Báo cáo so sánh chuyên sâu: Flutter và React Native trong năm 2024

1. Giới thiệu: Thiết lập bối cảnh cho Flutter và React Native trong năm 2024.

Flutter và React Native nổi lên như hai framework hàng đầu trong lĩnh vực phát triển ứng dụng di động đa nền tảng. Cả hai đều cho phép các nhà phát triển tạo ra các ứng dụng có thể chạy trên cả hệ điều hành iOS và Android từ một codebase duy nhất, mang lại hiệu quả và tiết kiệm chi phí đáng kể. Tuy nhiên, mỗi framework lại có những đặc điểm, ưu điểm và nhược điểm riêng. Gần đây, đã có những thông tin về việc React Native đã đạt được những cải tiến đáng kể trong hiệu suất, đặc biệt là trong việc giải quyết vấn đề nghẽn cổ chai ở cầu nối JavaScript (JS bridge), một yếu tố mà trước đây thường được những người ủng hộ Flutter đưa ra như một lợi thế của Flutter.[1] Điều này đã làm dấy lên những lo ngại trong cộng đồng các nhà phát triển Flutter về vị thế cạnh tranh của framework này trong tương lai.

Báo cáo này được thực hiện nhằm mục đích cung cấp một cái nhìn toàn diện và cập nhật về sự so sánh giữa Flutter và React Native trong năm 2024. Mục tiêu không chỉ dừng lại ở việc phân tích hiệu suất mà còn mở rộng ra các yếu tố quan trọng khác mà các doanh nghiệp thường cân nhắc khi lựa chọn giữa hai framework này. Báo cáo sẽ đi sâu vào các khía cạnh như hiệu suất, các yếu tố kinh doanh khác, trải nghiệm phát triển, hệ sinh thái, khả năng tái sử dụng mã, khả năng tích hợp và lộ trình phát triển trong tương lai của cả Flutter và React Native. Thông qua việc phân tích chi tiết này, báo cáo hy vọng sẽ mang đến cho các nhà phát triển và các nhà quản lý công nghệ một cơ sở thông tin vững chắc để đưa ra quyết định phù hợp cho các dự án phát triển ứng dụng di động của mình.

2. Bối cảnh hiệu suất: Phân tích sâu về tốc độ và hiệu quả.

- **2.1. Nhìn lại thách thức của cầu nối JavaScript trong React Native:**
  Trong kiến trúc truyền thống của React Native, cầu nối JavaScript (JS bridge) đóng vai trò là trung gian giao tiếp giữa mã JavaScript, nơi phần lớn logic ứng dụng được viết, và các thành phần giao diện người dùng (UI) gốc của hệ điều hành (iOS và Android). Khi một ứng dụng React Native cần hiển thị một thành phần UI gốc hoặc thực hiện một tác vụ liên quan đến hệ thống, thông tin sẽ được tuần tự hóa (serialized) thành dữ liệu JSON và gửi qua cầu nối đến phía native, nơi dữ liệu này sẽ được giải tuần tự hóa (deserialized) và các thành phần UI gốc tương ứng sẽ được tạo hoặc các tác vụ sẽ được thực hiện. Quá trình tuần tự hóa và giải tuần tự hóa này, đặc biệt khi diễn ra thường xuyên với lượng dữ liệu lớn, có thể gây ra độ trễ và trở thành một điểm nghẽn hiệu suất, đặc biệt đối với các ứng

dụng có giao diện người dùng phức tạp hoặc các hiệu ứng động.[1]
Vấn đề hiệu suất liên quan đến JS bridge này đã từng là một trong những điểm yếu chính của React Native so với các công nghệ phát triển ứng dụng native hoặc các framework đa nền tảng khác như Flutter. Những người ủng hộ Flutter thường chỉ ra rằng việc Flutter sử dụng ngôn ngữ Dart và biên dịch trực tiếp thành mã native đã giúp framework này vượt trội hơn React Native về hiệu suất, do không cần đến cầu nối trung gian.[1]

- **2.2. Phân tích các cải tiến và điểm chuẩn hiệu suất gần đây trong React Native:**
  Nhận thức được những hạn chế về hiệu suất do kiến trúc cầu nối gây ra, đội ngũ phát triển React Native đã giới thiệu một kiến trúc mới nhằm giải quyết vấn đề này một cách triệt để.[3] Kiến trúc mới này, được giới thiệu từ phiên bản 0.68 và trở thành mặc định từ phiên bản 0.76, bao gồm các thành phần chính như Fabric, Turbo Modules, JSI (JavaScript Interface) và Codegen.[3]
  - **2.2.1. Kiến trúc mới: Fabric, Turbo Modules và JSI:**
    Fabric là hệ thống render (kết xuất) mới của React Native, được thiết kế để mang lại hiệu quả cao hơn trong việc cập nhật giao diện người dùng.[3] Fabric giảm thiểu thời gian cần thiết để cập nhật UI, dẫn đến trải nghiệm người dùng mượt mà hơn với khả năng cuộn nhanh hơn, phản hồi tốt hơn với các thay đổi focus và các hiệu ứng động trôi chảy hơn. Điều này đạt được bằng cách đồng bộ hóa quá trình render và cho phép giao tiếp trực tiếp hơn giữa mã JavaScript và các thành phần native.[4]
    Turbo Native Modules là một hệ thống mới để viết các module native, cho phép chúng được tải một cách trì hoãn (lazy loading) khi cần thiết.[3] Quan trọng hơn, Turbo Native Modules bỏ qua cầu nối JavaScript và giao tiếp trực tiếp với mã native, làm giảm đáng kể chi phí giao tiếp giữa hai lớp này. Kết quả là, thời gian khởi động ứng dụng nhanh hơn và phản hồi từ lớp native cũng nhanh hơn, giải quyết một trong những điểm yếu thường gặp trong trải nghiệm người dùng ứng dụng di động.[3]
    JSI (JavaScript Interface) là một API C++ cung cấp một lớp trừu tượng cho JavaScript Runtime, cho phép JavaScript giữ các tham chiếu đến các đối tượng C++ host và gọi trực tiếp các phương thức trên chúng.[3] Điều này loại bỏ hoàn toàn chi phí giao tiếp bất đồng bộ và tuần tự hóa dữ liệu qua cầu nối, dẫn đến độ trễ thấp hơn đáng kể và giao diện người dùng mượt mà hơn. JSI cho phép giao tiếp đồng bộ giữa JavaScript và mã native, một điều không thể thực hiện được với kiến trúc cầu nối cũ.[5]
    Codegen là một công cụ tạo mã tự động, giúp tạo mã boilerplate cần thiết để kết nối JavaScript và native code, đảm bảo tính an toàn kiểu dữ liệu và giảm chi phí runtime.[3] Codegen cho phép các nhà phát triển viết đặc tả kiểu tĩnh

bằng TypeScript hoặc Flow, sau đó được sử dụng để tạo các tệp giao diện cần thiết cho các thành phần native của Fabric và TurboModules, đảm bảo rằng cả hai phía đều có thể tin tưởng vào việc xác thực dữ liệu.[5]
Ngoài ra, React Native 0.73 đã giới thiệu **Bridgeless Mode**, cho phép các nhà phát triển tùy chọn vô hiệu hóa hoàn toàn việc tạo cầu nối.[5] Điều này có thể dẫn đến thời gian khởi động ứng dụng nhanh hơn một chút bằng cách loại bỏ chi phí tải cầu nối và các thành phần runtime khác của React Native.

