Willie's Cyles

Cycle 1 Report

By

Ethan Coggin

John Boggan

Dillon Clary

Sean Walker

Shernovius Bennett

Submitted in partial fulfillment of the requirements for

COMP 4710 Senior Design

to the Department of Computer Science and Software Engineering,

Samuel Ginn College of Engineering, Auburn University

Auburn, Alabama

March 5, 2015

Table of Contents

1 Executive Summary (System Metaphor)

The Willie's Cycles project is comprised of two mobile applications - one for iOS, and one for Android. The purpose of the mobile applications is to allow clients of Willie’s Cycles to search for and purchase parts from Willie's vast inventory of motorcycle parts.

A user can search for parts by part name, make, and year. Part name refers to the name of the particular type of part the user is looking for, such as headlight, frame, etc. Make refers to the brand of motorcycle that the part is from. Year refers to the production year of the motorcycle that the part is from. The search will return a list of all parts that match the search criteria along with a listed price if one is available.

Once a user has searched for a part, they will have the option to purchase that part for the listed price (if a price is listed). Payments will be handled through PayPal. If a price is not listed, the user will have the option to request a quote for the part. Quote requests will be handled via email correspondence between Willie’s Cycles and the user.

2 Project Introduction

Willie’s Cycles is a motorcycle salvage yard located in Camp Hill, Alabama. Our project is to make a customer-facing application for Willie’s Cycles that will allow users to search for and purchase parts from their inventory of motorcycle parts. The purpose of this project is to allow Willie’s Cycles to improve their e-commerce presence and to expand their sales efforts. Our main goals for this semester are to setup an ASP.NET Web API server to facilitate querying their parts database, and to create two customer-facing mobile applications, one for iOS and one for Android. The goal with the mobile applications is to allow users to search for parts, purchase parts that have a listed price, and request quotes for parts that do not have a listed price.

2.1 Previous Development

Our primary goals for the architectural spike were to determine an effective communication system, establish user stories, create an architectural design, and identify technologies necessary for the project. We elected to use Google+ Hangouts as our communication system to allow asynchronous, archived conversations and to reduce our need for scheduled meetings. Additionally, we established our initial user stories that we used to guide our development during Cycle 1.

Our initial architectural design involved two customer-facing mobile applications and a client-facing application for, though we have since revised our design to only the two customer-facing apps. To faciliate communication between the parts inventory and the customer-facing apps, we decided to create API endpoints for the frontend apps to use.

As far as the technology stack, we decided to ASP.NET Web API for our web service because it is self-documenting and provides a lot of boiler plate code for handling networking, which significantly reduces the time and effort required to get the web service up and running. The mobile applications will be developed using a tool called Xamarin to allow for maximum codeshare, including things like business logic, payments, tests, and MVVM models and view models.

2.2 Intent This Cycle

Our primary goal for this cycle was to complete our Web API to pave way for development of the mobile applications in cycles 2 and 3. The primary tasks involved with this part of the product were to first get the Web API communicating with the Microsoft Access database and then to enable the Web API to successfully query for parts from the database with specific parameters. This component is essential to our project because the majority of the functionality of our mobile applications is tied to the Web API (and the database by extension).

Our secondary goal for this cycle was to have both of the mobile applications successfully connecting to the Web API and searching for parts by part name, make, and year. In this way, we would have the three components of our project - the database, Web API, and mobile applications - all hooked together and communicating with each other. This would be purely for functional purposes and would not involve any significant app screen design, but it paves the way for further front-end design in cycle 2.

Our stretch goal for this cycle was to work on interfacing with PayPal for logins. We did not actually work on integrating PayPal with our product, but we did further research into utilizing PayPal for this service. Based upon this research, we have opted against using PayPal for logins because it wouldn't really function as an account system in the way that we need it to. With PayPal, it would be difficult to maintain an account identity outside of the application itself, so it could not be used to maintain account information and status for our system. As such, we will be adopting a different approach in cycle 2.

2.3 Future Work

In our next cycle, we plan to address the following tasks:

- Develop our final user interface designs from our wireframe mockups
- Incorporate our UI design into the mobile applications
- Integrate the PayPal SDKs with the mobile applications to handle payments
- Incorporate a build system and unit testing

These tasks are the necessary steps required to complete our mobile applications in terms of minimum viable product. Our goal for the next cycle is to have the MVP versions of our applications completed and ready for submission to their respective application stores. This is primarily intended to accommodate for Apple's stringent review process and to guarantee that we will have a version of our applications publicly available and ready for download by the end of the semester.

