SUCCESS STORY FOR QA FOR MOBILE APPS DEVELOPMENT

BACKGROUND

The development of an iPhone application that is correlated to a major Enterprise scale Web Application was an unexplored domain for the client. The testing matrices for mobile applications are exponentially more complex than for Web and desktop applications. As a result the client had to face a multitude of challenges in designing a methodology for successful testing and defect detection of iPhone and iPad applications. The client leveraged Enosis's core competencies by availing iPhone application testing services to ensure proper application functionality on iPhones & iPads connected to live global networks. Enosis Solutions used its highly experienced Quality Assurance & Testing team that is well versed with the iPhone Application Testing, SDK, Mobile architectures, and challenges associated with building and testing mobile application.

THE CLIENT

SpoonByte is a US based "software on demand" service provider offering iPhone & Web app that allows restaurants to create promotions (e.g., meal discounts, menu specials, or events) to increase revenue. It's the first app that serves the food industry by focusing on Match-Based Deal Dissemination for the physical marketplace.
SpoonByte intelligently sends each promotion only to consumers that are near the restaurant and are predicted to act on the promotion and visit the restaurant.

THE APPLICATION

The client partnered with Enosis Solution to design and develop an Enterprise Scale Web Application that used advanced analytics and complex statistical algorithms to help both consumers and restaurant owners make more informed decisions. The application is a customized Business Intelligence Web based tool that uses the client's high end predictive modeling technology to lead consumers to restaurants that match their preference and proximity. This enabled consumers to find the right deal at the right time and allowed restaurant owners to attract more customers, thus increasing profitability. To allow consumers to reap the benefits of the Web 2.0 application at any time and from any location; the client collaborated with Enosis to design a similar application for iPhone & iPad devices. This mobile application matches individuals to the best local restaurants via offers, events, promos and by considering their proximity.

SpoonByte in iPhone

CHALLANGES

Enosis SQA team has to overcome many complex challenges in order to fulfill client's stringent quality requirements since the application has to support Restaurant owners at one end and hundreds of thousands of consumers at the other. Moreover there are many user-friendly and robust iPhone applications available in the market and that means a tough competition already exists. This 'Consumer Matching' application is a completely new concept and due to the complex nature of the application, it is very difficult for the QA team to depict the final product at the beginning. Enosis's QA Team had to ensure that the application:

- Worked flawlessly in the live environment
- Was compatible with any data service available
- Was accessible from any location
- Can be supported on any version of iPhone OS or handset

ENOSIS SOLUTIONS' APPROACH

The Enosis Solutions QA Team conducts extensive testing procedures on the iPhone application to detect possible errors in functionality, content, image display and data protection.

ENOSIS' APPROACH (CONT.)

- Enosis Solutions' App Testing Solutions include:
- Functionality Testing
- Device Certificate Verification
- Performance and Scalability Testing
- Usability & Compatibility Testing
- Testing UI Response of Application
- Checking settings response
- Identification of crash Points
- Testing in low connectivity mode
- Response Memory Handling
- Sleep Mode Testing
- Accelerometer Response Testing
- Battery Usage Validation
- Retrials in Case of network disconnect