- **2.2.2. Điểm chuẩn và kết quả kiểm thử hiệu suất:**
  Meta (trước đây là Facebook) đã thực hiện các thử nghiệm nội bộ để đánh giá hiệu suất của kiến trúc mới trên các thiết bị di động thực tế và đã ghi nhận những cải thiện đáng kể.[3] Các thử nghiệm này cho thấy sự cải thiện về thời gian phản hồi và hiệu suất tổng thể của các ứng dụng sử dụng kiến trúc mới. Cụ thể, trên một trình giả lập TV 1080p chạy Android 14, đã có sự cải thiện trung bình khoảng 500ms khi sử dụng kiến trúc mới.[3] Trên máy tính bảng Fire HD Plus (thế hệ thứ 10) chạy Fire OS 7.3, sự cải thiện trung bình là khoảng 900ms, và trên Fire TV Stick HD chạy Fire OS 7.6, sự cải thiện trung bình lên đến khoảng 1000ms.[3]
  Các nhà phát triển cộng đồng cũng đã báo cáo những cải thiện về hiệu suất khi chuyển sang kiến trúc mới. Ví dụ, Kraken đã nhận thấy thời gian render nhanh hơn đáng kể, mặc dù có sự khác biệt lớn giữa các màn hình khác nhau và sự cải thiện lớn nhất trên các thiết bị chậm nhất.[3] Alexandre, một người duy trì công cụ đo hiệu suất React Native Flashlight, cũng đã chia sẻ một số điểm chuẩn UI cho thấy những cải thiện rõ rệt.[3]
  Một bài viết trên dev.to đã trình bày các điểm chuẩn chi tiết hơn, cho thấy sự tăng tốc đáng kể trong tốc độ và khả năng phản hồi của các hoạt động module native với kiến trúc mới.[6] Các thử nghiệm đo số lần gọi hàm thành công trong một khoảng thời gian nhất định đã cho thấy sự cải thiện lên đến khoảng 99.60% trên các số lượng invocation khác nhau (từ 1.000 đến 100.000 lần gọi).[6]
  Cùng nguồn này cũng cung cấp các điểm chuẩn về thời gian render của các thành phần View, Text và Image trên các thiết bị Google Pixel 4 và iPhone 12 Pro.[6] Đáng chú ý, iPhone 12 Pro cho thấy sự cải thiện đáng kể hơn, với thời gian render nhanh hơn đến ~39% cho 5.000 thành phần View và ~33% cho 5.000 thành phần Image. Những cải thiện này đặc biệt rõ rệt trong các tình huống có số lượng lớn thành phần, dẫn đến ứng dụng phản hồi nhanh hơn và trải nghiệm người dùng mượt mà hơn.[6]
  Tuy nhiên, cần lưu ý rằng mức độ cải thiện hiệu suất có thể khác nhau tùy thuộc vào thiết kế và cách triển khai cụ thể của ứng dụng.[3] Một số thử nghiệm nội bộ tại Meta cho thấy tác động hiệu suất trung lập trên hầu hết các bề mặt

React Native trong ứng dụng Facebook.[5] Điều này cho thấy kiến trúc mới không phải lúc nào cũng mang lại sự tăng tốc đáng kể ngay lập tức mà còn tập trung vào việc xây dựng một nền tảng vững chắc cho các khả năng trong tương lai.[5] Ngoài ra, trong khi hiệu suất CPU có thể không thay đổi nhiều, mức tiêu thụ bộ nhớ có thể cao hơn với kiến trúc mới do việc biểu diễn UI hoàn toàn trong bộ nhớ, điều này mở ra các tính năng mới như render đồng thời.[7]

- **2.3. Lợi thế hiệu suất của Flutter: Biên dịch AOT và công cụ render:**
  Flutter sử dụng ngôn ngữ lập trình Dart, được thiết kế đặc biệt để phát triển giao diện người dùng và có khả năng biên dịch trực tiếp thành mã máy native (Ahead-Of-Time - AOT compilation).[1] Quá trình biên dịch AOT này thường mang lại hiệu suất vượt trội hơn, đặc biệt trong các tác vụ phức tạp như xử lý animation và thời gian khởi động ứng dụng.[8]
  Flutter sử dụng công cụ render riêng (Skia, với Impeller mới hơn trên iOS là mặc định từ Flutter 3.27 và trên Android thông qua một flag), vẽ các thành phần UI từ đầu.[9] Cách tiếp cận này đảm bảo hiệu suất nhất quán trên các nền tảng khác nhau.[9] Flutter thường vượt trội hơn React Native về mức sử dụng CPU và thời gian tải ứng dụng tổng thể.[9]
  Một lợi thế cốt lõi của Flutter là kiến trúc của nó không yêu cầu cầu nối JavaScript để tương tác với các thành phần native, điều này có thể gây ra sự chậm trễ hiệu suất trong React Native.[12] Việc loại bỏ cầu nối này giúp Flutter đạt được hiệu suất gần như native trong nhiều trường hợp, đặc biệt là khi xử lý các giao diện người dùng tùy chỉnh và các hiệu ứng động phức tạp.[9]
- **2.4. So sánh hiệu suất hiện tại: Một phân tích đối chiếu:**
  Tóm lại, bối cảnh hiệu suất hiện tại cho thấy React Native đã có những bước tiến đáng kể với kiến trúc mới của mình, giải quyết hiệu quả những lo ngại trước đây về JS bridge. Đối với hầu hết các ứng dụng tiêu chuẩn, hiệu suất của React Native thường là rất tốt.[18] Tuy nhiên, đối với các ứng dụng có độ phức tạp cao, yêu cầu nhiều animation, hoặc cần độ nhất quán giao diện pixel-perfect, Flutter vẫn có thể giữ lợi thế nhờ vào quy trình render tối ưu hóa và khả năng biên dịch trực tiếp thành mã native.[9] Một số điểm chuẩn cho thấy Flutter có hiệu suất tốt hơn khi tải công việc cao hơn.[19]
  Cần thừa nhận rằng trong một số trường hợp cụ thể, đặc biệt khi sử dụng các thành phần native, React Native có thể mang lại hiệu suất tốt hơn.[8] Tuy nhiên, lựa chọn "nhanh hơn" có thể khác nhau tùy theo từng trường hợp cụ thể và phụ thuộc vào yêu cầu dự án cụ thể cũng như kinh nghiệm của nhà phát triển trong việc tối ưu hóa cho từng framework.[20] Sự khác biệt về hiệu suất giữa Flutter và React Native đã thu hẹp đáng kể, nhưng kiến trúc của Flutter vẫn cung cấp một nền tảng vững chắc cho các ứng dụng đòi hỏi khắt khe về hiệu suất.