Our intent is to utilize cycle 3 primarily for polishing the applications and improving the user experience. We will also attempt to incorporate any other desired functionality of stretch goal as we are able to.

3 Requirements / User Stories

3.1 User Stories

3.1.1 Search for Parts

Summary:	As a customer, I can search the available inventory of Willie's Cycles so that I can find certain parts.
Description:	The user enters search parameters and presses the search button. If the user has selected a part name, make, and year the search will be attempted; otherwise, the user will be notified that they have left required fields blank. A list of parts matching the user's search criteria will be displayed.
Hours:	Total Planned: 40
	Planned this cycle: 40

Total Actual: 18.25

Actual this cycle: 18.25

Coder: Entire Team

Tester: John B., Sean W., Shernovius B.

Reviewer: Entire team

Status: Adversarial Development

3.1.2 Select a Make

Summary: As a customer, I can select a make of motorcycle that Willie’s Cycles carries parts for to narrow down the search results.

Description: The user selects a make from the dropdown of available motorcycle part makes. Valid options are Honda, Kawasaki, Suzuki, and Yamaha.

Hours: Total Planned: 5

Planned this cycle: 2

Total Actual: 1.5

Actual this cycle: 1.5

Coder: John B., Sean W.

Tester: Ethan C., John B., Sean W.

Reviewer: To be determined

Status: Collaborative Development

3.1.3 Select a Year

Summary: As a customer, I can select a year of motorcycle part that Willie’s Cycles carries to narrow down the search results.

Description: The user selects a year from the dropdown of available motorcycle part years. Valid options are include the range of years from the earliest year of a listed part in the inventory to the current year.

Hours:	Total Planned: 5
	Planned this cycle: 2
	Total Actual: 1.5
	Actual this cycle: 1.5
Coder:	John B., Sean W.
Tester:	Ethan C., John B., Sean W.
Reviewer:	To be determined
Status:	Collaborative Development

3.1.4 Select a Part Name

Summary:	As a customer, I can select the part name of motorcycle part that Willie's Cycles carries to narrow down the search results.
Description:	The user selects a part name from the dropdown of available motorcycle parts. Valid options are part names that appear in the inventory.
Hours:	Total Planned: 5
	Planned this cycle: 2
	Total Actual: 1.5
	Actual this cycle: 1.5
Coder:	John B., Sean W.
Tester:	Ethan C., John B., Sean W.
Reviewer:	To be determined
Status:	Collaborative Development

3.1.5 Change the Search Year

Summary:	As a customer, I can change the year of my search so that I can view similar search results without having to clear the search and start over.

Description:	The user can change the year of the search by opening a dropdown of other available years and selecting a year from it. The search results will be updated accordingly.
Hours:	Total Planned: 5 Planned this cycle: 0 Total Actual: 0 Actual this cycle: 0
Coder:	To be determined
Tester:	To be determined
Reviewer:	To be determined
Status:	Unstarted

3.1.6 Clear the Search

Summary:	As a customer, I can clear the search so that I can search for a different part.
Description:	The user can clear the search results. All results will be removed from the list and the user will be returned to the search screen. The search parameters will be cleared.
Hours:	Total Planned: 5 Planned this cycle: 0 Total Actual: 0 Actual this cycle: 0
Coder:	To be determined
Tester:	To be determined
Reviewer:	To be determined
Status:	Unstarted

3.1.7 Purchase a Part

Summary:	As a customer, I can purchase a part with a list price.

Description: The user can purchase a part if it has a listed price. If the user purchases the part, a bill of sales will be sent to Willie's Cycles and the part will be removed from the database.

Hours: Total Planned: 20

Planned this cycle: 0

Total Actual: 0

Actual this cycle: 0

Coder: Ethan C., John B., Shernovius B.
Tester: Dillon C., Sean W.
Reviewer: Entire team
Status: Unstarted

3.1.8 Pay for a Part with PayPal

Summary: As a customer, I can pay for a part with PayPal

Description: If a user selects to purchase a part, they will be presented with a PayPal payment page where they can input their payment information to complete the purchase.

