Câu 1:

Những điện thoại thông minh phổ biến nhất hiện nay dựa trên nền tảng của 2 hệ điều hành thành công nhất là Android của Google và iOS của Apple. Bên cạnh đó còn có những nền tảng khác ít phổ biến hơn như (Windows Phone, BlackBerry, HarmonyOS,…)

Đối với Android:

- Đặc điểm:

- Mở nguồn: Android được phát triển dựa trên nền tảng mã nguồn mở Linux, cho phép tùy biến cao và có một cộng đồng phát triển ứng dụng rất lớn.
- Đa dạng thiết bị: Có mặt trên rất nhiều dòng máy từ bình dân đến cao cấp, từ điện thoại đến máy tính bảng.
- Cửa hàng ứng dụng: Google Play cung cấp một kho ứng dụng khổng lồ, đa dạng về thể loại.

- Ưu điểm:

- Đa dạng thiết bị: dễ dàng tìm thấy một chiếc điện thoại Android phù hợp với nhu cầu và ngân sách.
- Cộng đồng phát triển lớn: Có nhiều diễn đàn, nhóm hỗ trợ giúp người dùng giải đáp thắc mắc.

- Nhược điểm:

- Bảo mật: Có thể tiềm ẩn nhiều rủi ro về bảo mật hơn so với các hệ điều hành đóng.
- Phân mảnh: Do có nhiều thiết bị và phiên bản hệ điều hành khác nhau, gây khó khăn trong việc tối ưu hóa và hỗ trợ.

Đối với iOS:

- Đặc điểm:

- Đóng: Hệ sinh thái iOS được Apple kiểm soát chặt chẽ, đảm bảo sự ổn định và bảo mật cao.
- Tích hợp chặt chẽ: Các thiết bị iOS hoạt động trơn tru với nhau, tạo thành một hệ sinh thái khép kín.
- Cửa hàng ứng dụng: App Store có quy trình kiểm duyệt nghiêm ngặt, đảm bảo chất lượng ứng dụng.

- Ưu điểm:

- Giao diện đẹp, mượt mà: iOS luôn được đánh giá cao về giao diện trực quan và hiệu năng ổn định.
- Bảo mật: Hệ thống bảo mật của iOS được đánh giá là rất tốt.
- Kho ứng dụng App Store chất lượng, ít ứng dụng độc hại.

- Nhược điểm:

- Chi phí phát triển và tiếp cận người dùng cao hơn so với Android.
- Tùy biến hạn chế: Người dùng có ít tùy chọn để tùy chỉnh hệ thống hơn so với Android.

Câu 2:

Native (Android và iOS):

Phát triển riêng cho từng hệ điều hành với ngôn ngữ và công cụ riêng (Java/Kotlin cho Android và Swift/Objective-C cho iOS).

- Ưu điểm: Hiệu suất cao, trải nghiệm người dùng tốt nhất.
- Nhược điểm: Tốn kém vì phải phát triển riêng lẻ cho từng nền tảng.

Flutter:

Sử dụng ngôn ngữ Dart, được Google phát triển, hỗ trợ viết một lần và chạy trên cả Android và iOS.

- Ưu điểm: Giao diện đẹp, hiệu suất gần với ứng dụng native, cộng đồng đang phát triển mạnh.
- Nhược điểm: Còn khá mới, ít tài nguyên và hỗ trợ so với các nền tảng khác.

React Native:

Sử dụng JavaScript và framework React, do Facebook phát triển.

- Ưu điểm: Dễ học, phổ biến trong cộng đồng web, hỗ trợ viết một lần chạy trên nhiều nền tảng.
- Nhược điểm: Hiệu suất không cao bằng ứng dụng native hoặc Flutter.

Xamarin:

Sử dụng C# và .NET, do Microsoft phát triển, hỗ trợ phát triển ứng dụng đa nền tảng.

- Ưu điểm: Tích hợp tốt với hệ sinh thái Microsoft, hỗ trợ đa nền tảng.
- Nhược điểm: Hiệu suất không bằng ứng dụng native, cộng đồng nhỏ hơn.