3. Vượt ra ngoài hiệu suất: Các yếu tố chính thúc đẩy quyết định kinh doanh.

- **3.1. Tốc độ phát triển và thời gian đưa sản phẩm ra thị trường:**
  Cả Flutter và React Native đều cung cấp các tính năng như hot reload/fast refresh, cho phép nhà phát triển xem các thay đổi mã ngay lập tức trên ứng dụng đang chạy mà không cần khởi động lại toàn bộ ứng dụng, giúp tăng tốc đáng kể quá trình phát triển.[1] React Native có thể mang lại lợi thế khởi đầu nhanh hơn cho các nhóm đã thành thạo JavaScript, do sự quen thuộc với ngôn ngữ và hệ sinh thái.[1] Tuy nhiên, Flutter với bộ widget dựng sẵn phong phú cũng có thể đẩy nhanh quá trình phát triển, đặc biệt đối với các ứng dụng có giao diện người dùng phức tạp.[8] Trải nghiệm phát triển tích hợp hơn của Flutter với sự hỗ trợ chặt chẽ từ các IDE cũng có thể góp phần vào hiệu quả phát triển.[1] Cả hai framework đều cho phép phát triển nhanh chóng, nhưng kỹ năng hiện có của nhóm phát triển có thể là yếu tố quyết định trong việc chọn lựa framework nào mang lại tốc độ nhanh hơn cho một dự án cụ thể.
- **3.2. Tính linh hoạt trong thiết kế UI/UX và khả năng tùy chỉnh:**
  Flutter với cách tiếp cận dựa trên widget cho phép tạo ra các giao diện người dùng có tính tùy biến cao và biểu cảm, có khả năng đáp ứng chặt chẽ nhận diện thương hiệu.[8] Flutter cung cấp một thư viện widget phong phú, bao gồm các widget tuân theo Material Design và Cupertino, giúp tạo ra các ứng dụng hấp dẫn về mặt thị giác.[8] Flutter mang lại khả năng kiểm soát tùy chỉnh cao hơn nhưng có thể làm mất đi một phần cảm giác native.[1] Ngược lại, React Native sử dụng các thành phần UI native, mang lại trải nghiệm native tự nhiên hơn nhưng có thể hạn chế khả năng tùy chỉnh UI mà không cần thêm nỗ lực.[1] Việc đạt được sự nhất quán UI trên các nền tảng có thể đòi hỏi nhiều nỗ lực hơn trong React Native đối với các thiết kế tùy chỉnh cao.[17] Flutter vượt trội trong việc tạo ra các UI tùy chỉnh cao với giao diện nhất quán trên các nền tảng, trong khi React Native nghiêng về việc cung cấp cảm giác native của nền tảng. Lựa chọn phụ thuộc vào tầm quan trọng của UI đặc trưng thương hiệu và trải nghiệm người dùng mong muốn.
- **3.3. Chi phí phát triển: Tuyển dụng, bảo trì và khả năng mở rộng:**
  Việc xây dựng ứng dụng đa nền tảng bằng cả hai framework thường rẻ hơn so với việc phát triển các ứng dụng native riêng biệt.[23] Chi phí tuyển dụng và duy trì nhà phát triển có thể tương đương nhau nhưng có thể khác nhau tùy thuộc vào vị trí địa lý và nhu cầu thị trường.[9] Việc React Native sử dụng JavaScript, một ngôn ngữ phổ biến rộng rãi, có thể giúp việc tìm kiếm nhà phát triển dễ dàng hơn.[1] Tuy nhiên, sự phổ biến ngày càng tăng của Flutter cũng đang làm tăng số lượng nhà phát triển Dart.[8] Khả năng bảo trì là rất quan trọng và cả hai framework đều được duy trì tích cực, nhưng các lựa chọn kiến trúc có thể ảnh hưởng đến việc bảo trì lâu dài.[1] Về khả năng mở rộng, cả hai framework đều được sử dụng để xây dựng

các ứng dụng lớn và phức tạp, nhưng các cân nhắc về hiệu suất có thể ảnh hưởng đến lựa chọn cho các ứng dụng đòi hỏi cao.[27] Mặc dù cả hai đều mang lại lợi thế về chi phí so với phát triển native, nhưng sự sẵn có và chi phí của các nhà phát triển quen thuộc với mỗi ngôn ngữ, cũng như nhu cầu bảo trì và khả năng mở rộng lâu dài, cần được xem xét.

- **3.4. Nguồn nhân lực và sự sẵn có của nhà phát triển:**
  Sự phổ biến rộng rãi của JavaScript mang lại cho React Native lợi thế về nguồn nhân lực sẵn có.[1] Tuy nhiên, Flutter đang ngày càng phổ biến và số lượng nhà phát triển học Dart đang tăng lên.[8] Đối với các nhóm đã thành thạo React, việc chuyển sang React Native sẽ dễ dàng hơn.[1] Sự dễ học của Flutter có thể thu hút các nhà phát triển từ các nền tảng khác.[17] Kỹ năng hiện có trong một nhóm phát triển thường đóng vai trò quan trọng trong việc lựa chọn framework do dễ dàng đào tạo và tận dụng chuyên môn hiện có.
- **3.5. Sự phù hợp với yêu cầu dự án và độ phức tạp:**
  Flutter có thể được ưu tiên cho các ứng dụng đòi hỏi hiệu suất cao, giao diện người dùng tùy chỉnh phức tạp và tính nhất quán trên nhiều nền tảng (di động, web và desktop).[1] React Native có thể phù hợp hơn cho các dự án ưu tiên giao diện native, phát triển nhanh chóng với kiến thức JavaScript hiện có và tận dụng hệ sinh thái thư viện bên thứ ba rộng lớn.[1] Có những trường hợp sử dụng cụ thể mà một framework có thể phù hợp hơn (ví dụ: Flutter cho các ứng dụng nặng về mặt thị giác, React Native cho các ứng dụng phụ thuộc nhiều vào các thành phần UI native hoặc các API nền tảng cụ thể).[1] Yêu cầu kỹ thuật cụ thể và độ phức tạp của dự án, bao gồm nhu cầu hiệu suất, kỳ vọng về UI/UX và sự phụ thuộc vào các tính năng native, là những yếu tố quyết định quan trọng trong việc lựa chọn framework phù hợp.

