Họ và tên : Đinh Thị Thu Phương

Mã số sinh viên : 2180607908

Câu 1.

Hiện nay, có ba nền tảng phổ biến cho thiết bị di động thông minh: Android, iOS, HarmonyOS.

1. Android

Đặc điểm

- Hệ điều hành mã nguồn mở, phát triển bởi Google.
- Sử dụng giao diện dựa trên các ứng dụng và widget.
- Hỗ trợ nhiều thiết bị từ các nhà sản xuất như Samsung, Xiaomi, Oppo ,Huawei.
- Dựa trên nền tảng Linux và phát triển qua các công cụ Java, Kotlin, gần đây là Flutter (cho phát triển đa nền tảng).

Ưu điểm

- Mã nguồn mở: Cộng đồng có thể tùy chỉnh giao diện và tính năng của hệ điều hành.
- Đa dạng ứng dụng: Số lượng ứng dụng phong phú từ Google Play Store và cả các nguồn bên ngoài.
- Tính tùy chỉnh cao: Người dùng dễ dàng thay đổi giao diện và cài đặt hệ thống.
- Phù hợp với nhiều phân khúc giá: Có nhiều mức giá từ rẻ đến cao cấp.

Khuyết điểm

- Bảo mật: Cho phép cài ứng dụng ngoài Play Store, dễ gặp vấn đề bảo mật và phần mềm độc hại.

- Sự phân mảnh: Nhiều phiên bản Android khác nhau trên các thiết bị, gây khó khăn cho việc cập nhật và hỗ trợ ứng dụng.
- Hiệu suất: Một số thiết bị giá rẻ có thể giảm hiệu năng sau thời gian sử dụng do phần cứng hạn chế.

2. iOS

Đặc điểm

- Hệ điều hành độc quyền của Apple, chạy trên các thiết bị như iPhone, iPad, iPod Touch.
- Giao diện đơn giản, trực quan và nhất quán trên tất cả các thiết bị.
- Là hệ điều hành đóng, chỉ tải ứng dụng từ App Store.

Ưu điểm

- Bảo mật cao: Kiểm soát ứng dụng nghiêm ngặt, mã hóa tốt và cập nhật bảo mật nhanh chóng.
- Hiệu năng tối ưu: Tối ưu hóa cho phần cứng của Apple, đem lại hiệu năng mượt mà, ổn định.
- Hệ sinh thái Apple: Kết nối chặt chẽ với các thiết bị và dịch vụ khác như Mac, iPad, Apple Watch, tạo ra trải nghiệm liền mạch.
- Cập nhật đồng bộ: Apple có thể cập nhật iOS cho hầu hết các thiết bị ngay khi phát hành phiên bản mới.

Khuyết điểm

- Giá cao: Thiết bị iOS thường có giá thành cao, khó tiếp cận với một số phân khúc người dùng.
- Tùy chỉnh hạn chế: Không cho phép tùy chỉnh giao diện và cài đặt hệ thống nhiều như Android.

- Phụ thuộc vào hệ sinh thái Apple: Tương thích tốt nhất khi sử dụng với các thiết bị và dịch vụ khác của Apple, gây khó khăn khi muốn chuyển đổi sang hệ điều hành khác.

Câu 2.

1. Nền tảng phát triển ứng dụng di động phổ biến

1.1 Native Development (Android Studio và Xcode)

- Android Studio: Sử dụng ngôn ngữ Kotlin và Java, là công cụ chính thức để phát triển ứng dụng Android.
- Xcode: Sử dụng Swift và Objective-C, là công cụ chính thức để phát triển ứng dụng iOS.

1.2 React Native

Sử dụng JavaScript và React để phát triển ứng dụng đa nền tảng. Các ứng dụng có thể chạy trên cả iOS và Android từ một codebase duy nhất.

1.3 Flutter

Sử dụng ngôn ngữ Dart, phát triển bởi Google. Hỗ trợ xây dựng ứng dụng đa nền tảng với khả năng tùy chỉnh cao và giao diện mượt mà.

1.4 Xamarin

Sử dụng C# và .NET, phát triển bởi Microsoft. Hỗ trợ xây dựng ứng dụng đa nền tảng với codebase chung, có thể chia sẻ mã giữa iOS, Android và Windows.

1.5 Ionic

Sử dụng HTML, CSS, JavaScript (Angular, React, hoặc Vue). Các ứng dụng chạy trong WebView của thiết bị, cung cấp giải pháp phát triển đa nền tảng thông qua công nghệ web.

1.6 Unity

Chủ yếu được sử dụng cho phát triển game đa nền tảng, nhưng cũng có thể phát triển các ứng dụng di động thông thường. Sử dụng ngôn ngữ C#.

