Câu 1 :

Các nền tảng thiết bị di động : IOS, Android , BackBerry, window phone

Android:

Ưu điểm:

- Sự đa dạng về thiết bị và giá cả, phù hợp với nhiều đối tượng người dùng.
- Khả năng tùy biến và cài đặt ứng dụng linh hoạt.
- Cộng đồng phát triển mạnh mẽ và phong phú.

Nhược điểm :

- Sự phân mảnh: Nhiều phiên bản Android cùng tồn tại làm khó khăn trong việc cập nhật và hỗ trợ.
- Bảo mật: Do tính mở và khả năng cài đặt ứng dụng từ nguồn không chính thống, nguy cơ bảo mật cao hơn.

IOS:

Ưu điểm :

- Tính ổn định và hiệu năng cao.
- Bảo mật và bảo vệ quyền riêng tư mạnh mẽ.
- Hệ sinh thái Apple đồng nhất và tương thích cao giữa các thiết bị.

Nhược Điểm :

- Độc quyền và giá cả cao: Giới hạn đối tượng người dùng.
- Hạn chế tùy biến: Hạn chế trong việc tùy biến hệ điều hành và cài đặt ứng dụng từ bên ngoài AppStore.

Window Phone:

Ưu điểm :

- Giao diện người dùng trực quan và dễ sử dụng.
- Tích hợp chặt chẽ với các dịch vụ của Microsoft.
- Hiệu năng mượt mà trên các thiết bị cấu hình trung bình.

Nhược Điểm :

- Số lượng ứng dụng hạn chế so với Android và iOS.
- Thiếu sự hỗ trợ và cập nhật từ Microsoft sau khi ngừng phát triển.
- Ít sự lựa chọn về thiết bị.

BackBerry :

Ưu điểm :

- **Bảo mật mạnh mẽ**: Mã hóa và bảo vệ dữ liệu tốt, lý tưởng cho môi trường doanh nghiệp.
- **Quản lý email hiệu quả**: Hệ thống push email và đồng bộ hóa nhanh chóng.
- **Bàn phím vật lý**: Tiện lợi cho việc gõ văn bản, đặc biệt trong công việc.

- **Quản lý doanh nghiệp tốt**: Dịch vụ BlackBerry Enterprise Server giúp quản lý thiết bị từ xa.
- T**hời gian sử dụng pin lâu**: Tối ưu hóa phần mềm giúp tiết kiệm pin.

Nhược điểm :

- **Thiếu ứng dụng**: Cửa hàng ứng dụng hạn chế, ít lựa chọn.
- **Giao diện lỗi thời**: UI không bắt kịp xu hướng hiện đại.
- **Chậm phát triển**: Không thể theo kịp Android và iOS trong việc đổi mới.
- **Thị trường hạn chế**: Chủ yếu phục vụ người dùng doanh nghiệp, ít thu hút người tiêu dùng phổ thông.
- **Khó khăn trong đổi mới phần cứng**: Hạn chế về mẫu mã và tính năng phần cứng.

Câu 2 :

Tiêu chí	**Native App**	**Hybrid App**	**Cross-Platform App**
Định nghĩa	Ứng dụng được phát triển riêng biệt cho từng hệ điều hành (iOS, Android).	Ứng dụng web được bọc trong một khung (container), chạy như ứng dụng di động.	Ứng dụng được phát triển một lần và chạy trên nhiều nền tảng (iOS, Android).
Ngôn ngữ lập trình	Swift, Objective-C (iOS), Java, Kotlin (Android)	HTML, CSS, JavaScript (Dùng với các framework như Cordova, Ionic)	JavaScript (React Native), Dart (Flutter), C# (Xamarin)
Hiệu suất	Cao nhất, trực tiếp truy cập phần cứng và API của hệ điều hành.	Thấp hơn Native, phụ thuộc vào trình duyệt và khả năng của WebView.	Tốt hơn Hybrid, nhưng không thể bằng Native, phụ thuộc vào framework.
Tốc độ	Nhanh nhất vì tối ưu cho từng hệ điều hành cụ thể.	Thường chậm hơn vì chạy trong WebView hoặc dùng phần mềm trung gian.	Nhanh hơn Hybrid, nhưng không thể so sánh với Native về tốc độ.
Khả năng truy cập API	Truy cập đầy đủ và trực tiếp vào các API của hệ điều hành.	Truy cập hạn chế vào API của hệ điều hành, thông qua các plugin hoặc WebView.	Truy cập phần lớn API của hệ điều hành, nhưng có thể cần các plugin hoặc module đặc biệt.
Phát triển và bảo trì	Phát triển riêng biệt cho mỗi nền tảng, tốn thời gian và chi phí cao.	Phát triển một lần, dễ bảo trì, nhưng gặp khó khăn khi yêu cầu hiệu suất cao.	Phát triển một lần cho nhiều nền tảng, dễ bảo trì, nhưng có thể gặp phải vấn đề về hiệu suất và tính năng đặc thù của hệ điều hành.