4. Trải nghiệm phát triển: So sánh ở cấp độ Framework.

- **4.1. Dễ học và tiếp thu: Dart so với JavaScript/React:**
  React Native có thể có lợi thế ban đầu cho các nhà phát triển đã quen thuộc với JavaScript và React do cú pháp và các khái niệm tương đồng.[1] Flutter sử dụng Dart, có thể gây ra thách thức học tập cho các nhà phát triển chủ yếu có kinh nghiệm với các ngôn ngữ khác.[1] Tuy nhiên, Dart thường được coi là dễ học, đặc biệt đối với những người có kinh nghiệm về các ngôn ngữ hướng đối tượng.[8] Cách tiếp cận dựa trên widget của Flutter được nhiều nhà phát triển đánh giá là trực quan.[10] Mặc dù sự phổ biến của JavaScript mang lại một lượng lớn nhà phát triển ban đầu cho React Native, nhưng Dart của Flutter được thiết kế cho phát triển UI và được coi là tương đối dễ tiếp thu.
- **4.2. Công cụ phát triển và tích hợp IDE:**
  Flutter có khả năng tích hợp tuyệt vời với các IDE phổ biến như Visual Studio Code

và Android Studio, cung cấp các tính năng như tự động hoàn thành mã và công cụ gỡ lỗi.[1] React Native tận dụng sự linh hoạt của việc sử dụng bất kỳ IDE hoặc trình soạn thảo văn bản nào tương thích với JavaScript.[17] Flutter cung cấp giao diện dòng lệnh (CLI) và các công cụ như Flutter Doctor, hỗ trợ thiết lập môi trường phát triển.[21] React Native cũng có CLI riêng và các công cụ như Expo để đơn giản hóa việc thiết lập dự án.[21] Cả hai framework đều cung cấp các công cụ mạnh mẽ, với Flutter mang lại trải nghiệm tích hợp hơn và React Native cung cấp sự linh hoạt cao hơn trong việc lựa chọn IDE.

- **4.3. Tác động của Hot Reload/Fast Refresh đến năng suất:**
  Cả Flutter (Hot Reload) và React Native (Fast Refresh) đều cung cấp các tính năng cho phép nhà phát triển xem các thay đổi mã ngay lập tức mà không cần khởi động lại hoàn toàn ứng dụng, giúp tăng tốc đáng kể quá trình phát triển.[1] Hot Reload của Flutter thường được khen ngợi về tốc độ và độ tin cậy.[1] Fast Refresh của React Native cũng cung cấp khả năng lặp lại nhanh chóng.[32] Những tính năng này rất quan trọng đối với năng suất của nhà phát triển trong cả hai framework, cho phép lặp lại và thử nghiệm nhanh chóng.
- **4.4. Khả năng gỡ lỗi và profiling:**
  Flutter cung cấp các công cụ gỡ lỗi mạnh mẽ, bao gồm kiểm tra cây UI và giám sát hiệu suất.[1] React Native tích hợp với Flipper, một nền tảng gỡ lỗi mạnh mẽ cung cấp thông tin chi tiết về hiệu suất ứng dụng.[32] React Native cũng có một trình gỡ lỗi tích hợp cho các ứng dụng iOS và Android.[17] Cả hai framework đều cung cấp các công cụ đầy đủ để gỡ lỗi và phân tích hiệu suất, cho phép nhà phát triển xác định và giải quyết vấn đề một cách hiệu quả.
- **4.5. Hệ sinh thái quản lý trạng thái:**
  Flutter cung cấp nhiều giải pháp quản lý trạng thái khác nhau, từ đơn giản đến phức tạp (Provider, BLoC, Riverpod, v.v.).[17] React Native dựa vào hệ sinh thái JavaScript rộng lớn hơn cho các thư viện quản lý trạng thái như Redux, Zustand và các thư viện khác.[32] Cả hai framework đều có hệ sinh thái trưởng thành để quản lý trạng thái ứng dụng, cung cấp cho nhà phát triển nhiều lựa chọn dựa trên độ phức tạp của dự án và sở thích của nhóm.

5. Hệ sinh thái và cộng đồng hỗ trợ: Sức mạnh từ số lượng và chất lượng.

- **5.1. Quy mô, tính tích cực và mức độ tương tác của cộng đồng:**
  React Native, ra đời trước, có một cộng đồng lớn hơn và trưởng thành hơn.[1] Tuy nhiên, Flutter có một cộng đồng đang phát triển nhanh chóng và tích cực, nhận được sự hỗ trợ và công nhận mạnh mẽ.[1] Một số khảo sát cho thấy Flutter có tỷ lệ người hâm mộ cao hơn trong giới nhà phát triển.[8] React Native vẫn có sự hiện diện mạnh mẽ trong số các nhà phát triển chuyên nghiệp.[10] Mặc dù React Native hiện có cộng đồng lớn hơn, nhưng cộng đồng của Flutter đang phát triển nhanh

chóng và rất năng động, cho thấy một tương lai đầy hứa hẹn.

- **5.2. Tính sẵn có và chất lượng của các thư viện và gói bên thứ ba:**
  React Native hưởng lợi từ hệ sinh thái npm rộng lớn và trưởng thành, cung cấp các gói cho hầu hết mọi nhu cầu.[1] Flutter có một hệ sinh thái mới hơn nhưng đang phát triển nhanh chóng, tập trung vào các gói chất lượng cao có sẵn trên pub.dev.[1] Cần lưu ý rằng hệ sinh thái npm lớn hơn có thể có nguy cơ phân mảnh và bảo trì không nhất quán.[1] Các gói được quản lý của Flutter thường nhận được sự hỗ trợ mạnh mẽ từ cộng đồng và các công ty.[1] Hệ sinh thái lớn hơn của React Native cung cấp nhiều lựa chọn hơn, trong khi hệ sinh thái đang phát triển của Flutter ưu tiên chất lượng và khả năng bảo trì.
- **5.3. Tài liệu, tài nguyên học tập và các kênh hỗ trợ:**
  Flutter được đánh giá cao về tài liệu rõ ràng, toàn diện và được tổ chức tốt, bao gồm danh mục widget tương tác và các hướng dẫn dành riêng cho các nhà phát triển từ các nền tảng khác nhau.[17] Tài liệu của React Native, mặc dù phong phú, đôi khi có thể cảm thấy rời rạc.[17] Cả hai framework đều có vô số hướng dẫn, bài viết và tài nguyên từ bên thứ ba nhờ cộng đồng tích cực của họ.[1] Flutter thường được khen ngợi về tài liệu xuất sắc, đây có thể là một lợi thế đáng kể cho các nhà phát triển, đặc biệt là những người mới làm quen với framework.
- **5.4. Vai trò của sự hỗ trợ từ công ty mẹ: Flutter của Google so với React Native của Meta:**
  Google có sự hỗ trợ mạnh mẽ và tích cực phát triển Flutter, bao gồm cả việc sử dụng nó trong các sản phẩm của chính Google.[8] Meta (trước đây là Facebook) cũng hỗ trợ React Native cùng với cộng đồng lớn mạnh của nó.[8] Cả hai framework đều hưởng lợi từ nguồn lực và chuyên môn của các công ty mẹ tương ứng. Sự hỗ trợ từ các công ty công nghệ lớn mang lại sự ổn định và hỗ trợ lâu dài cho cả hai framework, đảm bảo với các nhà phát triển về sự liên tục của chúng.