2. So sánh các nền tảng phát triển ứng dụng di động

Nền tảng	Công nghệ chính	Đối tượng hỗ trợ	Ưu điểm	Khuyết điểm
Native (Android Studio, Xcode)	Kotlin, Java, Swift, Objective-C	Android và iOS	Hiệu năng tốt nhất, tối ưu hóa tối đa cho từng hệ điều hành	Yêu cầu phát triển riêng biệt cho từng hệ điều hành
React Native	JavaScript, React	Android, iOS	Codebase chung, cộng đồng lớn, hiệu suất tốt cho ứng dụng không phức tạp	Hạn chế hiệu năng với ứng dụng yêu cầu tính năng đồ họa cao
Flutter	Dart	Android, iOS, Web, Desktop	UI mượt mà, nhiều widget tùy chỉnh, hiệu năng cao gần như native	Ngôn ngữ Dart chưa phổ biến, kích thước ứng dụng thường lớn
Xamarin	C#, .NET	Android, iOS, Windows	Tích hợp tốt với hệ sinh thái Microsoft, có thể chia sẻ mã cho ứng dụng Windows	Dung lượng ứng dụng lớn, hiệu năng không bằng native
Ionic	HTML, CSS, JavaScript	Android, iOS, Web	Dễ học cho lập trình viên web, phát triển	Hiệu suất thấp hơn do chạy trên WebView, không

Nền tảng	Công nghệ chính	Đối tượng hỗ trợ	Ưu điểm	Khuyết điểm
			nhanh, codebase chung	phù hợp ứng dụng phức tạp
Unity	C#	Android, iOS, nhiều nền tảng	Hỗ trợ phát triển game mạnh mẽ, hiệu suất đồ họa cao	Thích hợp chủ yếu cho game, không phù hợp ứng dụng không liên quan đến đồ họa

3. Sự khác biệt chính giữa các nền tảng

- **Hiệu suất**:
  - Native (Android Studio, Xcode) có hiệu suất tốt nhất vì ứng dụng được xây dựng trực tiếp cho từng hệ điều hành.
  - Flutter và React Native có hiệu suất cao với khả năng tối ưu hóa gần như native nhưng vẫn có giới hạn, nhất là trong các ứng dụng phức tạp về đồ họa.
  - Ionic và Unity có hiệu suất thấp hơn khi dùng WebView hoặc chạy game phức tạp.
- **Khả năng tùy chỉnh giao diện (UI)**:
  - Flutter cung cấp nhiều widget và công cụ tùy chỉnh giao diện, giúp tạo UI đẹp và mượt mà.
  - Native và React Native cũng có khả năng tùy chỉnh UI tốt nhưng đòi hỏi kiến thức sâu hơn về từng nền tảng.
- **Dễ học và cộng đồng hỗ trợ**:
  - React Native và Ionic dễ tiếp cận đối với lập trình viên web do sử dụng JavaScript, HTML, và CSS.

    - Xamarin và Flutter yêu cầu học các ngôn ngữ mới như C# hoặc Dart.
- **Khả năng mở rộng và hỗ trợ đa nền tảng**:
    - Flutter, React Native và Xamarin hỗ trợ đa nền tảng tốt cho cả Android và iOS từ một codebase duy nhất.
    - Native chỉ phát triển trên một nền tảng nên cần nhiều nỗ lực hơn khi mở rộng sang nền tảng khác.
- **Đối tượng ứng dụng phù hợp**:
    - Native thích hợp cho các ứng dụng yêu cầu hiệu suất cao, phức tạp về đồ họa.
    - Flutter và React Native phù hợp cho ứng dụng đa nền tảng cần UI mượt mà.
    - Ionic và Xamarin phù hợp cho các ứng dụng đơn giản hoặc có tính năng cơ bản.
    - Unity lý tưởng cho các ứng dụng hoặc game phức tạp về đồ họa.

Câu 3.

Flutter đã trở thành một lựa chọn phổ biến trong phát triển ứng dụng đa nền tảng nhờ một số đặc điểm nổi bật sau đây, đặc biệt khi so sánh với các nền tảng như React Native và Xamarin:

1. Hiệu suất cao và gần giống Native

- **Flutter**: Flutter sử dụng bộ công cụ đồ họa Skia, giúp tạo ra các giao diện trực tiếp mà không cần đến các lớp cầu nối (bridges) giữa ngôn ngữ và native code. Nhờ đó, ứng dụng Flutter có hiệu suất gần như ứng dụng native, mang lại trải nghiệm mượt mà, phản hồi nhanh.
- **React Native**: Dù hiệu năng khá cao, React Native phụ thuộc vào các cầu nối (bridges) giữa JavaScript và native code, nên trong một số trường hợp (như các ứng dụng có hoạt động đồ họa phức tạp), tốc độ phản hồi có thể chậm hơn Flutter.

- **Xamarin**: Xamarin cũng cho phép tạo các ứng dụng gần như native nhờ biên dịch mã C# thành mã native. Tuy nhiên, hiệu năng đôi khi không bằng Flutter, và kích thước ứng dụng khi phát triển với Xamarin thường lớn hơn Flutter.

