, providing near-native performance.
- **Frameworks:** React Native, Flutter, Xamarin
- **Tools:** IDEs like Visual Studio, Android Studio
- **Example App:**
 - **React Native:** Facebook
 - **Flutter:** Google Ads
 - **Xamarin:** Microsoft Outlook



Native App vs Cross Platform

Native App Development	Cross Platform App Development
Native App Development is costly.	It is cost-effective.
Code cannot be reused	It supports code reusability. Same codebase is used across multiple platforms
Native apps are faster.	Cross Platform apps are slower than native apps



Self Reading

- Recommendation 1: <https://railsware.com/blog/native-vs-hybrid-vs-cross-platform/>
- Recommendation 2: <https://litslink.com/blog/mobile-applications-development-native-web-cross-platform>

Thank You!