6. Khả năng tái sử dụng mã và tính nhất quán của UI: Tối đa hóa hiệu quả và nhận diện thương hiệu.

- **6.1. Khả năng chia sẻ mã trên các nền tảng (iOS, Android, Web, Desktop):**
  Cả hai framework đều cho phép chia sẻ mã đáng kể giữa iOS và Android.[1] Flutter có khả năng mạnh mẽ trong việc mở rộng khả năng tái sử dụng mã sang web, desktop (Windows, macOS, Linux) và thậm chí cả các hệ thống nhúng.[9] React Native hỗ trợ web thông qua React Native Web, cho phép tái sử dụng mã trên các nền tảng di động và web, mặc dù có thể yêu cầu thêm các thư viện hoặc giải pháp thay thế.[8] React Native cũng đang tăng cường hỗ trợ cho các nền tảng desktop như macOS và Windows.[35] Flutter cung cấp một giải pháp toàn diện hơn cho phát triển đa nền tảng ngoài iOS và Android, trong khi React Native phụ thuộc nhiều hơn vào các thư viện bên ngoài cho web và desktop.

- **6.2. Cách tiếp cận phát triển UI: Hệ thống dựa trên Widget của Flutter so với các thành phần Native của React Native:**
  Flutter sử dụng kiến trúc dựa trên widget, nơi UI được xây dựng bằng cách sử dụng một hệ thống phân cấp các widget mô tả giao diện của chúng dựa trên cấu hình và trạng thái.[1] Flutter render tất cả các thành phần trên canvas riêng bằng công cụ render của nó (Skia/Impeller), mang lại UI nhất quán trên các nền tảng.[1] React Native sử dụng các thành phần UI native (ví dụ: UIButton trên iOS, Button trên Android), render chúng theo nền tảng.[1] Sự khác biệt cơ bản này trong cách tiếp cận render UI có những tác động đáng kể đến tính nhất quán của UI và "cảm giác native" của các ứng dụng.
- **6.3. Đạt được tính nhất quán của UI và duy trì giao diện Native:**
  Flutter ưu tiên tính nhất quán của UI trên các nền tảng do khả năng kiểm soát quá trình render và việc cung cấp các bộ widget Material và Cupertino.[1] Mặc dù Flutter hướng đến một UI giống native, nhưng nó có thể không hoàn toàn giống native và người dùng có thể nhận thấy những khác biệt nhỏ.[1] React Native vốn cung cấp giao diện và trải nghiệm native hơn bằng cách sử dụng các thành phần UI dành riêng cho nền tảng.[1] Việc đạt được tính nhất quán UI pixel-perfect trên các nền tảng trong React Native có thể đòi hỏi nhiều nỗ lực hơn và có khả năng cần mã hoặc thư viện bên thứ ba dành riêng cho nền tảng.[8] Lựa chọn giữa ưu tiên tính nhất quán của UI (Flutter) so với giao diện native (React Native) phụ thuộc vào mục tiêu cụ thể của dự án và trải nghiệm người dùng mong muốn trên mỗi nền tảng.
- **Bảng 1: Tóm tắt về khả năng tái sử dụng mã và cách tiếp cận tính nhất quán của UI.**

Khía cạnh	Flutter	React Native
Khả năng tái sử dụng mã (Di động)	Rất cao, gần 100% giữa iOS và Android [5]	Cao, thường khoảng 80-90% giữa iOS và Android [5]
Khả năng tái sử dụng mã (Web)	Mạnh mẽ, hỗ trợ tốt cho web [9]	Hỗ trợ thông qua React Native Web, có thể yêu cầu thêm thư viện [8]
Khả năng tái sử dụng mã (Desktop)	Hỗ trợ tốt cho Windows, macOS và Linux [9]	Đang phát triển hỗ trợ cho macOS và Windows [35]

Cách tiếp cận render UI	Dựa trên widget, render mọi thứ trên canvas riêng bằng công cụ Skia/Impeller [1]	Sử dụng các thành phần UI native của nền tảng [1]
Tính nhất quán của UI	Ưu tiên tính nhất quán trên các nền tảng, cung cấp bộ widget Material và Cupertino [1]	Có thể đòi hỏi nhiều nỗ lực hơn để đạt được tính nhất quán pixel-perfect trên các nền tảng cho các thiết kế tùy chỉnh [8]
Giao diện Native	Hướng đến giao diện giống native nhưng có thể có những khác biệt nhỏ [1]	Cung cấp giao diện native hơn do sử dụng các thành phần UI native của nền tảng [1]

7. Flutter trong thực tế: Ứng dụng thực tế và các trường hợp điển hình.

- **7.1. Nêu bật các công ty và ứng dụng nổi bật được xây dựng bằng Flutter:** Nhiều công ty lớn và các ứng dụng phổ biến đã lựa chọn Flutter cho việc phát triển ứng dụng di động của họ, chứng minh sự trưởng thành và khả năng của framework này trong việc xây dựng các ứng dụng sẵn sàng cho sản xuất.[2] Một số ví dụ bao gồm:
  - Google (Google Ads, Google Earth) [2]
  - Alibaba (ứng dụng thương mại điện tử toàn cầu) [9]
  - eBay [10]
  - Tencent [10]
  - BMW (các tính năng xe hơi kết nối, trải nghiệm lái xe thống nhất) [2]
  - The New York Times [10]
  - Toyota (hệ thống thông tin giải trí trên xe hơi) [25]
  - SAS Airlines (ứng dụng khách hàng) [25]
  - Credit Agricole (ứng dụng ngân hàng di động bán lẻ) [25]
- **7.2. Phân tích lý do đằng sau lựa chọn Flutter của họ:** Dựa trên tài liệu nghiên cứu, có thể thấy nhiều lý do khiến các công ty này lựa chọn Flutter [2]:
  - Hiệu suất mượt mà và giao diện nhất quán trên các nền tảng (Alibaba, Google Pay).[9]
  - Phát triển đa nền tảng hiệu quả (BMW).[9]
  - Khả năng hiệu suất cao và giao diện nhất quán cho các ứng dụng phong phú về mặt thị giác.[17]
  - Hỗ trợ cho các thiết bị nhúng (Toyota).[25]
  - Khả năng tạo ra trải nghiệm thực sự đa nền tảng (Google Earth).[25]

    - Nhu cầu về một ứng dụng ngân hàng di động bán lẻ mạnh mẽ và hấp dẫn về mặt thị giác (Credit Agricole).[25]

Việc các công ty lớn trong nhiều ngành khác nhau chấp nhận Flutter cho thấy sự trưởng thành và tính phù hợp của nó trong việc xây dựng các ứng dụng sẵn sàng cho sản xuất. Những ví dụ thực tế này và lý do đằng sau việc áp dụng chúng có thể trấn an người dùng về khả năng và giá trị mà Flutter mang lại cho các doanh nghiệp.