2. Thư viện widget phong phú và dễ tùy chỉnh

- **Flutter**: Cung cấp sẵn một bộ thư viện widget phong phú và được thiết kế tối ưu cho cả Android lẫn iOS, giúp việc tùy chỉnh giao diện trở nên đơn giản và đồng nhất. Flutter không phụ thuộc vào các thành phần giao diện của hệ điều hành mà tạo các widget tùy chỉnh từ đầu, cho phép giao diện giống nhau trên mọi thiết bị.
- **React Native**: Sử dụng các component native của hệ điều hành, nên khả năng tùy chỉnh phụ thuộc vào phiên bản hệ điều hành, có thể gây ra một số lỗi tương thích. Để tùy chỉnh cao hơn, cần phải dùng các thư viện hoặc module của bên thứ ba.
- **Xamarin**: Dù cũng cung cấp các component cho Android và iOS, nhưng khả năng tùy chỉnh không phong phú và linh hoạt như Flutter. Xamarin Forms giúp dễ dàng tạo giao diện nhưng có thể thiếu tính nhất quán khi triển khai trên nhiều hệ điều hành khác nhau.

3. Cộng đồng hỗ trợ và thư viện mở rộng

- **Flutter**: Được hỗ trợ mạnh mẽ từ Google, Flutter có một cộng đồng đang phát triển nhanh chóng với nhiều thư viện hỗ trợ, package mở rộng phong phú và tài liệu cập nhật. Google đã và đang đẩy mạnh sử dụng Flutter cho các sản phẩm nội bộ của họ, tạo thêm động lực cho cộng đồng.
- **React Native**: Được Facebook phát triển và có một cộng đồng rất lớn, giúp cho việc tìm kiếm tài liệu, hỗ trợ và thư viện mở rộng dễ dàng. Tuy nhiên, React Native cũng có các thư viện mở rộng ít được cập nhật hoặc không chính thức.
- **Xamarin**: Dù có cộng đồng ổn định và sự hỗ trợ từ Microsoft, Xamarin không có sự phát triển mạnh mẽ như Flutter hay React Native. Tài liệu và

thư viện mở rộng không đa dạng bằng hai nền tảng kia, mặc dù vẫn được hỗ trợ tốt trong hệ sinh thái của Microsoft.

4. Ngôn ngữ phát triển

- **Flutter**: Sử dụng ngôn ngữ Dart, được tối ưu hóa cho phát triển ứng dụng UI. Mặc dù Dart không phổ biến bằng JavaScript, nhưng cú pháp dễ học và tối ưu cho Flutter, giúp lập trình viên làm quen nhanh chóng.
- **React Native**: Sử dụng JavaScript và JSX, là các ngôn ngữ phổ biến trong cộng đồng lập trình web. Đây là lợi thế lớn vì nhiều lập trình viên web có thể nhanh chóng làm quen với React Native.
- **Xamarin**: Sử dụng C#, ngôn ngữ quen thuộc với lập trình viên .NET và trong hệ sinh thái Microsoft. Điều này thuận tiện cho các lập trình viên đã quen với hệ sinh thái này nhưng có thể là rào cản cho các lập trình viên không chuyên về C#.

5. Hot Reload / Fast Refresh

- **Flutter**: Hỗ trợ Hot Reload, cho phép cập nhật và xem thay đổi trong thời gian thực, giúp rút ngắn thời gian phát triển và kiểm tra ứng dụng.
- **React Native**: Cũng hỗ trợ tính năng Fast Refresh (tương đương Hot Reload) nhưng không được mượt mà như Flutter.
- **Xamarin**: Có tính năng tương tự gọi là Live Reload, nhưng tính năng này chưa mượt mà bằng Flutter và React Native.

6. Khả năng phát triển đa nền tảng

- **Flutter**: Hỗ trợ cả Android, iOS, Web, Windows, macOS, và Linux, giúp nó trở thành công cụ phát triển đa nền tảng lý tưởng.
- **React Native**: Hỗ trợ Android và iOS. Với một số framework phụ trợ, có thể chạy trên Web nhưng chưa chính thức.
- **Xamarin**: Hỗ trợ Android, iOS, và Windows (UWP), nhưng khả năng mở rộng sang web và các nền tảng khác chưa phổ biến như Flutter.

Tóm lại, Flutter mang đến sự mượt mà về giao diện người dùng, hiệu suất cao, tính năng Hot Reload, thư viện widget phong phú, cùng khả năng phát triển đa nền tảng thực sự. Nó đặc biệt thích hợp cho các dự án yêu cầu giao diện tùy chỉnh cao và hiệu suất tốt gần như ứng dụng native. So với React Native và Xamarin, Flutter hiện đang chiếm ưu thế nhờ vào khả năng hỗ trợ nhiều nền tảng và sự hỗ trợ mạnh mẽ từ Google, mặc dù React Native và Xamarin cũng có thế mạnh riêng khi được sử dụng đúng trường hợp và nhu cầu.

Câu 4.

