Scan Quest

By Jon Keilholz

Pat Ruskowski

Submitted to the Faculty of the
School of Information Technology in
Partial Fulfillment of the
Requirements for the Degree of
Bachelor of Science in Information
Technology

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2016

Contents

Abstract

Scan Quest is an adventure game for the Android platform where the player captains a space exploration mission to the deepest edges of the universe. Traditionally, video games don't engage the user outside of the video game world. Scan Quest aims to fix this shortcoming by allowing the player's interactions in the real world to influence the in-game world. The player uses the Android's camera feature to scan UPC barcodes in their day-do-day life, which incorporates real world information into the virtual universe. The scanned UPC bar codes are used to unlock power-ups for the player's ship, allowing them to progress further, beating enemies along the way, and score higher on our in-game leaderboards.

Introduction

Scan Quest is an Android game that appeals to the casual gamer. The user progresses through multiple levels of the game with the intent of simply getting to the final level. The user utilizes their device's camera to unlock items and upgrades to aid them in their quest to get to the final level.

Problem Description

The project involves two core pieces; the actual client side of the game on the android device and a back end server. The client side is programmed in Java and utilizes the bar code scanning API called Zebra Crossing. The client device reads in barcodes and user inputs then pass that data to the back end server for processing. The back end is used to process barcode inputs as well as run checks to make sure the user is unable to cheat the game by doing things like resetting the client devices time settings. The back end server also stores user progress in a SQL Database along with logs.

Solution

Simply put the idea of Scan Quest is "Play-Scan-Win", the solution to our problem and/or observation is to provide kids or a casual gamer a fun activity to do while shopping or lounging at their desk – by bringing the interaction from just being in the virtual world on the phone to a quest in the physical world as well by searching for valuable barcodes. We wanted our app to first and foremost be fun to use to attract a user following. The second solution is to allow brands raise brand or product awareness, this plays second to being "fun" – after all without fun this app is just a fancy barcode scanner with no users. This app can be used to promote a specific product line, marketing, or even used similarly to the Kroger Plus card by running some data analytics on shoppers spending habits to offer the best supply, etc.

User Profile

Figure 1 below outlines our expected user profile. The user profile is the typical intended user, functionality, user experience, frequency of use, as well as intended user actions and interactions.

Potential Users	Software and Interface Experience	Task Experience	Frequency of Use	Key Interface Design Requirements that the Profile Suggests
Children	Codemon	Ability to Operate the Android Camera	Daily – Weekly depending on user interest	Main Menu – -- Menu Activity Buttons in grid layout
Casual Gamers	Candycrush			Camera Scanner – Camera Interface Activity
	Monster Rancher			Battle – a map with a path and dots to talk to as “levels” or “battles”
				Character Info – Image of avatar with specs and ability to equip new items
				Friends – List of friends who you “follow” to compare scores with

Figure 1: Scan Quest User Profile

Timeline

Gantt Chart

Figure 2 below is our final timeline for our project along with our deliverables for the Tech Expo.