8. Khả năng tích hợp với các module Native và các công nghệ khác.

- **8.1. Khám phá các kênh nền tảng và khả năng tương tác Native của Flutter:** Flutter cung cấp tính năng "add-to-app", cho phép tích hợp từng phần các module Flutter vào các ứng dụng native Android và iOS hiện có.[47] Flutter sử dụng các kênh nền tảng (MethodChannel, EventChannel, BasicMessageChannel) làm cầu nối giao tiếp giữa mã Flutter và mã native.[48] Các trường hợp sử dụng tích hợp native bao gồm truy cập các tính năng cụ thể của thiết bị hoặc tích hợp với các SDK dành riêng cho nền tảng.[48] Flutter cũng hỗ trợ multi-engine và multi-view cho các kịch bản tích hợp phức tạp hơn.[47] Flutter cung cấp các cơ chế mạnh mẽ để tích hợp với các codebase native hiện có và truy cập các chức năng dành riêng cho nền tảng khi cần thiết.
- **8.2. Xem xét hệ thống module Native và Turbo Modules của React Native:** React Native cung cấp Hệ thống Module Native, cho phép viết mã native (Java/Kotlin cho Android, Objective-C/Swift cho iOS) và hiển thị nó cho JavaScript.[55] Các trường hợp sử dụng cho module native trong React Native bao gồm truy cập các tính năng thiết bị không được hiển thị theo mặc định hoặc tích hợp với các thư viện native bên thứ ba.[55] Turbo Native Modules là API thế hệ tiếp theo để kết nối JavaScript và mã native trong Kiến trúc Mới, mang lại lợi ích về hiệu suất so với cầu nối cũ.[3] Turbo Modules sử dụng JSI để tương tác trực tiếp với JavaScript engine, cho phép các API đồng bộ và giao tiếp nhanh hơn.[3] React Native cũng cung cấp các cách toàn diện để tích hợp với mã native, với Kiến trúc Mới tiếp tục nâng cao hiệu quả của việc tích hợp này.
- **8.3. So sánh tính dễ dàng và linh hoạt của tích hợp:** Cả hai framework đều cung cấp các giải pháp hiệu quả cho việc tích hợp native, nhưng các chi tiết triển khai cụ thể và trải nghiệm của nhà phát triển có thể khác nhau. Lựa chọn có thể phụ thuộc vào sự quen thuộc của nhóm với các phương pháp tích hợp nền tảng tương ứng và các yêu cầu cụ thể của việc tích hợp.
- **8.4. Tích hợp với các dịch vụ Backend và nền tảng đám mây:** Cả Flutter và React Native đều có thể tích hợp với nhiều dịch vụ backend và nền tảng đám mây (Firebase, AWS Amplify, Google Cloud, backend tùy chỉnh, v.v.).[1] Flutter có khả năng tương thích liền mạch với Firebase (một sản phẩm của Google).[17] React Native có nhiều thư viện bên thứ ba để tích hợp backend trong

hệ sinh thái JavaScript.[1] Cả hai framework đều cung cấp sự linh hoạt tuyệt vời trong việc tích hợp với các dịch vụ backend và nền tảng đám mây, mang đến cho nhà phát triển nhiều lựa chọn.

9. Bức tranh tương lai: Lộ trình phát triển và các tính năng dự kiến.

- **9.1. Lộ trình phát triển của Flutter: Các lĩnh vực trọng tâm và những cải tiến sắp tới:**
  Flutter đang tiếp tục nỗ lực cải thiện hiệu suất đa nền tảng, bao gồm tối ưu hóa công cụ render Skia/Impeller.[33] Dự kiến sẽ có những cải tiến trong hỗ trợ desktop, chẳng hạn như các tính năng dành riêng cho nền tảng tốt hơn, các thành phần UI native và các tương tác bàn phím/con trỏ được tinh chỉnh.[33] Hiệu suất web cũng sẽ được cải thiện với một quy trình render nâng cao và các cải tiến về bố cục đáp ứng.[33] Các công cụ phát triển cũng sẽ được nâng cấp, chẳng hạn như các công cụ gỡ lỗi thông minh hơn và khả năng profiling hiệu suất tích hợp trong Flutter DevTools.[33] Flutter có thể sẽ tích hợp sâu hơn với các công nghệ AI và Machine Learning như TensorFlow Lite.[31] Các giải pháp quản lý trạng thái và hỗ trợ đồng thời trong Dart cũng sẽ tiếp tục được cải thiện.[31] Flutter đang ngày càng được áp dụng trong lĩnh vực hệ thống nhúng.[33] Lộ trình phát triển của Flutter đến năm 2025 tập trung vào di động, cải thiện các công cụ xây dựng (SwiftPM, Kotlin trong Gradle) và nâng cao Flutter web.[64] Dart 3.0 cũng mang đến những cải tiến đáng chú ý như khả năng null safety tốt hơn và hỗ trợ cải thiện cho lập trình bất đồng bộ.[31] Lộ trình phát triển của Flutter cho thấy sự cam kết mạnh mẽ đối với việc liên tục cải thiện trên tất cả các nền tảng được hỗ trợ, tập trung vào hiệu suất, trải nghiệm nhà phát triển và mở rộng khả năng.
- **9.2. Lộ trình phát triển của React Native: Sự phát triển của Kiến trúc Mới và các kế hoạch trong tương lai:**
  React Native đang tập trung vào việc tiếp tục triển khai và áp dụng Kiến trúc Mới (Fabric, Turbo Modules, JSI) và tác động của nó đối với hiệu suất và khả năng.[3] Kế hoạch là sẽ biến Kiến trúc Mới thành mặc định vào cuối năm 2024.[6] React Native cũng giới thiệu các tính năng CSS mới để cải thiện bố cục và kiểu dáng, phù hợp hơn với các tiêu chuẩn web.[70] Trong tương lai, React Native có thể sẽ tích hợp với AR/VR, các thiết bị IoT và các công nghệ AI/ML.[35] Việc mở rộng sang các nền tảng desktop (macOS, Windows) cũng đang được tiến hành.[35] React Native cũng đang nỗ lực hướng tới sự bền vững, bao gồm giảm tiêu thụ năng lượng và thời gian tải nhanh hơn.[35] Các công cụ phát triển như Flipper cũng sẽ được cải thiện.[35] Việc sử dụng Concurrent Mode (các tính năng của React 18) sẽ giúp cải thiện khả năng phản hồi của UI.[35] Tương lai của React Native tập trung vào sự phát triển và áp dụng Kiến trúc Mới, hứa hẹn những cải tiến đáng kể về hiệu suất và kiến trúc.
- **9.3. Các yếu tố gây gián đoạn tiềm năng và xu hướng mới nổi:**

Cần xem xét các xu hướng hoặc công nghệ mới nổi có thể ảnh hưởng đến bối cảnh phát triển đa nền tảng và sự phù hợp của Flutter và React Native trong tương lai. Điều này có thể bao gồm những tiến bộ trong phát triển native, sự trỗi dậy của các framework đa nền tảng khác hoặc những thay đổi đáng kể trong hệ sinh thái nền tảng. Mặc dù cả hai framework hiện đang dẫn đầu trong không gian đa nền tảng, nhưng cần nhận thức được tính năng động của công nghệ và khả năng xuất hiện các yếu tố gây gián đoạn trong tương lai.

10. Kết luận: Tại sao Flutter vẫn là một lựa chọn hấp dẫn trong năm 2025 và hơn thế nữa.

Tóm lại, báo cáo này cho thấy rằng mặc dù React Native đã có những tiến bộ đáng kể trong việc giải quyết các lo ngại về hiệu suất, Flutter vẫn tiếp tục mang lại những lợi thế đáng kể trong nhiều lĩnh vực. Flutter vẫn giữ vững vị thế của mình nhờ hiệu suất vượt trội (đặc biệt đối với các UI phức tạp), tính nhất quán của UI trên các nền tảng, khả năng hỗ trợ đa nền tảng toàn diện (bao gồm web và desktop), trải nghiệm phát triển tuyệt vời và một hệ sinh thái đang phát triển nhanh chóng.