Các ngôn ngữ lập trình chính được sử dụng để phát triển ứng dụng trên Android bao gồm:

1. Java

- **Đặc điểm**: Java là ngôn ngữ truyền thống và ban đầu được dùng để phát triển Android. Nó là ngôn ngữ lập trình hướng đối tượng với hệ sinh thái mạnh mẽ, ổn định và hỗ trợ tốt từ các thư viện chuẩn của Android.
- **Lý do được chọn**:
    - Tích hợp sâu với Android SDK: Java được Google chọn là ngôn ngữ chính cho Android từ khi hệ điều hành này ra mắt. Do đó, các API và công cụ phát triển của Android được thiết kế tương thích tốt với Java.
    - Đã qua kiểm nghiệm thực tế: Với nhiều năm phát triển trên Android, Java có độ ổn định cao và khả năng tương thích tốt với hầu hết các phiên bản Android cũ.
    - Cộng đồng hỗ trợ lớn: Số lượng tài liệu, hướng dẫn và thư viện phong phú giúp lập trình viên dễ dàng tìm kiếm giải pháp cho vấn đề gặp phải.

2. Kotlin

- **Đặc điểm**: Kotlin là ngôn ngữ hiện đại, được phát triển bởi JetBrains, và được Google chính thức hỗ trợ cho Android từ 2017. Nó là ngôn ngữ tĩnh, hỗ trợ nhiều tính năng hiện đại như lập trình hàm và xử lý an toàn null.
- **Lý do được chọn**:
    - Cú pháp ngắn gọn, dễ hiểu: So với Java, Kotlin có cú pháp ngắn gọn, giúp tăng năng suất lập trình và giảm số lượng lỗi tiềm ẩn.
    - Tương thích ngược với Java: Kotlin hoàn toàn tương thích với mã Java, cho phép các dự án Java chuyển sang Kotlin mà không cần viết lại từ đầu.
    - Được Google khuyến khích sử dụng: Google đã chọn Kotlin làm ngôn ngữ ưu tiên cho Android, nhờ đó có sự hỗ trợ mạnh mẽ từ công cụ và tài liệu.
    - Tính năng an toàn null: Kotlin giúp giảm thiểu lỗi null pointer, một trong những lỗi phổ biến và khó phát hiện trong Java.

3. C++

- **Đặc điểm**: C++ là ngôn ngữ lập trình có khả năng xử lý mạnh mẽ và cho phép can thiệp sâu vào hệ thống, nhờ đó tối ưu hiệu suất ở mức thấp.
- **Lý do được chọn**:
    - Hiệu suất cao: C++ được dùng trong các ứng dụng đòi hỏi hiệu năng, như game hoặc xử lý đồ họa phức tạp. Việc biên dịch trực tiếp xuống mã máy giúp giảm tải bộ xử lý và bộ nhớ.
    - Khả năng tái sử dụng code: Các thư viện hoặc công cụ viết bằng C++ có thể được sử dụng trên nhiều nền tảng khác nhau, giúp tăng hiệu quả trong các dự án đa nền tảng.
    - NDK (Native Development Kit): Android cung cấp NDK, cho phép lập trình viên viết một số phần ứng dụng bằng C++, nhằm tăng hiệu suất cho các phần quan trọng.

4. Dart (Flutter)

- **Đặc điểm**: Dart là ngôn ngữ chính cho Flutter, một bộ công cụ UI phát triển đa nền tảng của Google. Flutter cho phép lập trình viên xây dựng ứng dụng Android và iOS từ một codebase duy nhất.
- **Lý do được chọn**:
    - Hỗ trợ đa nền tảng: Dart/Flutter cho phép lập trình viên viết một lần, triển khai trên nhiều nền tảng khác nhau, giúp tiết kiệm chi phí và công sức.
    - Hiệu năng gần giống native: Dart được biên dịch trực tiếp thành mã máy và tận dụng bộ công cụ đồ họa Skia, giúp ứng dụng Flutter đạt hiệu suất mượt mà gần như ứng dụng native.
    - Cộng đồng và hỗ trợ từ Google: Flutter có sự hỗ trợ lớn từ Google và cộng đồng ngày càng phát triển, tạo sự thuận tiện trong việc tiếp cận tài liệu và thư viện.

5. JavaScript (React Native)

- **Đặc điểm**: JavaScript là ngôn ngữ chính được dùng trong React Native, một framework phát triển ứng dụng đa nền tảng do Meta phát triển.
- **Lý do được chọn**:
    - Cộng đồng lớn và phổ biến: JavaScript là ngôn ngữ phổ biến, với nhiều tài liệu và cộng đồng lập trình viên phong phú.
    - Tiết kiệm thời gian phát triển đa nền tảng: Tương tự Dart/Flutter, JavaScript/React Native cho phép viết một lần, triển khai trên cả Android và iOS.
    - Hỗ trợ từ Meta: React Native được Meta phát triển và sử dụng trong các ứng dụng lớn, đảm bảo được cập nhật và có độ tin cậy cao.

Tóm lại, Java và Kotlin là các ngôn ngữ chủ đạo cho phát triển Android, trong đó Kotlin ngày càng chiếm ưu thế nhờ cú pháp hiện đại và sự hỗ trợ mạnh mẽ từ

Google. Ngoài ra, C++ phù hợp với các ứng dụng cần hiệu năng cao, trong khi Dart và JavaScript giúp phát triển đa nền tảng dễ dàng hơn nhờ Flutter và React Native.