TESTING STRATEGIES

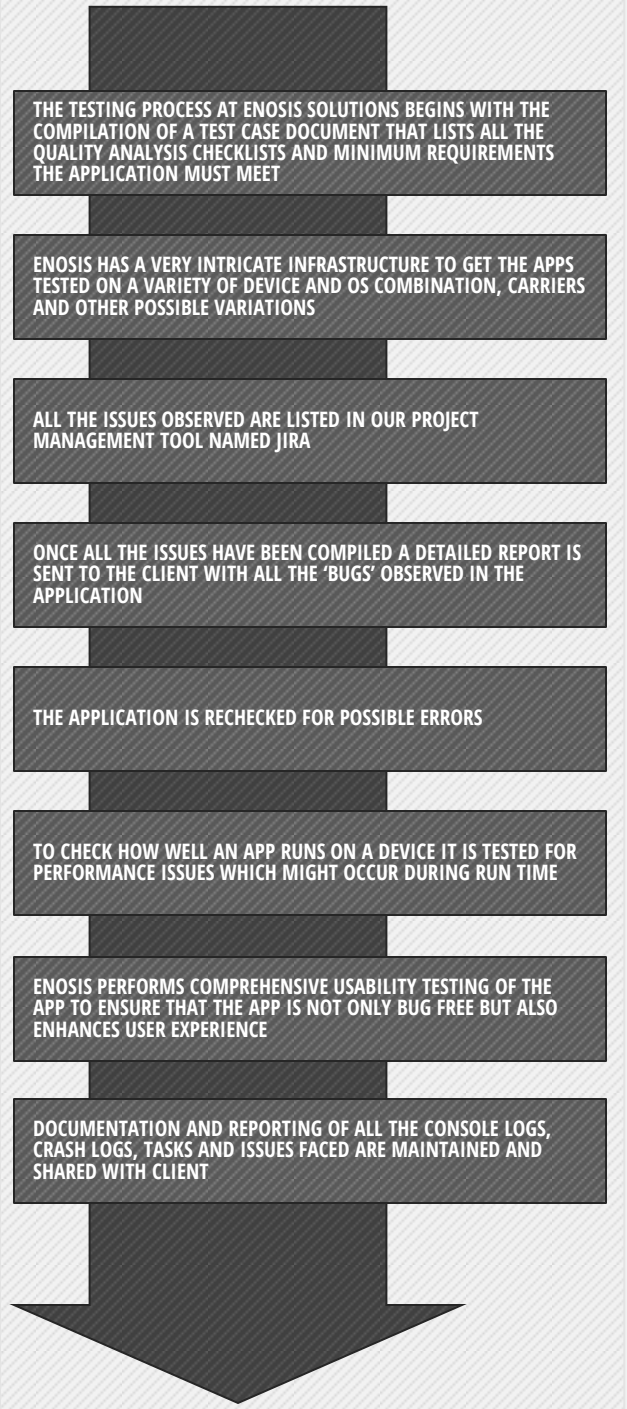
As the iPhone applications become so ubiquitous, the need for reliable testing services has become even more compelling. Enosis has outlined several testing strategies to increase the testing coverage and allow in time defect detection.

- **Accurately report available memory:** Many of the non-reproducible bugs encountered when testing iPhone apps are related to memory problems. It is quintessential to know and report available free memory before launching an application. In most cases, the reproducibility of a "crashing" iPhone app bug is related to low memory conditions because a crashing defect may disappear when there's plenty of free memory. Enosis QA Team uses **Memory Sweep**, a tool to determine the free memory and mock crashing bugs by forcing free memory to a very low level, e.g. < 3MB, before proceeding with the tests. Manually free memory can be suppressed by opening several Safari windows before starting the testing.

- **Perform surveillance on the app from the console:** iPhone apps report application and system level warnings to the console. These warnings are reviewed in real-time using **Apple's iPhone Configuration Tool** to help refine the steps needed to reproduce complex (and memory related) problems.

- **Test unique byte-code generation for a consumer:** Multiple byte-codes cannot be generated for an offer accepted by a consumer. This scenario is tested by URL tempering. As accepting an offer is a post request, so Http Resource Test (Firefox plug-in) is used.

Testing Methodology



- **Test the push notification service:** Whenever there is a new offer available for a registered consumer, the push notification service sends a request (with device token ID's) to the APNs to send a push notification. The push notification service is tested by **manipulating data of the database.** So, several sets of test data is generated as per scenario requirement.

TESTING STRATEGIES (CONT.)

- **Test the "test drive" feature for anonymous user:** An anonymous user can get some recommended offers by providing a valid zip code and rate the restaurants within that zip code. This feature has been tested in both the offline mode (when no network connection is available) and in the online mode. Test environment setup to be very through to get the actual results.

- **Testing the Server Response:** One can easily send a **HTTP GET** request to the server by calling the service from the browser Address Bar. But **HTTP POST** requests cannot be tested in similar fashion. So to test the iPhone SpoonByte app **POST** requests; **HTTP Resource Test** Firefox plug-in is used and the corresponding JSON is checked to ensure the request has been successfully executed by the server or not. Unit test codes are written to submit the POST requests from controller level.

- **Testing the Services:** There is a significant limitation in the availability of appropriate tools to check the functionality of the mobile services. So, unit test codes are developed and executed. Consequently, an automated test suite is created.

- **Testing the HTTPS request:** For security reason the login and registration step send HTTPS request to the server. To test this behavior, all the incoming and outgoing packets to and from the server has to be captured. To triumph over the challenge of tracking all the incoming and outgoing packets, **WireShark** (network sniffer tool) is used and the packet data contents are properly checked.