Người dùng, với tư cách là một nhà phát triển Flutter có kinh nghiệm, có thể yên tâm rằng Flutter vẫn là một framework mạnh mẽ và phù hợp cho việc phát triển ứng dụng di động trong năm 2025 và được định vị tốt cho tương lai với lộ trình phát triển mạnh mẽ và sự đổi mới liên tục. Cả Flutter và React Native đều là những công cụ quan trọng trong bối cảnh phát triển đa nền tảng và sẽ tiếp tục phát triển để đáp ứng nhu cầu ngày càng cao của thị trường.

Works cited

1. Flutter vs React Native: Which framework is best for your app in ..., accessed April 15, 2025, https://www.appwrite.io/blog/post/flutter-vs-react-native
2. React-native: why JS bridge is the performance bottleneck - Stack Overflow, accessed April 15, 2025, https://stackoverflow.com/questions/50789227/react-native-why-js-bridge-is-the-performance-bottleneck
3. How does React Native's New Architecture affect performance ..., accessed April 15, 2025, https://dev.to/anishamalde/how-does-react-natives-new-architecture-affect-performance-1ioe
4. How does React Native's New Architecture affect performance? - DEV Community, accessed April 15, 2025, https://dev.to/amazonappdev/how-does-react-natives-new-architecture-affect-performance-1dkf
5. Experiment With the New Architecture of React Native | {callstack}, accessed

April 15, 2025, https://www.callstack.com/blog/experiment-with-new-architecture-of-react-native
6. React Native's New Architecture: An Overview of Performance ..., accessed April 15, 2025, https://dev.to/yoel/react-native-new-architecture-what-you-need-to-know-1eke
7. Performance benchmarks · reactwg react-native-new-architecture · Discussion #85 - GitHub, accessed April 15, 2025, https://github.com/reactwg/react-native-new-architecture/discussions/85
8. Flutter vs React Native: A Comparison Chart for Your 2024 Projects, accessed April 15, 2025, https://www.leanware.co/insights/flutter-vs-react-native-comparison-chart-2024
9. Flutter vs React Native in 2024: An In-Depth Guide | TechAhead, accessed April 15, 2025, https://www.techaheadcorp.com/blog/flutter-vs-react-native-in-2024-the-ultimate-showdown-for-app-development-dominance/
10. Flutter vs React Native for Mobile App Development in 2024, accessed April 15, 2025, https://www.bacancytechnology.com/blog/flutter-vs-react-native
11. Flutter vs Angular vs React: The Ultimate Showdown [2024] - Simplilearn.com, accessed April 15, 2025, https://www.simplilearn.com/flutter-vs-angular-vs-react-article
12. Flutter vs React Native: Which one should you opt for your Business? - Antino, accessed April 15, 2025, https://www.antino.com/blog/flutter-vs-react-native
13. React Native vs Flutter: What to Choose in 2025 | BrowserStack, accessed April 15, 2025, https://www.browserstack.com/guide/flutter-vs-react-native
14. Flutter vs React Native: Which Framework Wins in 2024? - WTF Blog, accessed April 15, 2025, https://blog.flutter.wtf/flutter-vs-react-native/
15. Flutter vs React Native in 2024: Which one is faster? - Primotech, accessed April 15, 2025, https://www.primotech.com/flutter-vs-react-native-in-2024/
16. What's new in the docs - Flutter Documentation, accessed April 15, 2025, https://docs.flutter.dev/release/whats-new
17. Flutter vs React Native – Which is Better for Your Project? | Blog - Droids On Roids, accessed April 15, 2025, https://www.thedroidsonroids.com/blog/flutter-vs-react-native-comparison
18. Flutter vs React Native: Which One To Choose? - DesignRush, accessed April 15, 2025, https://www.designrush.com/agency/mobile-app-design-development/trends/flutter-vs-react-native
19. Flutter Performance Breakdown: Is Flutter Fast? - Flatirons Development, accessed April 15, 2025, https://flatirons.com/blog/flutter-performance-breakdown-is-flutter-fast/
20. Flutter vs. React Native: Which is Faster for App Development in 2024? - Reddit, accessed April 15, 2025, https://www.reddit.com/r/FlutterDev/comments/193ytm2/flutter_vs_react_native_which_is_faster_for_app/

21. Flutter vs. React Native in 2025 — Detailed Analysis - Nomtek, accessed April 15, 2025, https://www.nomtek.com/blog/flutter-vs-react-native
22. Flutter vs React Native 2024 : r/FlutterDev - Reddit, accessed April 15, 2025, https://www.reddit.com/r/FlutterDev/comments/18yg593/flutter_vs_react_native_2024/
23. Flutter vs. React Native: Which Framework to Choose for Your Next Mobile App Development? - Netguru, accessed April 15, 2025, https://www.netguru.com/blog/flutter-vs-react-native
24. Flutter vs. React Native: Which Framework to Choose in 2025? - Simplilearn.com, accessed April 15, 2025, https://www.simplilearn.com/tutorials/reactjs-tutorial/flutter-vs-react-native
25. Flutter vs. React Native Comparison: Key Differences - LeanCode, accessed April 15, 2025, https://leancode.co/blog/flutter-vs-react-native
26. Flutter vs React Native: Which One is the Best Framework? - Radixweb, accessed April 15, 2025, https://radixweb.com/blog/flutter-vs-react-native
27. Flutter vs React Native: A Comprehensive Comparison for 2024 - Stackademic, accessed April 15, 2025, https://blog.stackademic.com/flutter-vs-react-native-a-comprehensive-comparison-for-2024-a19bd9f68d54
28. Comprehensive Guide to React Native App Development in 2025 - Tekrevol, accessed April 15, 2025, https://www.tekrevol.com/blogs/react-native-app-development-guide/
29. Optimizing Flutter App Performance: Best Practices for 2024, accessed April 15, 2025, https://ingeniousmindslab.com/blogs/flutter-app-performance/
30. Ionic vs. React Native vs. Flutter - Research - Hugging Face Forums, accessed April 15, 2025, https://discuss.huggingface.co/t/ionic-vs-react-native-vs-flutter/128132
31. Continuing to Rely on Flutter in 2025: A Smart Choice [Update] - Blog - Codigee, accessed April 15, 2025, https://codigee.com/blog/continuing-to-rely-on-flutter-a-smart-choice
32. Optimising App Performance in React Native: A Detailed Guide ..., accessed April 15, 2025, https://dev.to/rushi-patel/optimising-app-performance-in-react-native-a-detailed-guide-2g8e
33. What's New for Flutter in 2025: A Look Ahead to Flutter 4.0 and Beyond - 200OK Solutions, accessed April 15, 2025, https://200oksolutions.com/blog/whats-new-for-flutter-in-2025-flutter-4-0/
34. Top Tips to Boost React Native Performance in 2025 - Netguru, accessed April 15, 2025, https://www.netguru.com/blog/react-native-performance
35. React Native in 2025: Key Developments and Future Potential - 200OK Solutions, accessed April 15, 2025, https://200oksolutions.com/blog/react-native-in-2025-key-developments-and-future-potential/
36. A Complete Flutter Roadmap for 2025 - igmGuru, accessed April 15, 2025, https://www.igmguru.com/blog/flutter-roadmap