Câu 5.

1. Objective-C

- **Đặc điểm**: Objective-C là ngôn ngữ lập trình lâu đời, được Apple sử dụng chính cho phát triển iOS trước khi Swift ra đời. Đây là ngôn ngữ lập trình hướng đối tượng, mở rộng từ C, với cú pháp phức tạp nhưng linh hoạt.
- **Lý do được chọn**:
    - Tích hợp sâu với hệ sinh thái iOS: Là ngôn ngữ chính của iOS trong nhiều năm, Objective-C vẫn hỗ trợ tốt trong các thư viện cốt lõi của Apple và nhiều ứng dụng legacy.
    - Thừa kế từ C: Objective-C có thể dễ dàng tích hợp với mã C và C++, điều này hữu ích cho các ứng dụng cần hiệu năng cao.
    - Tương thích với Swift: Mặc dù đã cũ, Objective-C có thể được sử dụng đồng thời với Swift, giúp dễ dàng nâng cấp dần các ứng dụng mà không cần viết lại toàn bộ.

2. Swift

- **Đặc điểm**: Swift là ngôn ngữ lập trình hiện đại do Apple phát triển, ra mắt vào năm 2014. Nó hướng tới sự đơn giản, an toàn và hiệu quả, là ngôn ngữ chính được Apple khuyến nghị cho phát triển ứng dụng iOS mới.
- **Lý do được chọn**:
    - Hiệu suất cao và an toàn: Swift tối ưu hiệu suất và xử lý các lỗi thường gặp trong Objective-C như lỗi truy cập null, giúp tăng độ tin cậy cho ứng dụng.
    - Cú pháp đơn giản và dễ đọc: Swift có cú pháp ngắn gọn và hiện đại, giúp lập trình viên viết mã dễ hiểu hơn và giảm bớt lỗi lập trình.

- Hỗ trợ trực tiếp từ Apple: Swift được Apple ưu tiên phát triển và liên tục cập nhật tính năng mới, giúp lập trình viên dễ tiếp cận tài liệu và công cụ hỗ trợ.

3. C++

- **Đặc điểm**: C++ là ngôn ngữ lập trình mạnh mẽ với khả năng kiểm soát bộ nhớ và hiệu suất cao. Trong iOS, C++ thường được sử dụng cho các phần cần hiệu năng, đặc biệt là game hoặc các thư viện xử lý đồ họa phức tạp.
- **Lý do được chọn**:
  - Hiệu năng cao: C++ cho phép lập trình viên tối ưu hiệu suất của ứng dụng ở mức thấp, giúp xử lý nhanh chóng các tác vụ nặng.
  - Dễ tích hợp vào Objective-C và Swift: iOS hỗ trợ "Objective-C++", giúp lập trình viên có thể kết hợp mã Objective-C với C++ trong các dự án.
  - Khả năng đa nền tảng: Một số thư viện hoặc mô-đun viết bằng C++ có thể dễ dàng tái sử dụng trên nhiều nền tảng khác nhau (ví dụ: Android, iOS), giúp tiết kiệm công sức khi phát triển đa nền tảng.

4. Dart (Flutter)

- **Đặc điểm**: Dart là ngôn ngữ lập trình chính cho Flutter, một framework phát triển đa nền tảng do Google phát triển. Flutter cho phép lập trình viên viết một lần, triển khai trên cả iOS và Android.
- **Lý do được chọn**:
  - Phát triển nhanh, đa nền tảng: Dart/Flutter cho phép phát triển một codebase duy nhất cho cả iOS và Android, giúp giảm thời gian và chi phí phát triển.
  - Hiệu năng gần native: Flutter sử dụng engine đồ họa Skia, giúp các ứng dụng có trải nghiệm mượt mà gần giống native.

- Cộng đồng lớn và phát triển nhanh: Flutter đang ngày càng phổ biến và có cộng đồng lập trình viên đông đảo, tài liệu phong phú.

5. JavaScript (React Native)

- **Đặc điểm**: JavaScript là ngôn ngữ chính trong React Native, một framework do Meta phát triển để phát triển ứng dụng di động đa nền tảng.
- **Lý do được chọn**:
  - Viết một lần, triển khai trên nhiều nền tảng: JavaScript và React Native cho phép triển khai ứng dụng trên cả iOS và Android, giúp tiết kiệm thời gian và tài nguyên.
  - Cộng đồng lớn: JavaScript là ngôn ngữ phổ biến với nhiều thư viện và tài liệu hỗ trợ, giúp lập trình viên dễ tiếp cận khi bắt đầu phát triển ứng dụng.
  - Hỗ trợ mạnh mẽ từ Meta: React Native được sử dụng trong các ứng dụng lớn của Meta và nhiều công ty khác, giúp đảm bảo tính ổn định và khả năng mở rộng.