Trải nghiệm người dùng (UX)	Tốt nhất, vì được thiết kế tối ưu cho từng nền tảng.	Thấp hơn, giao diện có thể không mượt mà như Native, vì phải dựa vào WebView.	Tốt hơn Hybrid, nhưng đôi khi không thể cung cấp trải nghiệm người dùng như Native.
Khả năng tùy chỉnh giao diện	Tùy chỉnh cao nhất, có thể tận dụng đầy đủ các thành phần giao diện của hệ điều hành.	Tùy chỉnh giao diện hạn chế, vì phải dựa vào các công cụ web.	Tùy chỉnh khá tốt, nhưng vẫn bị hạn chế bởi framework và thư viện sử dụng.
Chi phí phát triển	Cao, vì phải phát triển và duy trì riêng biệt cho từng nền tảng.	Thấp, chỉ cần phát triển một ứng dụng duy nhất cho cả hai nền tảng.	Trung bình, phát triển một ứng dụng cho nhiều nền tảng nhưng có thể cần các thư viện hoặc plugin bổ sung.
Ví dụ điển hình	Facebook, Instagram, WhatsApp, TikTok (trước đây)	Facebook (trước đây), Uber (phiên bản cũ), Gmail (WebApp)	React Native: Facebook, Instagram (hiện tại); Flutter: Google Ads, Alibaba, BMW

Câu 3 :

Lí do Flutter lại phổ biến :

- **Flutter** nổi bật với **hiệu suất cao**, **UI tùy chỉnh linh hoạt** và **một mã nguồn duy nhất** cho nhiều nền tảng (iOS, Android, Web, Desktop). Với sự hỗ trợ mạnh mẽ từ **Google**, Flutter đang trở thành một lựa chọn hàng đầu cho các ứng dụng đa nền tảng.

Tiêu chí	Flutter	React Native	Xamarin
Ngôn ngữ lập trình	Dart	Javascript	C#
Hieu suất	Biên dịch trực tiếp thành mã máy, hiệu suất cao	Sử dụng các thành phần UI gốc, hiệu suất gần gốc	Truy cập đầy đủ API gốc, hiệu suất cao
Hệ thống Widget	Hệ thống widget phong phú, tùy biến cao	Component-based, sử dụng các component UI gốc	Xamarin.Forms cho phép chia sẻ mã nguồn UI
Hot Reload	Có	Có	Có
Ưu điểm	Tốc độ phát triển nhanh, giao diện đẹp, cộng đồng hỗ trợ mạnh mẽ	Sử dụng JavaScript phổ biến, tích hợp tốt với hệ sinh thái React, hiệu suất cao	Chia sẻ mã nguồn lớn, sử dụng C#, tích hợp tốt với dịch vụ Microsoft

Nhược điểm	Dart chưa phổ biến rộng rãi, kích thước ứng dụng lớn	Yêu cầu kiến thức lập trình gốc, cộng đồng không đồng đều	Kích thước ứng dụng lớn, cộng đồng nhỏ hơn

Câu 4 :

Java:

- **Ưu điểm**: Được sử dụng lâu dài, có cộng đồng lớn, dễ dàng học và hỗ trợ tốt từ Android SDK.
- **Nhược điểm**: Cú pháp dài dòng và đôi khi không linh hoạt như các ngôn ngữ hiện đại.

Kotlin:

- **Ưu điểm**: Ngôn ngữ hiện đại, ngắn gọn, an toàn với null, dễ dàng tích hợp với Java. Được Google hỗ trợ chính thức.
- **Nhược điểm**: Mới hơn Java nên có thể thiếu một số tài nguyên hoặc cộng đồng lớn như Java.

C++:

- **Ưu điểm**: Hiệu suất cao, truy cập trực tiếp vào phần cứng, phù hợp với các ứng dụng yêu cầu xử lý nặng hoặc đồ họa.
- **Nhược điểm**: Phát triển phức tạp hơn và khó bảo trì, ít được sử dụng cho ứng dụng Android phổ biến.

Dart (Flutter):

- **Ưu điểm**: Phát triển ứng dụng đa nền tảng với hiệu suất cao, hỗ trợ hot reload, dễ dàng phát triển cả trên Android và iOS từ một mã nguồn duy nhất.
- **Nhược điểm**: Cộng đồng nhỏ hơn và thiếu một số thư viện hỗ trợ so với Java hoặc Kotlin.