37. React-Native-Advanced-Guide/Performance-Optimization ... - GitHub, accessed April 15, 2025, https://github.com/anisurrahman072/React-Native-Advanced-Guide/blob/master/Performance-Optimization/Performance-Optimization-coding-guide.md
38. Did the new React Native architecture make it faster/as fast as flutter? - Reddit, accessed April 15, 2025, https://www.reddit.com/r/reactnative/comments/1aoz3kv/did_the_new_react_native_architecture_make_it/
39. React native performance : r/reactnative - Reddit, accessed April 15, 2025, https://www.reddit.com/r/reactnative/comments/1iq6nn9/react_native_performance/
40. Performance Overview · React Native, accessed April 15, 2025, https://reactnative.dev/docs/performance
41. Bridge Android Native to React Native get Slow Performance - Stack Overflow, accessed April 15, 2025, https://stackoverflow.com/questions/64263945/react-native-bridge-android-native-to-react-native-get-slow-performance
42. August 2024: Flutter 3.24 Highlights, Flutter vs RN Benchmarks, HTTP Clients Deep Dive, accessed April 15, 2025, https://codewithandrea.com/newsletter/august-2024/
43. Why Flutter Destroys React Native in Benchmarks | Deep dive! - YouTube, accessed April 15, 2025, https://www.youtube.com/watch?v=GmyMZMchR-0
44. Flutter beats React Native in virtually every benchmark : r/FlutterDev - Reddit, accessed April 15, 2025, https://www.reddit.com/r/FlutterDev/comments/1exkjcg/flutter_beats_react_native_in_virtually_every/
45. Lynx vs. React Native: Performance Implications and Benchmarking - Rutvik Bhatt, accessed April 15, 2025, https://www.rutvikbhatt.com/lynx-vs-react-native-performance-implications-and-benchmarking/
46. React Native Database Performance Comparison - PowerSync, accessed April 15, 2025, https://www.powersync.com/blog/react-native-database-performance-comparison
47. Add Flutter to an existing app - Flutter Documentation, accessed April 15, 2025, https://docs.flutter.dev/add-to-app
48. Integrating Native SDKs in a Flutter Application - Wawandco, accessed April 15, 2025, https://wawand.co/blog/posts/integrating-native-sdks-in-flutter/
49. Writing custom platform-specific code - Flutter Documentation, accessed April 15, 2025, https://docs.flutter.dev/platform-integration/platform-channels
50. Migrating existing native Android/iOS applications to Flutter - AgileVision.io, accessed April 15, 2025, https://agilevision.io/blog/migrating-existing-native-android-ios-applications-to-flutter/
51. How to integrate Flutter into an existing native app: 2 options - Funda Engineering

blog, accessed April 15, 2025, https://blog.funda.nl/how-to-integrate-flutter-into-an-existing-native-app-two-approaches/
52. Flutter module integration with native project - Stack Overflow, accessed April 15, 2025, https://stackoverflow.com/questions/79272009/flutter-module-integration-with-native-project
53. Integrating Flutter modules inside native apps? : r/flutterhelp - Reddit, accessed April 15, 2025, https://www.reddit.com/r/flutterhelp/comments/1bwihwq/integrating_flutter_modules_inside_native_apps/
54. Add Flutter to your existing app - Tencent Cloud, accessed April 15, 2025, https://www.tencentcloud.com/document/product/1047/51456
55. How to integrate Native Modules in React Native iOS Apps - Morrow Digital, accessed April 15, 2025, https://www.themorrow.digital/blog/how-to-integrate-native-modules-in-react-native-ios-apps
56. Integrating Native Modules in React Native: Bridging the Gap - Metadesign Solutions, accessed April 15, 2025, https://metadesignsolutions.com/integrating-native-modules-in-react-native-bridging-the-gap/
57. Native Modules Intro - React Native, accessed April 15, 2025, https://reactnative.dev/docs/next/native-modules-intro
58. Native Modules: Introduction - React Native, accessed April 15, 2025, https://reactnative.dev/docs/turbo-native-modules-introduction
59. Communication between native and React Native, accessed April 15, 2025, https://reactnative.dev/docs/communication-ios
60. How to Easily Create Native Libraries With Nitro and Turbo Modules | {callstack}, accessed April 15, 2025, https://www.callstack.com/blog/bridgeless-native-development
61. Using Native Modules for Extending React Native Features: Understanding the Need for Native Modules : Course Building Cross-Platform Apps with React Native | Cursa, accessed April 15, 2025, https://cursa.app/en/page/using-native-modules-for-extending-react-native-features-understanding-the-need-for-native-modules
62. How to Integrate React Native With an Existing App | {callstack}, accessed April 15, 2025, https://www.callstack.com/blog/how-to-integrate-react-native-with-an-existing-app
63. Reflecting on Flutter in 2024: Breakthroughs, Innovations, and What's Next in 2025, accessed April 15, 2025, https://somniosoftware.com/blog/reflecting-on-flutter-in-2024-breakthroughs-innovations-and-whats-next-in-2025
64. Google's Flutter Roadmap has been updated for 2025 : r/FlutterDev - Reddit, accessed April 15, 2025,

https://www.reddit.com/r/FlutterDev/comments/1jrz4cd/googles_flutter_roadmap_has_been_updated_for_2025/
65. Flutter in 2023: What are the Plans & Roadmap for Flutter? - ashutec, accessed April 15, 2025, https://www.ashutec.com/blog/flutter-in-2022-what-are-the-plans-roadmap-for-flutter-1655ef658fce
66. Future of Flutter Trends and Predictions for 2025 : r/FlutterDev - Reddit, accessed April 15, 2025, https://www.reddit.com/r/FlutterDev/comments/1hhsnrq/future_of_flutter_trends_and_predictions_for_2025/
67. Five years of React Native at Shopify (2025), accessed April 15, 2025, https://shopify.engineering/five-years-of-react-native-at-shopify
68. React Native 0.79 - Faster tooling and much more, accessed April 15, 2025, https://reactnative.dev/blog/2025/04/08/react-native-0.79
69. What's next for React Native in 2024 | Adapptor, accessed April 15, 2025, https://www.adapptor.com.au/blog/whats-next-for-react-native-in-2024
70. React Native 0.77 - New Styling Features, Android's 16KB page support, Swift Template, accessed April 15, 2025, https://reactnative.dev/blog/2025/01/21/version-0.77
71. Why React Native Is the Future and How to Master it in 2025 - YouTube, accessed April 15, 2025, https://www.youtube.com/watch?v=HbCBN8o3Xno
72. About the New Architecture - React Native, accessed April 15, 2025, https://reactnative.dev/architecture/landing-page