Tóm lại, Swift và Objective-C là hai ngôn ngữ chính cho phát triển ứng dụng iOS, trong đó Swift là ngôn ngữ mới, hiện đại và ngày càng chiếm ưu thế. C++ được dùng cho các ứng dụng cần hiệu suất cao, trong khi Dart/Flutter và JavaScript/React Native giúp phát triển ứng dụng đa nền tảng, phù hợp cho những dự án cần triển khai trên cả iOS và Andorid.

Câu 6.

Windows Phone đã gặp nhiều thách thức lớn, dẫn đến sự suy giảm thị phần và cuối cùng là việc ngừng phát triển. Dưới đây là những thách thức chính và nguyên nhân chính của sự sút giảm :

1. Thiếu sự hỗ trợ từ nhà phát triển ứng dụng

- **Thách thức**: Hệ sinh thái ứng dụng của Windows Phone rất hạn chế. Nhiều ứng dụng phổ biến trên iOS và Android không có mặt trên Windows

Phone, hoặc có bản cũ, ít cập nhật. Điều này làm giảm sức hấp dẫn đối với người dùng, đặc biệt là khi so sánh với các hệ điều hành khác.

- **Nguyên nhân**: Microsoft đã cố gắng khuyến khích các nhà phát triển xây dựng ứng dụng cho Windows Phone, nhưng thị phần thấp và nền tảng chưa phổ biến khiến các nhà phát triển ít ưu tiên. Nền tảng Windows Phone cũng đòi hỏi ngôn ngữ và công nghệ riêng (như C# và XAML), tạo rào cản cho nhà phát triển đã quen với iOS (Objective-C, Swift) và Android (Java, Kotlin).

2. Cạnh tranh mạnh mẽ từ iOS và Android

- **Thách thức**: Windows Phone ra mắt vào thời điểm iOS và Android đã chiếm lĩnh thị trường với lượng người dùng lớn và các tính năng phong phú. Người dùng không có nhiều động lực để chuyển sang một hệ điều hành mới, đặc biệt khi iOS và Android liên tục cải tiến.
- **Nguyên nhân**: iOS và Android đã sớm thiết lập vị thế với hệ sinh thái ứng dụng phong phú, trải nghiệm người dùng tốt và hỗ trợ từ các nhà sản xuất thiết bị lớn. Microsoft đến trễ và gặp khó khăn trong việc thuyết phục cả người dùng lẫn nhà sản xuất chuyển sang Windows Phone.

3. Chiến lược và triển khai thiếu nhất quán

- **Thách thức**: Microsoft thiếu nhất quán trong chiến lược phát triển Windows Phone, thay đổi hướng đi nhiều lần. Điều này tạo sự bất ổn định trong trải nghiệm của người dùng và làm các nhà phát triển mất niềm tin.
- **Nguyên nhân**: Microsoft liên tục thay đổi từ Windows Mobile sang Windows Phone và cuối cùng là Windows 10 Mobile, gây nhầm lẫn về định hướng phát triển và gây khó khăn trong việc duy trì một hệ sinh thái ổn định. Nhiều thiết bị Windows Phone không nhận được cập nhật hoặc hỗ trợ mới, dẫn đến trải nghiệm người dùng giảm sút.

4. Tích hợp kém với các ứng dụng và dịch vụ Google

- **Thách thức**: Do xung đột giữa Microsoft và Google, nhiều dịch vụ và ứng dụng Google phổ biến (như YouTube, Gmail, Google Maps) không được hỗ trợ tốt trên Windows Phone. Điều này gây bất lợi khi người dùng đã quen thuộc và lệ thuộc vào các dịch vụ Google.
- **Nguyên nhân**: Sự cạnh tranh giữa Microsoft và Google khiến Google không phát triển ứng dụng chính thức cho Windows Phone, gây ra hạn chế lớn về tính năng cho người dùng Windows Phone.

5. Thiếu sự đa dạng và sức hấp dẫn của thiết bị

- **Thách thức**: Windows Phone chủ yếu được hỗ trợ bởi Nokia (sau này là Microsoft Lumia), trong khi các nhà sản xuất lớn như Samsung, HTC, và LG ít đầu tư vào nền tảng này. Điều này giới hạn sự lựa chọn của người dùng.
- **Nguyên nhân**: Nhiều nhà sản xuất không muốn đầu tư vào một nền tảng không chắc chắn và có thị phần thấp. Vì thế, Windows Phone thiếu đi các thiết bị có thiết kế và tính năng cạnh tranh, khiến người dùng ít lựa chọn hơn so với Android.

6. Thiếu tính linh hoạt và tùy biến của Android

- **Thách thức**: Windows Phone có giao diện người dùng khá hạn chế và không cho phép tùy chỉnh sâu như Android. Giao diện Metro tuy mới mẻ nhưng không phù hợp với nhiều người dùng, và thiếu tính tùy biến cao.
- **Nguyên nhân**: Microsoft tập trung vào trải nghiệm người dùng đơn giản, nhưng không nắm bắt được nhu cầu tùy biến cao của người dùng di động, điều mà Android thực hiện rất tốt.