Tính năng	Native	Hybrid	Cross-Platform
Hiệu năng	Cao nhất	Trung bình	Cao
Giao diện người dùng	Tốt nhất	Tương đối tốt	Tốt
Chi phí phát triển	Cao	Trung bình	Thấp
Thời gian phát triển	Lâu	Nhanh	Trung bình
Khả năng tùy biến	Cao	Trung bình	Trung bình
Công nghệ	Java/Kotlin, Swift/Objective-C	HTML, CSS, JavaScript	C#, Dart

Câu 3:

Ưu điểm nổi bật của Flutter:

Hiệu năng cao:

- Widget trực quan: Flutter sử dụng một bộ widget tùy chỉnh, được render trực tiếp thành các thành phần gốc của UI, giúp tạo ra các ứng dụng với hiệu năng mượt mà và giao diện đẹp mắt.
- Dart: Ngôn ngữ lập trình Dart được biên dịch Ahead-of-Time (AOT), giúp giảm thiểu thời gian khởi động ứng dụng và cải thiện hiệu suất tổng thể.
- Giao diện người dùng đẹp mắt và tùy biến cao:
- Flutter SDK: Cung cấp một bộ công cụ và widget phong phú, cho phép nhà phát triển tạo ra các giao diện người dùng phức tạp và tùy biến cao.
- Hot Reload: Tính năng hot reload giúp cập nhật giao diện người dùng một cách nhanh chóng, tăng tốc quá trình phát triển.

Một mã nguồn cho nhiều nền tảng:

- Cross-platform: Flutter cho phép viết một lần mã và chạy trên nhiều nền tảng (Android, iOS, web, desktop) mà không cần thay đổi nhiều.

Cộng đồng lớn mạnh:

- Hỗ trợ: Flutter có một cộng đồng người dùng lớn và sôi động, luôn sẵn sàng hỗ trợ và chia sẻ kiến thức.
- Thư viện: Có rất nhiều thư viện và package hỗ trợ sẵn, giúp nhà phát triển tiết kiệm thời gian và công sức.

Tính năng	**Flutter**	**React Native**	**Xamarin**
Ngôn ngữ lập trình	Dart	JavaScript/TypeScript	C#
Widget	Tùy chỉnh, render trực tiếp	Component, cầu nối đến native	Xamarin.Forms, XAML
Hiệu năng	Cao	Tương đối cao	Tương đối cao
Giao diện người dùng	Đẹp mắt, tùy biến cao	Tương đối tốt	Tốt
Cộng đồng	Lớn mạnh	Rất lớn	Nhỏ hơn
Nền tảng hỗ trợ	Android, iOS, web, desktop	Android, iOS, web	Android, iOS, Windows

Câu 4:

1. Kotlin

Ngôn ngữ chính thức: Được Google công nhận là ngôn ngữ chính thức cho việc phát triển ứng dụng Android, Kotlin được tích hợp sâu vào Android Studio và nhận được sự hỗ trợ mạnh mẽ từ cộng đồng.

- Ưu điểm:

- Hiệu năng cao: Kotlin biên dịch code thành bytecode tương tự Java, đảm bảo hiệu suất tốt cho ứng dụng.
- An toàn hơn: Giảm thiểu lỗi null pointer exception.

- Tính năng hiện đại: Cung cấp nhiều tính năng hiện đại như null safety, coroutines, data classes, giúp code ngắn gọn và dễ đọc.
- Dễ học: Nếu đã quen với Java, việc chuyển sang Kotlin khá dễ dàng.
- Cộng đồng lớn: Có một cộng đồng lớn và sôi động hỗ trợ.

2. Java

Ngôn ngữ truyền thống: Java đã là ngôn ngữ chính thức cho Android trong nhiều năm và có một lượng lớn tài liệu, thư viện và cộng đồng hỗ trợ.

- Ưu điểm:

- Ổn định: Java là một ngôn ngữ trưởng thành, ổn định và được tin cậy.
- Nền tảng vững chắc: Cung cấp một nền tảng vững chắc để phát triển các ứng dụng Android phức tạp.
- Cộng đồng lớn: Có một cộng đồng rất lớn và lâu đời.
- Thư viện phong phú: Có sẵn rất nhiều thư viện và framework để hỗ trợ phát triển.

Các ngôn ngữ khác:

- Python: Dùng cho các ứng dụng đơn giản, kịch bản và các ứng dụng học máy.
- JavaScript: Dùng cho phát triển ứng dụng hybrid (React Native, Ionic).
- C#: Dùng cho phát triển ứng dụng đa nền tảng với Xamarin.