Python (với Kivy và BeeWare):

- **Ưu điểm**: Dễ học, phát triển nhanh, phù hợp cho các ứng dụng đơn giản hoặc prototype.
- **Nhược điểm**: Không phải ngôn ngữ chính thức của Android và không tối ưu cho các ứng dụng phức tạp.

Câu 5:

Swift: Ngôn ngữ hiện đại, hiệu suất cao, an toàn và dễ học, được Apple hỗ trợ chính thức cho phát triển iOS.

Objective-C: Ngôn ngữ lâu đời, được duy trì hỗ trợ, với tài liệu phong phú và cộng đồng lớn.

C# (với Xamarin): Dành cho phát triển đa nền tảng, sử dụng C# và .NET, tiết kiệm thời gian phát triển.

Dart (với Flutter): Dành cho phát triển đa nền tảng, hiệu suất cao, hỗ trợ tính năng hot reload.

JavaScript (với React Native): Phát triển ứng dụng đa nền tảng, dễ học, và có cộng đồng mạnh mẽ.

Câu 6 :

Windows Phone phải đối mặt với nhiều thách thức dẫn đến sự sụt giảm thị phần và thất bại:

1. **Thiếu ứng dụng**: Windows Phone không thể thu hút đủ các nhà phát triển tạo ứng dụng phổ biến, dẫn đến thiếu các ứng dụng quan trọng như Facebook, Instagram và WhatsApp, khiến người dùng chuyển sang nền tảng khác.
2. **Giao diện và trải nghiệm người dùng**: Mặc dù có giao diện **tile** độc đáo, nhưng người dùng cảm thấy không trực quan và khó sử dụng so với iOS và Android.
3. **Cạnh tranh từ Android và iOS**: Android và iOS đã chiếm lĩnh thị trường di động, có kho ứng dụng phong phú và hệ sinh thái mạnh mẽ, khiến Windows Phone không thể cạnh tranh.
4. **Chiến lược không rõ ràng**: Việc Microsoft mua lại Nokia không giúp Windows Phone thành công. Quá trình chuyển giao gặp khó khăn và thiếu chiến lược tiếp thị mạnh mẽ.
5. **Môi trường phát triển hạn chế**: Hệ điều hành không thu hút đủ nhà phát triển, thiếu ứng dụng và tính năng nổi bật.
6. **Ngừng phát triển**: Microsoft quyết định ngừng phát triển Windows Phone vào năm 2014, chuyển sang tập trung phát triển ứng dụng cho Android và iOS.

Câu 7 :

Ngôn ngữ lập trình chính:

1. **HTML, CSS, JavaScript**: Ba ngôn ngữ cơ bản để phát triển ứng dụng web di động, được hỗ trợ rộng rãi trên tất cả các trình duyệt.
2. **TypeScript**: Superset của JavaScript, giúp kiểm tra kiểu dữ liệu và nâng cao khả năng bảo trì cho ứng dụng.

Frameworks và thư viện:

1. **React.js (React Native)**: Thư viện JavaScript phổ biến, với **React Native** giúp phát triển ứng dụng di động đa nền tảng.
2. **Vue.js**: Framework nhẹ, dễ học, nhưng ít công cụ phát triển di động sẵn có.

3. **Angular**: Framework mạnh mẽ cho ứng dụng web quy mô lớn, có thể kết hợp với **Ionic** để phát triển ứng dụng di động.
4. **Ionic Framework**: Framework phát triển ứng dụng di động đa nền tảng, dựa trên **Angular** và **Cordova**.
5. **Flutter (Dart)**: Framework phát triển ứng dụng di động và web từ một mã nguồn duy nhất, cung cấp hiệu suất gần như native.
6. **Apache Cordova (PhoneGap)**: Công cụ phát triển ứng dụng di động đa nền tảng, sử dụng công nghệ web như HTML, CSS, JavaScript.

Công cụ hỗ trợ:

1. **Progressive Web Apps (PWA)**: Ứng dụng web có thể hoạt động giống ứng dụng native, không cần tải từ cửa hàng ứng dụng.
2. **WebView**: Nhúng trang web trong ứng dụng di động native.
3. **Crosswalk Project**: WebView tối ưu cho Android, cải thiện hiệu suất ứng dụng web trên di động.

Câu 8 :

Trung bình

- intern là 8tr – 12tr
- từ 1 – 2 năm kinh nghiệm là 15tr – 25 tr
- 3+ năm kinh nghiệm là 25tr trờ lên