ID	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	**Phase 1**	**26 days**	**Mon 8/24/15**	**Mon 9/28/15**		
2	Come up with Project idea - Establish a problem and a solution	21 days	Mon 8/24/15	Mon 9/21/15		
3	**Meet with Ms. Carew**	**1 day**	**Mon 9/21/15**	**Mon 9/21/15**		
4	Project Approved					
5	**Phase 2**	**69 days**	**Mon 9/28/15**	**Thu 12/31/15**		
6	Prepare and Deliver Elevator/Pitch Speech	6 days	Mon 9/21/15	Mon 9/28/15		
7	Determine Scope and expectations for December	6 days	Mon 9/28/15	Mon 10/5/15		
8	Start Working on Database and Web API restFul Service - Patrick	41 days	Mon 9/28/15	Mon 11/23/15		
9	Get Android Client to Read barcode and setup basic scoring Algorithm - Jon	41 days	Mon 9/28/15	Mon 11/23/15		
10	Deploy the DB and ASP.Net Web API solution to the server.	2 days	Tue 11/24/15	Wed 11/25/15		
11	Get the Client and Server communicating	4 days	Wed 11/25/15	Sat 11/28/15		
12	Give Demo	1 day	Mon 11/16/15	Mon 11/16/15		
13	**Phase 3**	**65 days**	**Fri 1/1/16**	**Thu 3/31/16**		
14	Add Authentication and clean up SQLite DB actions.	32 days?	Fri 1/1/16	Mon 2/15/16		
15	Improve DB, Web API based on testing- Patrick	43 days	Fri 1/1/16	Tue 3/1/16		
16	Work on Android App/UI - Jon/Pat	43 days	Fri 1/1/16	Tue 3/1/16		
17	Deploy app for User Testing first half of March	15 days	Tue 3/1/16	Sat 3/19/16		
18	Review feedback from Alpha Testing -- Make fixes were possible otherwise document future improvements.	10 days	Sat 3/19/16	Thu 3/31/16		
19	**(Tentative) Consult with a third party to get a digital artwork for possible battle scenes.**	**21 days**	**Thu 3/3/16**	**Thu 3/31/16**		
20	Add battle scene digital artwork					
21	**Phase 4**	**12 days**	**Fri 4/1/16**	**Sat 4/16/16**		
22	Deploy Final Changes to Web Server for DB and Web API	2 days	Fri 4/1/16	Mon 4/4/16		
23	Finalize Android App	2 days	Fri 4/1/16	Mon 4/4/16		
24	Present at Expo	6 days	Fri 4/8/16	Fri 4/15/16		

Figure 2: Our timeline with our established deliverables for the IT Expo.

Deliverables

In time for Tech Expo we planned to have a fully functional data layer that spans from the SQLite DB on the phone to the Cloud Server at AWS. We have successfully implemented security and authentication using OAuth 2 with our own STS, or Secure Token Service. The gameplay is limited to one world with five levels. The user can create a username, if they are a new user. The user can then log in, scan items, equip gear, and battle against the enemy computer characters. We also produced a better user experience from User Testing in March.

Technical Elements

The technical terms on the server-side aspects of this project are mostly Software as a Service hosted on Amazon Web Services (AWS) in the form of a RESTful Web API. The client-side aspect is a native Android Application written in Java. The software aspect dealing with programming languages is SQL for the database in Microsoft SQL Server, C#.Net/ASP.Net for the hosted Web Services RESTFul Service which is hosted though Microsoft Internet Information Services, and Java for the client Android application. Included is also the Zebra Crossing Barcode Scanning API, which inputs the barcode strings, as well as Google GSON library to convert data from objects to JSON to send "through the pipe" from the Android Client to the AWS Web API.

Budget

Here is a chart on estimated expenses of year one. Our total expenses for the project was $414. The budget is subject to change as the project progresses as well as different years having different rates. Amazon Web Services allows us to have a first year being a "free trial" and only paying for a domain name, while the second year we would have to pay for uptime. The current costs and expenses are mostly for development. Phase 2 and Year 2 will be more geared on maintenance, service, and update costs.

Name of Expense	**Cost**
Amazon Web Services Hosting	~140
Domain Name	$12 a year (Amazon Web Services/Route53)
Dev Android Device	$90

Coding/Research Hours **(Payscale.com)**	$70,000 / yr (Junior/Mid-level Software Developer)
Visual Studio Professional 2015	$500 (Microsoft Visual Studio Marketplace)
Microsoft SQL Server Express	Free
Third Party Graphic Designer	$180

*Microsoft Figures from Microsoft MSDN

Table 1: Actual Budget for the project.