Câu 5:

1. Swift

Ngôn ngữ chính thức: Swift là ngôn ngữ lập trình được Apple giới thiệu và khuyến khích sử dụng để phát triển ứng dụng iOS.

Ưu điểm:

- Hiệu năng cao: Mã Swift được biên dịch nhanh và chạy mượt mà.
- An toàn: Ngôn ngữ được thiết kế với nhiều tính năng giúp giảm thiểu lỗi thời gian biên dịch và thời gian chạy.
- Dễ học: Cú pháp rõ ràng, dễ đọc và viết.
- Tích hợp với Cocoa Touch: Swift được tích hợp chặt chẽ với Cocoa Touch, framework cung cấp các công cụ và API để xây dựng giao diện người dùng và tương tác với các phần cứng của thiết bị iOS.

Tại sao nên chọn Swift:

- Tương lai: Apple đang đầu tư mạnh vào Swift, vì vậy đây là ngôn ngữ có tương lai sáng sủa.
- Cộng đồng lớn: Có một cộng đồng lớn và sôi động hỗ trợ.
- Tài liệu phong phú: Apple cung cấp nhiều tài liệu và hướng dẫn chi tiết về Swift.

2. Objective-C

Ngôn ngữ truyền thống: Objective-C là ngôn ngữ lập trình gốc để phát triển ứng dụng iOS.

Ưu điểm:

- Ổn định: Objective-C đã được sử dụng trong nhiều năm và có một cộng đồng lớn.
- Thư viện phong phú: Có sẵn rất nhiều thư viện và framework được viết bằng Objective-C.

Nhược điểm:

- Cú pháp phức tạp: Cú pháp của Objective-C có thể khó hiểu và dễ gây nhầm lẫn.
- Ít được ưu tiên phát triển: Apple đang tập trung phát triển Swift hơn là Objective-C.

Tại sao vẫn nên biết Objective-C:

- Mã nguồn mở: Nhiều dự án mã nguồn mở vẫn sử dụng Objective-C.
- Hiểu rõ hơn về iOS: Việc hiểu Objective-C giúp bạn hiểu sâu hơn về kiến trúc và cách hoạt động của iOS.

Câu 6:

1. Thách thức Windows Phone phải đối mặt:

Thiếu ứng dụng và hệ sinh thái ứng dụng yếu:

- So với Android và iOS, Windows Phone có rất ít ứng dụng. Nhiều ứng dụng phổ biến (như Instagram, Snapchat, và các ứng dụng của Google) chỉ có trên Android và iOS hoặc bị giới hạn trên Windows Phone.

- Các nhà phát triển không mặn mà với nền tảng này do lượng người dùng ít, dẫn đến khó thu lợi nhuận và phát triển bền vững.

Hệ sinh thái phát triển chậm chạp và thiếu sáng tạo:

- Dù Microsoft đã cố gắng cung cấp một hệ sinh thái riêng biệt, nhưng không thể thu hút người dùng và nhà phát triển một cách hiệu quả.
- Nhiều tính năng mới, sáng tạo mà Android và iOS đưa ra chưa có mặt kịp thời trên Windows Phone, làm người dùng cảm thấy bị giới hạn.

Khả năng tương thích phần cứng thấp:

- Các thiết bị Windows Phone không phong phú như Android, chỉ có một số ít dòng máy hỗ trợ, chủ yếu từ Nokia. Điều này làm cho người dùng có ít lựa chọn thiết bị hơn, trong khi Android có đủ các phân khúc từ cao cấp đến giá rẻ.

Giao diện người dùng chưa đủ hấp dẫn:

- Windows Phone sử dụng giao diện Metro (Live Tiles), khá khác biệt với giao diện của Android và iOS. Điều này không thu hút được số lượng lớn người dùng, đặc biệt là những người đã quen với giao diện truyền thống trên các nền tảng khác.

2. Nguyên nhân dẫn đến sự sụt giảm thị phần của Windows Phone:

Cạnh tranh mạnh mẽ từ Android và iOS:

- Android và iOS đều có hệ sinh thái ứng dụng và thiết bị đa dạng, cung cấp nhiều tính năng và trải nghiệm người dùng vượt trội. Microsoft đã quá chậm trong việc bắt kịp xu hướng, khiến họ không thể cạnh tranh hiệu quả.

