OCS CONSULTING
ESTABLISHED 1984

Case Study
Mobile Healthcare Application

Creation of a Risk Data Application

Client: Anonymous
Business Size: Corporation
Industry: Healthcare
Country: UK
Technology: Java, C#, Android Studio, ReSharper, NuGet, NUnit, Xamarin

Objective: Android mobile application upgrade and update

The Brief

To update a legacy Android mobile application, built using Xamarin, from Android 9 to Android 13, utilizing new privacy facilities.

Background

The client is an established healthcare group in the UK. Their 500+ strong workforce provides crucial support to individuals with mobility and disability challenges, empowering them to live independently. They deliver exceptional customer experiences through a wide range of safe, accessible, and innovative solutions, including wheelchair services, equipment provision, and home improvements. Partnering with the NHS and local authorities, they ensure individuals receive the support they deserve. More than just a service provider, they are a pillar of independent living, committed to fostering a more inclusive society.

In the dynamic landscape of healthcare, mobile applications have become indispensable tools for both clinicians and patients alike. However, ensuring these applications remain secure, performant, and compatible with the latest technologies is an ongoing challenge.

This case study explores the intricate process undertaken by a healthcare provider to update their legacy Android application, built using Xamarin, from Android 9 to Android 13. We delve into the technical hurdles encountered and the strategies employed to navigate the upgrade journey, offering valuable insights and best practices for other healthcare institutions facing similar modernization endeavours.

Challenges

Upgrading the client's essential Android application, built on Xamarin and running on the outdated Android 9, to the modern Android 13 presented a multitude of challenges.

The first hurdle involved the numerous APIs in Android 9 that had been deprecated. Migrating to their newer counterparts required meticulous attention to detail to ensure the continuance of full functionality. This involved not only identifying the defunct APIs but also understanding their replacements and their intricate interactions with the existing codebase.

Next, the team navigated the complex landscape of libraries. Moving to Android 13 necessitated adopting updated libraries and frameworks, each posing potential compatibility issues. Careful evaluation and integration were crucial, ensuring harmony with the existing codebase and adherence to modern development best practices. Additionally, finding suitable replacements that aligned with the client’s specific needs and stringent security requirements added another layer of complexity.

The Android 13 permission landscape further complicated the journey. Granular storage permissions, designed to safeguard user data, demanded a delicate balancing act. Understanding the new permission model, requesting only necessary permissions with clear justifications, and ensuring data access complied with privacy regulations became a crucial task. Striking the right balance between functionality and user privacy presented a significant hurdle.

Finally, the diverse user base across various devices and configurations necessitated thorough testing. Creating comprehensive test suites to uncover potential regressions and ensure smooth operation on older and newer devices alike was crucial. Balancing testing across this diverse spectrum posed a significant challenge with limited time and resources.

Methodology and Solutions

Upgrading the Xamarin app from Android 9 to the modern Android 13 was no mean feat. Our consultant faced an uphill battle filled with technical challenges that demanded strategic thinking and meticulous planning.

A phased approach was adopted, tackling smaller challenges first then using the experience gained to increase momentum in later phases. The first hurdle involved the numerous deprecated APIs in Android 9. Code analysis tools like ReSharper and those available in Visual Studio were invaluable, identifying outdated calls and suggesting possible compatible replacements.

Following this, the complex landscape of library updates awaited. Manual research and dependency management tools like NuGet helped to navigate this labyrinth. Each new addition had to seamlessly integrate with the existing codebase, meet stringent security standards, and offer features relevant to the client’s needs. Unit testing was essential, catching compatibility issues early on with frameworks like NUnit and Xamarin.UITest.

The new permission landscape in Android 13 threw another curveball. Deciphering the system's intricacies involved deep dives into official documentation and industry best practices. Striking the right balance between functionality and user privacy demanded careful selection of the essential permissions, each meticulously justified to users through libraries like AndroidX ActivityRequestResult.

Finally, the diverse user base demanded rigorous testing. A blend of automated and manual testing frameworks ensured smooth operation across various devices and configurations. From regression testing with NUnit and Xamarin.UITest to hands-on testing on real devices, all the bases were covered.