Issues and Problems

Issues in the project stemmed from inexperience. While one member was familiar with programming for Android in Java, our project involves libraries and many aspects of the Java/Android API that he had little to no experience with. We also were working with the SQLLite database on the Android client which is a new area for both of us. The data on the SQLLite Database and Web Server will is matched 1-1 and syncs at various parts or "checkpoints" throughout the game. On the server side development, one developer is familiar with C#.Net and ASP.Net but did not have any knowledge or experience of OAuth 2 and security. Successfully securing our Web API as well as creating logins was one of the more challenging aspects on the server side.

Proposed Architecture Solution Diagram

Figure 3 below outlines our tech stack for Scan Quest and also shows how the various pieces work together.

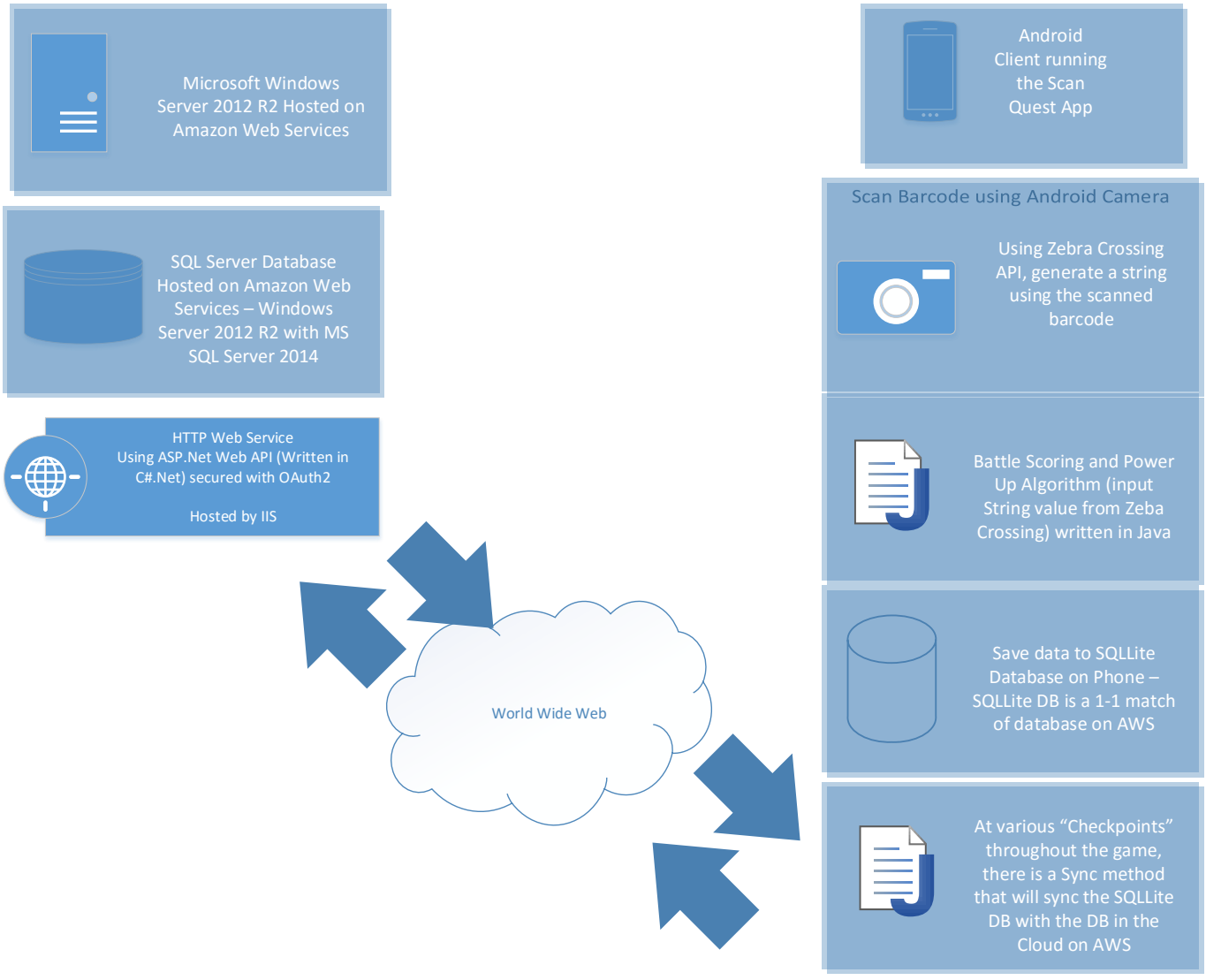

Figure 3: The Architecture diagram for Scan Quest

Proposed & Preliminary Testing Plan

Overview

The testing plan of Scan Quest will consist mostly of dealing with our web services handling multiple users at a time as well as the ability of the user to login, successfully scan items, battle, log out, and log back in and have the data persisted from session to session.

Scope

It is important for Scan Quest Web API to be able to handle multiple transactions at a time with little to no data loss or locking. There are various "end points" or queues to launch a sync from the client SQLite DB through the web services to the database hosted on Amazon Web Services.

Objective

Our biggest challenge will be persisting data after changes are made in the game. The player stats are always changing after almost any activity and a successful sync from client to server is crucial. We also want to verify the app is not hogging resources on the phone such as the network or camera before/after use or when on "pause" by running in the background.

Tested Function	Steps to Reproduce	Expected Results	Notes
Create Account	Open App, press "Create Account"	Enter desired username and password If username is not already in use and password is valid, user will be notified of success	

Login	Open App, enter Username and Password	Using valid username and valid password, user should be able to sign in and be brought to main menu with player data persisted OR Invalid username/password will trigger a message to enter valid username and password	
Scan Item	Go to Scan Tab	Camera activity should load with barcode scanner ready to scan a barcode --- once barcode is scanned it will return to the main menu and equip the player with an item or power-up in some shape or form.	Log out, log back in. See if the scanned power up is persisted. *try and use camera before and after scanning to make sure the app does not keep the camera constantly busy/in use