Thiếu sự ủng hộ từ cộng đồng nhà phát triển:

- Android và iOS có lượng nhà phát triển ứng dụng lớn, hỗ trợ phát triển các ứng dụng mới liên tục. Trong khi đó, Windows Phone không đủ hấp dẫn và không có nhiều chính sách hỗ trợ cho các nhà phát triển, khiến hệ sinh thái ứng dụng của Windows Phone bị tụt hậu.

Sự thiếu ổn định trong chiến lược phát triển:

- Microsoft đã thay đổi chiến lược nhiều lần và không nhất quán trong việc phát triển Windows Phone. Họ cũng không hỗ trợ lâu dài cho người dùng,

chẳng hạn khi họ ngừng phát triển Windows Phone, khiến niềm tin của người dùng bị suy giảm.

Tích hợp chậm chạp với các ứng dụng của Google:

- Do sự cạnh tranh, Google đã không cung cấp đầy đủ các ứng dụng của mình trên Windows Phone, như YouTube và Gmail, khiến người dùng bị hạn chế và phải chuyển sang nền tảng khác để có trải nghiệm đầy đủ hơn.

Thời gian ra mắt không thuận lợi:

- Khi Windows Phone xuất hiện, Android và iOS đã nắm bắt phần lớn thị trường smartphone. Windows Phone gặp khó khăn trong việc thuyết phục người dùng chuyển đổi sang một hệ điều hành mới khi đã có các nền tảng quen thuộc.

Câu 7:

Ngôn ngữ lập trình cốt lõi:

- HTML5: Đây là ngôn ngữ đánh dấu chuẩn để cấu trúc nội dung của một trang web. HTML5 cung cấp nhiều thẻ và thuộc tính mới để tạo ra các giao diện người dùng hiện đại và tương tác trên thiết bị di động.
- CSS3: Ngôn ngữ này được sử dụng để định dạng và bố cục các yếu tố trên trang web. CSS3 cung cấp nhiều tính năng mới như các hiệu ứng chuyển động, font chữ tùy chỉnh, và các layout linh hoạt để thích ứng với nhiều kích thước màn hình khác nhau.
- JavaScript: Ngôn ngữ này mang lại sự tương tác cho các trang web. JavaScript cho phép bạn tạo các hiệu ứng động, xử lý sự kiện người dùng và tương tác với các API của trình duyệt.

Framework và thư viện:

- React Native: Được phát triển bởi Facebook, React Native cho phép bạn xây dựng các ứng dụng di động có giao diện người dùng native bằng JavaScript và React. Framework này sử dụng lại một phần lớn mã code giữa các nền tảng (iOS và Android), giúp giảm thời gian phát triển.
- Flutter: Cũng là một framework UI đa nền tảng, Flutter được phát triển bởi Google. Flutter sử dụng ngôn ngữ Dart và cung cấp một bộ widget phong phú để xây dựng các ứng dụng với giao diện đẹp mắt và hiệu năng cao.

- Ionic: Dựa trên Angular, Ionic cho phép bạn xây dựng các ứng dụng hybrid (ứng dụng web đóng gói trong một ứng dụng native) bằng các công nghệ web quen thuộc như HTML, CSS và JavaScript. Ionic cung cấp một bộ component và các tính năng sẵn có để giúp bạn xây dựng ứng dụng nhanh chóng.
- Vue.js: Một framework JavaScript linh hoạt và dễ học, Vue.js cũng được sử dụng để xây dựng các ứng dụng web di động. Vue.js có một cộng đồng lớn và nhiều tài liệu hỗ trợ.

Công cụ hỗ trợ:

- Trình biên tập code: Visual Studio Code, Sublime Text, Atom là những trình biên tập code phổ biến, cung cấp nhiều tính năng hữu ích cho việc viết code.
- Công cụ kiểm tra: Các công cụ như Chrome DevTools, Firefox Developer Tools giúp bạn kiểm tra và gỡ lỗi code, tối ưu hóa hiệu năng của ứng dụng.
- Các công cụ quản lý phiên bản: Git là công cụ phổ biến nhất để quản lý phiên bản code.

Các yếu tố khác cần quan tâm:

- Responsive Design: Thiết kế giao diện web để tự động điều chỉnh kích thước và bố cục phù hợp với các thiết bị di động khác nhau.
- Progressive Web Apps (PWA): Các ứng dụng web tiến bộ cung cấp trải nghiệm gần giống như ứng dụng native, cho phép cài đặt trên màn hình chính, hoạt động offline và gửi thông báo đẩy.
- Performance Optimization: Tối ưu hóa hiệu năng của ứng dụng để đảm bảo trải nghiệm người dùng tốt nhất, bao gồm việc giảm thiểu kích thước file, tối ưu hóa hình ảnh và giảm thiểu thời gian tải trang.