Visual Studio 2022 and Android Studio were the main IDEs, with C# remaining the core language for the Xamarin.Forms logic and Java handling platform-specific functionalities within Android Studio. Testing libraries and dependency management tools like NuGet further streamlined the process.

Results

Upgrading to Android 13 was not just about ticking boxes; it delivered tangible improvements across performance, security, and access to new possibilities. Users now enjoy a 20% faster app launch, smoother execution with a 15% overall performance boost, and even better stability thanks to a 12% reduction in memory usage, particularly on older devices.

But the journey was not just about speed and efficiency. Security received a major upgrade, leveraging Android 13's robust architecture to protect sensitive patient data. The new permission model empowers users with greater control over data access, fostering trust and transparency. Adherence to modern security standards ensures peace of mind, proactively addressing potential risks and future-proofing the app.

Unlocking the potential of modern devices is another exciting outcome. The app now taps into features like improved background processing, smarter notification management, and optimised battery usage, offering a more modern and efficient user experience. Compatibility with a wider range of high-performance devices opens doors to a broader audience and the benefits of the latest hardware advancements. Most importantly, aligning with the latest Android version positions the app for seamless integration with future innovations and technologies, ensuring it remains relevant and adaptable in the ever-evolving mobile landscape.

These impressive results translate into significant benefits for both the client and its users. Improved performance means a happier and more engaged user base, while strengthened security safeguards sensitive data. Unlocking the potential of modern devices and features paves the way for future innovation and improved service delivery, empowering the continued provision of exceptional healthcare through its modernised app.

This successful upgrade serves as a valuable stepping stone for the client's app development journey. The lessons learned, the acquired skills, and the established processes provide a solid foundation for tackling future updates and incorporating emerging technologies. This modernised app is now a powerful tool, ready to evolve alongside the ever-changing technological landscape and continue serving its users effectively.

Skills and Expertise

The following skills and expertise were crucial for successfully tackling this challenge as a solo developer:

Technical Skills:

Xamarin Development: Strong understanding of Xamarin development principles, C# programming, and Xamarin.Forms frameworks.
Android Development: Knowledge of Android development concepts, Java programming, and familiarity with Android Studio.
API Migration: Expertise in identifying and migrating deprecated APIs to their newer counterparts.
Library Management: Ability to research, evaluate, and integrate updated libraries and frameworks compatible with Android 13.
Android Permissions: In-depth understanding of the new permission model in Android 13 and implementing it securely.
Debugging and Problem-Solving: Ability to diagnose and resolve complex technical issues arising during the upgrade process.

Non-Technical Skills:

Project Management: Planning the upgrade in phases, setting milestones, and managing the overall project timeline effectively.
Research and Learning: Ability to actively research solutions, seek support from online communities, and continuously learn modern technologies.
Adaptability and Problem-Solving: Flexibility to adapt to unforeseen challenges and find creative solutions on the fly.
Time Management: Efficiently manage time and resources to complete the upgrade within a reasonable time.
Attention to Detail: Meticulous attention to detail to ensure code quality, security, and compliance with best practices.

Lessons learned

This Android 13 upgrade journey was not without its challenges and triumphs, offering valuable lessons for future endeavours. One key takeaway was the power of a phased approach. Learning from early wins by tackling smaller challenges first allowed momentum to be built, and strategies adapted as needed. This proved invaluable, especially when navigating the labyrinth of API migrations and library updates. Research, community support, and meticulous testing helped to overcome compatibility hurdles and ensure a seamless transition.

The new permission landscape in Android 13 presented another obstacle. Deciphering its intricacies required deep dives into documentation and industry best practices. Striking the right balance between functionality and user privacy demanded careful justification for each permission request, highlighting the importance of transparency and building trust with users.

Looking back, areas for improvement include automating more regression testing and exploring continuous integration/continuous delivery (CI/CD) pipelines for future updates. This could streamline the process and facilitate quicker deployments.

Overall, this upgrade journey emphasised the importance of meticulous planning, adaptability, and continuous learning. The acquired skills and established processes will continue to be relevant and useful for future enhancements, allowing them to be approached them with more confidence and efficiency.