Equip Item	Go to Equip tab	Should have a detailed list view of items the player can be equipped with	After you equip item, log out and log back in and verify it is persisted.
Battle	Go to Battle Tab	User should be presented with the world and battle screen. Should be able to battle, win or lose the data should be persisted.	After win or loss, log off and log back in and see if data is persisted.
Top 10	Go to Community Tab	Shows top 10 players of game – possibly be able to add ability to follow friends playing the game*	

Table 2: Quality Assurance User Testing plan we have in place for testing basic, deliverable functionality of Scan Quest.

Future Enhancements/Lessons Learned

One of the major enhancements made from the first iteration to now was instead of passing the Player/Character objects from activity to activity using the parceable interface. We implemented the SQLite database on the client in order to persist data which is a lot more efficient. The parceable interface is not a very resource-effective way to pass data activity to activity and can get very complex. The SQLite DB on the client is a 1-1 match to the cloud database and syncs at various times throughout the progression of the game/app. We will also created a STS – Token generating service using the SQL Server database and User Table for OAuth 2 security and login data. Our last elements on the app before we release for Expo are to focus on the graphics and overall UI/animation of the app and gameplay as well as prevent data locking and preventing data loss between the client and server.

Conclusion

The development of Scan Quest has been a learning process through trial and error. Both members have experience in Java, C#.net, and ASP.Net but adding new libraries and security into the mix added for another level of complexity. Scan Quest is one of a small breed of unique apps to raise product awareness, run data analytics on shoppers, and give kids/casual games a fun interactive way to shop. The game takes on new light by allowing the player to not only go on a virtual quest but also a physical quest as they Play, Scan, and Win in order to progress through multiple levels of the game with the intent of simply getting to the final level.

References

"Junior Software Engineer Salary (United States)." Junior Software Engineer Salary. Accessed December 4, 2015. http://www.payscale.com/research/US/Job=Junior_Software_Engineer/Salary.

"Visual Studio Marketplace." Visual Studio Marketplace. Accessed December 4, 2015. https://marketplace.visualstudio.com/#VSSubscriptions.