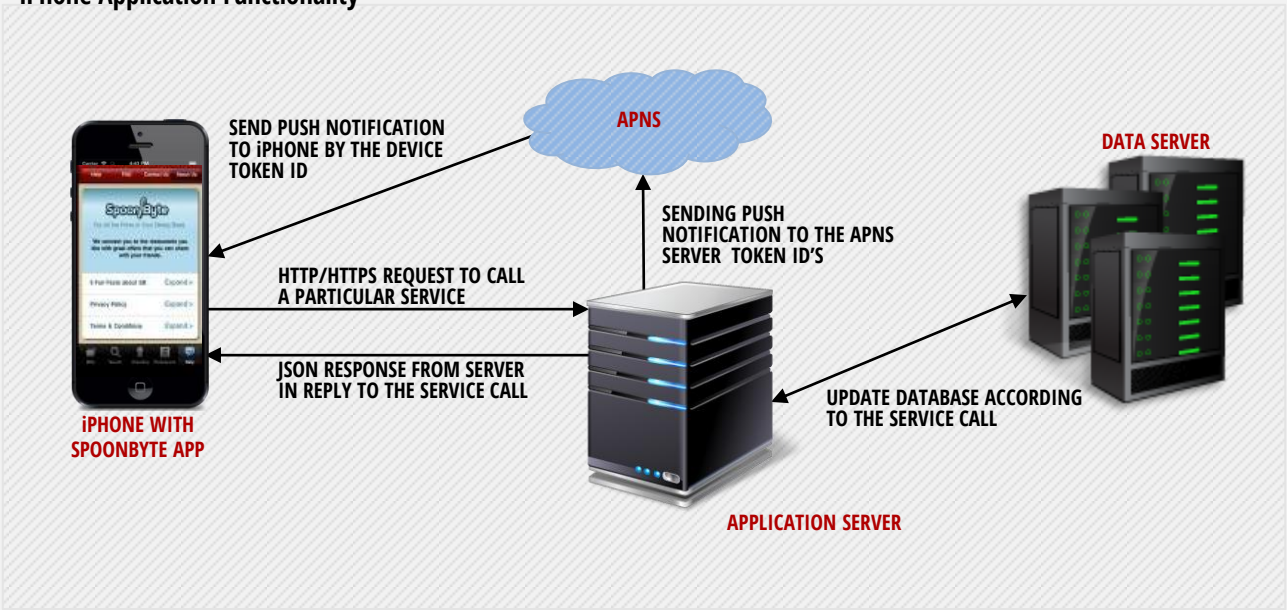
- **Provide 'crash reporter' logs with the defect reports:** Each time an iPhone application crashes, a .crash file is created on the iPhone. This file is retrieved when the iPhone is synched with iTunes and associated with reported bug for better analysis.

- **Using screenshots:** Screenshots always help to clarify a UI bug to a developer. Using the iPhone's built-in screen capture function, the screenshot is captured and attached with the bug report.

- **Provide useful defect characterization information:** Developers always need support in their debugging process, and useful defect characterization helps them narrow down the root cause of a bug. If a crash happens under low memory conditions, then it is tested under conditions where there's lots of memory available, e.g. >40MB. If a problem occurs under iPhone OS 2.2.x, then it is tested under 3.x. The exact environment is identified and mentioned in the reported bug.

- **Create connectivity problems:** The iPhone app being tested depends on the internet connectivity. So it is necessary to test for the degraded or unavailable connectivity. It is easy to make connectivity unavailable by simply turning on Airplane Mode. To degrade connectivity, especially on Edge or 3G, a metallic "shield" was placed on top of the iPhone.

iPhone Application Functionality



TESTING STRATEGIES (CONT.)

- **Boundary test data input:** For testing iPhone app text input function a large amount of text was copied and pasted into each text field and doing this caused functionality errors with the apps. Additionally, application errors were generated when entering the special characters such as **!@#$%^&*()_** into text fields. Furthermore, holding down letters (A, E, I, O, etc) or symbols ($, !, &, etc) on the onscreen keyboard generates a keyboard popup that includes localized and 2-byte characters. These should also be entered into text fields.
- **Employ background applications:** Applications that continue running in the background on the iPhone are Safari, iPod and Mail. There are also reminders and push notifications. These "interrupters" can affect the behavior of an application under test. Moreover, since iPhone is a device that users buy primarily to use as a phone, it is vital to test that an app can gracefully handle situations where the user receives a call or plays music, from their music library (iTunes), while the app is running. Several issues were identified in which the apps were not multi-tasking smoothly under the above circumstances.

TOOLS AND TECHNOLOGIES

TECHNOLOGIES:
Programming Language: Objective C, JavaScript
Framework: Cocoa
Operating System: MAC OS
Platform: Xcode
Data transfer protocol: JSON
Web Programming languages: AJAX, HTML, CSS

TOOLS USED:

- MSTest - Service automation
- HTTP Resource Test
- FireBug
- IE Development Toolbar
- JSONVIEW
- FoneMonkey
- WireShark
- Live HTTP Headers
- YSlow and Memory Sweep