Kết luận :

Tổng hợp các yếu tố trên đã khiến Windows Phone mất dần sức hút và không thể cạnh tranh với iOS và Android. Thiếu sự hỗ trợ từ nhà phát triển, chiến lược không nhất quán, cùng các vấn đề về phần mềm và thiết bị đã dẫn đến sự giảm thị phần và cuối cùng là sự suy thoái của Windows Phone trên thị trường di động

Câu 7.

Phát triển ứng dụng web cho thiết bị di động yêu cầu sử dụng một số ngôn ngữ lập trình và công cụ để đảm bảo ứng dụng hoạt động hiệu quả trên các nền tảng di động khác nhau. Dưới đây là các ngôn ngữ và công cụ phổ biến nhất cho việc phát triển ứng dụng web trên thiết bị di động :

1. HTML, CSS, và JavaScript

- HTML (HyperText Markup Language): Là ngôn ngữ đánh dấu cơ bản để cấu trúc nội dung của trang web. HTML5 đã giới thiệu nhiều tính năng mới giúp tối ưu hóa trang web cho thiết bị di động như hỗ trợ video, âm thanh, và tính năng đa phương tiện.
- CSS (Cascading Style Sheets): CSS được sử dụng để thiết kế giao diện người dùng và bố cục của trang web. Các framework như Bootstrap giúp xây dựng giao diện web di động một cách nhanh chóng, đặc biệt là với tính năng responsive design (thiết kế đáp ứng), giúp giao diện tự động thay đổi tùy thuộc vào kích thước màn hình.
- JavaScript: Đây là ngôn ngữ lập trình phía khách hàng giúp tạo ra các tương tác động và xử lý sự kiện trên trang web. Với các thư viện như jQuery Mobile hoặc React, bạn có thể tạo ra các ứng dụng web động và tương tác nhanh chóng, tương thích trên các thiết bị di động.

Ưu điểm:

- Tính tương thích cao với các trình duyệt trên di động và máy tính để bàn.
- Hỗ trợ mạnh mẽ từ các thư viện và framework.

Khuyết điểm:

- Cần phải tối ưu hóa mã nguồn để tránh làm chậm trang web trên các thiết bị di động.

2. React (React.js)

- React là một thư viện JavaScript phổ biến được phát triển bởi Facebook. React cho phép phát triển giao diện người dùng động và linh hoạt, đặc biệt là khi xây dựng các ứng dụng web di động.
- React Native là một framework cho phép phát triển ứng dụng di động sử dụng JavaScript và React. Tuy nhiên, React.js chỉ dành cho phát triển ứng dụng web.
- React DOM cung cấp khả năng cập nhật giao diện người dùng trong thời gian thực khi có sự thay đổi.

Ưu điểm:

- Tính năng component-based architecture giúp tái sử dụng mã dễ dàng.
- Cộng đồng lớn và tài liệu hỗ trợ tốt.
- Tương thích tốt với các công cụ và thư viện phát triển ứng dụng web di động.

Khuyết điểm:

- Đôi khi việc tối ưu hóa có thể gặp khó khăn với những ứng dụng lớn và phức tạp.

3. Vue.js

- Vue.js là một framework JavaScript nhẹ và dễ học, giúp tạo ra các ứng dụng web động và responsive. Vue.js cung cấp tính năng tương tự như React nhưng có cú pháp dễ hiểu hơn, làm cho việc phát triển ứng dụng web trên di động trở nên dễ dàng hơn.
- Vue hỗ trợ tích hợp với các công cụ như Vuex (cho quản lý trạng thái) và Vue Router (cho định tuyến trong ứng dụng web).

Ưu điểm:

- Dễ học và dễ tích hợp với các ứng dụng web hiện có.
- Có thể dễ dàng xây dựng các ứng dụng có hiệu suất cao và gọn nhẹ.

Khuyết điểm:

- Cộng đồng và hệ sinh thái nhỏ hơn so với React.

4. Angular

- Angular là một framework JavaScript do Google phát triển, giúp phát triển các ứng dụng web động với kiến trúc MVC (Model-View-Controller).
- Angular rất mạnh trong việc xây dựng các ứng dụng web phức tạp và có thể sử dụng Angular Mobile Toolkit để tối ưu hóa cho thiết bị di động.

Ưu điểm:

- Cung cấp một bộ công cụ toàn diện cho phát triển ứng dụng web.
- Tính năng two-way data binding giúp đồng bộ dữ liệu giữa view và model.
- Hỗ trợ mạnh mẽ từ Google và cộng đồng.

Khuyết điểm:

- Cần thời gian để học vì có một cú pháp phức tạp và nhiều khái niệm cần nắm vững.

5. Flutter Web

- Flutter là một framework do Google phát triển, chủ yếu dùng để xây dựng ứng dụng di động nhưng cũng hỗ trợ phát triển ứng dụng web. Flutter Web cho phép phát triển giao diện web chất lượng cao và nhanh chóng, sử dụng cùng mã nguồn cho cả ứng dụng di động và web.