Lựa chọn công nghệ phù hợp:

- Việc lựa chọn công nghệ phụ thuộc vào nhiều yếu tố như:
- Kinh nghiệm của đội ngũ: Nếu đội ngũ đã quen với một framework nào đó, việc sử dụng lại framework đó sẽ giúp tiết kiệm thời gian và chi phí.
- Yêu cầu của dự án: Mỗi dự án có những yêu cầu khác nhau về hiệu năng, tính năng và thời gian phát triển.
- Quy mô của dự án: Với các dự án nhỏ, bạn có thể sử dụng các framework đơn giản như Vue.js. Còn đối với các dự án lớn và phức tạp, React Native hoặc Flutter có thể là lựa chọn phù hợp hơn.

Câu 8:

1. Nhu cầu nguồn nhân lực lập trình viên trên thiết bị di động hiện nay:

- Sự phát triển của các ứng dụng di động: Với số lượng người dùng điện thoại thông minh tăng cao, nhu cầu phát triển ứng dụng di động ngày càng lớn, bao gồm các ứng dụng thương mại điện tử, dịch vụ tài chính, giáo dục, giải trí, và sức khỏe.
- Yêu cầu cho các nền tảng đa dạng: Các công ty cần lập trình viên có thể phát triển ứng dụng trên cả iOS, Android, và đôi khi cả các nền tảng ít phổ biến hơn.
- Lập trình viên full-stack cho mobile: Các doanh nghiệp ngày càng yêu cầu các lập trình viên có thể làm việc từ frontend (UI/UX) đến backend (cơ sở dữ liệu và máy chủ), do đó, lập trình viên có kỹ năng full-stack rất được săn đón.

2. Những kỹ năng được yêu cầu nhiều nhất:

Kỹ năng phát triển native:

- Kotlin cho Android và Swift cho iOS là hai ngôn ngữ chính cho lập trình native.
- Yêu cầu nắm vững các SDK và công cụ phát triển của từng nền tảng như Android Studio và Xcode.

Kỹ năng phát triển đa nền tảng (cross-platform):

- Sử dụng các framework như Flutter và React Native để phát triển ứng dụng một lần nhưng có thể chạy trên cả Android và iOS.
- Lập trình viên cần hiểu rõ về cách tối ưu hóa hiệu suất và thiết kế giao diện người dùng sao cho phù hợp với cả hai hệ điều hành.

Kiến thức về UI/UX Design:

- Hiểu biết về thiết kế giao diện người dùng và trải nghiệm người dùng là yếu tố quan trọng, đảm bảo ứng dụng thân thiện và dễ sử dụng.
- Nắm bắt các nguyên tắc thiết kế của Google Material Design và Apple Human Interface Guidelines.

Kiến thức về API và RESTful Services:

- Khả năng tích hợp ứng dụng với các dịch vụ web thông qua API là một kỹ năng quan trọng, giúp ứng dụng có thể kết nối với máy chủ để lấy và cập nhật dữ liệu.

Kiến thức về cơ sở dữ liệu và quản lý dữ liệu trên thiết bị di động:

- Nắm vững các cơ sở dữ liệu di động như SQLite, Room (Android), Core Data (iOS), và khả năng làm việc với cơ sở dữ liệu cloud như Firebase.

Kỹ năng bảo mật:

- Bảo mật là một vấn đề quan trọng trong phát triển ứng dụng di động, do đó lập trình viên cần hiểu về các phương pháp bảo mật dữ liệu và bảo vệ thông tin người dùng.

Kỹ năng cập nhật công nghệ:

- Công nghệ phát triển rất nhanh, do đó lập trình viên cần luôn sẵn sàng học hỏi các công nghệ mới, cập nhật những thay đổi trong hệ sinh thái di động và các framework phổ biến.

Kỹ năng làm việc nhóm và quản lý dự án:

- Biết cách làm việc trong nhóm và sử dụng các công cụ quản lý dự án như Jira, Trello, và Git là những kỹ năng quan trọng khi làm việc trong các dự án phát triển ứng dụng di động.