Ưu điểm:

- Có khả năng tái sử dụng mã giữa ứng dụng di động và web.
- Tạo giao diện người dùng đẹp và mượt mà với khả năng tùy biến cao.
- Tích hợp tốt với các dịch vụ Google và có cộng đồng phát triển lớn.

Khuyết điểm:

- Dù Flutter Web đang phát triển nhưng vẫn chưa hoàn toàn ổn định so với các công nghệ khác.

6. Cordova (PhoneGap)

- Apache Cordova (trước đây là PhoneGap) là một công cụ mã nguồn mở cho phép phát triển ứng dụng di động và web từ cùng một cơ sở mã nguồn, sử dụng HTML, CSS và JavaScript.
- Cordova cung cấp khả năng truy cập vào các tính năng của thiết bị (camera, GPS, v.v.) thông qua các plugin, giúp phát triển các ứng dụng web di động giống như các ứng dụng gốc.

Ưu điểm:

- Được sử dụng rộng rãi và có thể phát triển ứng dụng cho nhiều nền tảng (iOS, Android, Windows Phone).
- Có thể tận dụng các kỹ năng HTML, CSS, JavaScript mà không cần học thêm nhiều công nghệ mới.

Khuyết điểm:

- Ứng dụng di động có thể không mượt mà như ứng dụng gốc.
- Các plugin của Cordova đôi khi không hỗ trợ đầy đủ các tính năng mới nhất của thiết bị.

7. Progressive Web Apps (PWA)

- PWA là một loại ứng dụng web có thể hoạt động như ứng dụng gốc trên di động. Nó sử dụng các công nghệ như Service Workers và App Shell để tạo ra trải nghiệm người dùng nhanh chóng và mượt mà, ngay cả khi không có kết nối internet.

Ưu điểm:

- Không cần cài đặt từ cửa hàng ứng dụng, giúp người dùng dễ dàng truy cập.

- Tiết kiệm chi phí phát triển vì có thể phát triển duy nhất một ứng dụng cho cả web và di động.

Khuyết điểm:

- Không thể truy cập tất cả các tính năng của thiết bị như các ứng dụng gốc.
- Cần phải tối ưu hóa và hỗ trợ trình duyệt để mang lại trải nghiệm tốt nhất.

Câu 8.

Nhu cầu tuyển dụng lập trình viên trên thiết bị di động vẫn đang tăng cao, vì :

1. **Xu hướng số hóa và di động hóa:**
   - Hầu hết doanh nghiệp đang chuyển đổi số, tập trung vào việc cung cấp ứng dụng di động để tương tác với khách hàng
   - Các lĩnh vực như thương mại điện tử, tài chính, y tế và giáo dục đều cần ứng dụng di động để cạnh tranh và phát triển
2. **Sự phát triển của hệ sinh thái di động:**
   - Nhu cầu phát triển ứng dụng cho Android và iOS ngày càng mở rộng
   - Ngoài điện thoại, các thiết bị IoT và đồng hồ thông minh cũng đang thúc đẩy nhu cầu này
3. **Thiếu hụt nhân lực chất lượng cao:**
   - Mặc dù có nhiều lập trình viên di động trên thị trường, số lượng chuyên gia có kỹ năng chuyên sâu và kinh nghiệm vẫn chưa đủ để đáp ứng nhu cầu.

Những kỹ năng được yêu cầu nhiều nhất :

1. **Kỹ năng ngôn ngữ lập trình**
   - Android : Java, Kotlin
   - iOS : Swift, Ojective-C
   - Cross-platform : React Native, Flutter, Xamarin
2. **Hiểu biết về kiến trúc ứng dụng di động**
   - Kiến thức MVC, MVVM, hoặc MVP để xây dựng ứng dụng có tính linh hoạt và dễ bảo trì.
3. **Kinh nghiệm làm việc với API và cơ sở dữ liệu**

- Thành thạo RESTful APIs và GraphQL để giao tiếp với server
- Hiểu biết về cơ sở dữ liệu như SQLite, Realm hoặc Firebase

4. **Kỹ năng giao diên người dùng (UI/UX)**
   - Khả năng thiết kế giao diện thân thiện với người dùng và phù hợp với từng nền tảng
   - Kinh nghiệm làm việc với các framework UI như Jetpack Compose (Android) hoặc SwiftUI (iOS)
5. **Kỹ năng kiểm thử và tối ưu hóa**
   - Thành thạo unit testing, integration testing, và công cụ kiểm thử tự động (Junit, XCTest).
   - Kỹ năng tối ưu hiệu suất, giảm thiểu tiêu hao bộ nhớ và pin.
6. **Kiến thức về công cụ phát triển và quy trình**
   - Sử dụng thành thạo Android Studio, Xcode, hoặc các IDE liên quan.
   - Hiểu biết về Git, CI/CD, và các công cụ quản lý dự án như Jira, Trello.
7. **Kỹ năng mềm về tư duy logic**
   - Làm việc nhóm hiệu quả, giao tiếo tốt với các bộ phận khác
   - Tư duy giải quyết vấn đề và khả năng tự học hỏi