Hours: Total Planned: 20

Planned this cycle: 0

Total Actual: 2.5

Actual this cycle: 2.5

Coder: Ethan C., John B., Shernovius B.
Tester: Dillon C., Sean W.
Reviewer: Entire team
Status: Collaborative Development

3.1.9 Request Additional Information About a Part

Summary: As a customer, I want to be able to request additional information about a part.

Description: The user can select to inquire further about a part to receive more information, a photo, etc. If the user selects to inquire about a part a pre-formatted email will be opened that can then be sent to Willie's Cycles. If a part does not have a listed price, a user can request a quote from Willie's Cycles.

Hours: Total Planned: 10

Planned this cycle: 0

Total Actual: 0

Actual this cycle: 0

Coder: To be determined

Tester: To be determined

Reviewer: To be determined

Status: Unstarted

3.1.10 View About Us Screen

Summary: As a customer, I can select to view an About Us screen with additional information about Willie's Cycles.

Description: The user can select to view the About Us screen which contains information about Willie's Cycles. The About Us screen will display the address of Willie's Cycles, pertinent phone numbers, and pertinent emails. The user can choose to get directions to the address, call any of the phone numbers, or email any of the emails from this screen.

Hours: Total Planned: 5

Planned this cycle: 0

Total Actual: 0

Actual this cycle: 0

Coder: To be determined

Tester: To be determined

Reviewer: To be determined

Status: Unstarted

4 Design Documentation

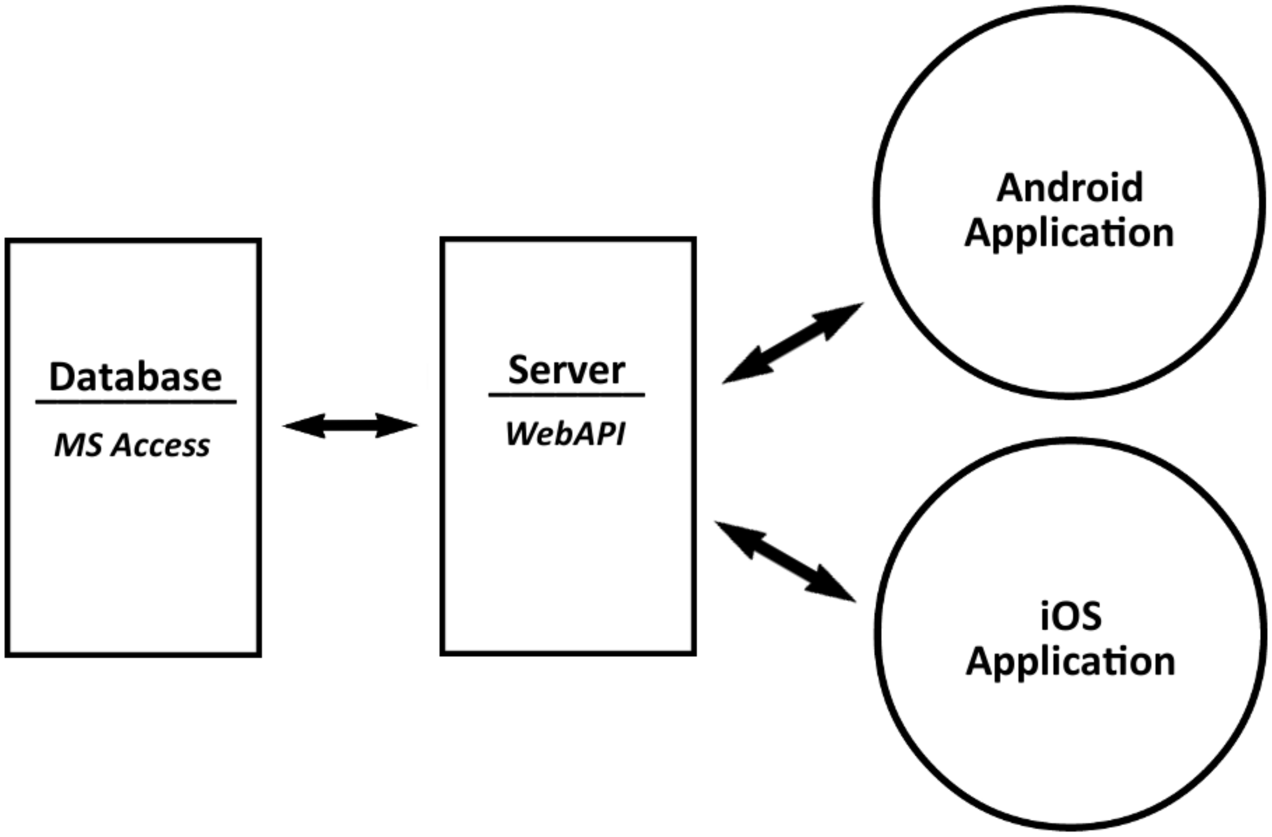


Figure shows High Level Architectural Representation

- **Architecture** -- Consists of 3 main components:
    - **Database** -- Holds all information and values for Willie's Cycles Inventory
    - **Server** -- Hosts the Web API that holds all core functionality for interaction with the database
    - **Mobile Applications** -- provides an interface for user interaction with the inventory (database), allowing searching and inquiring. Applications on 2 mobile platforms:
        - Android
        - iOS

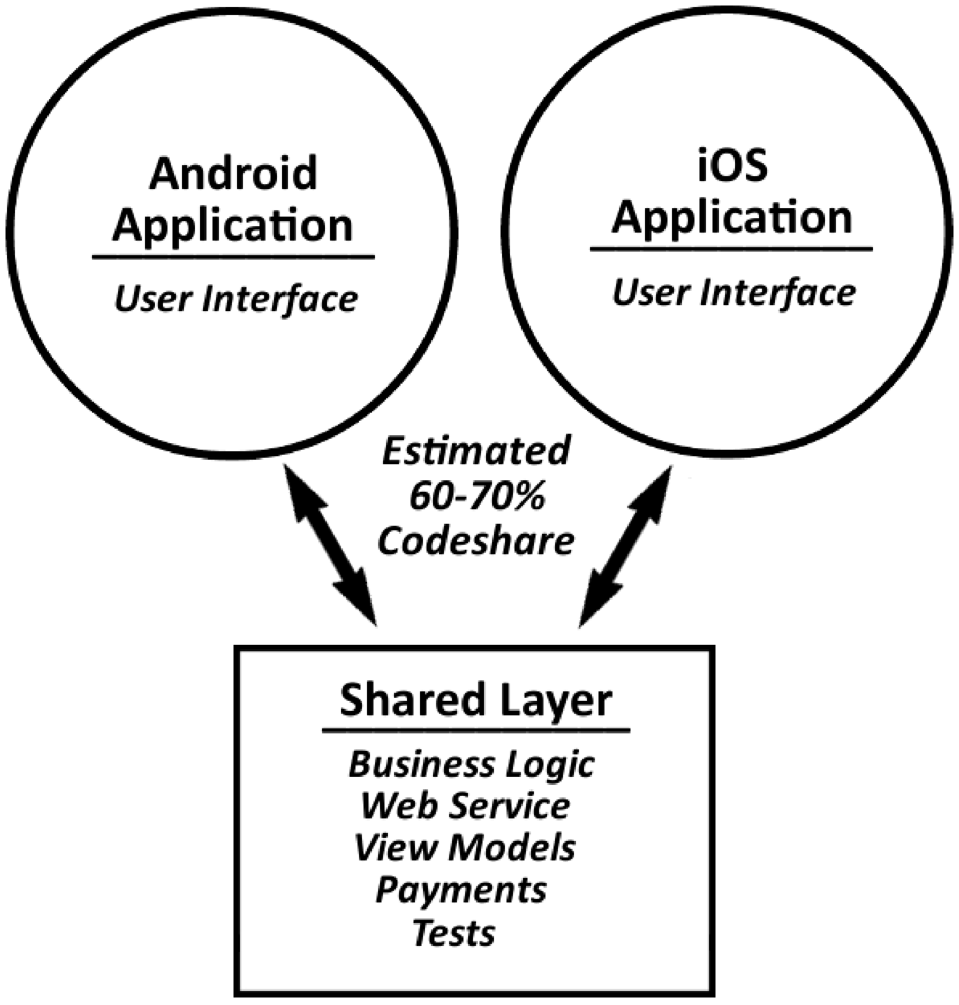


Figure shows how logic will be shared across both mobile platforms

- **<u>Structure</u>**
  - Xamarin -- used to give the ability of cross-platform development on Android and iOS using the Microsoft .Net Framework. Using a codebase in C# and software Xamarin, applications with native user interfaces could be built using shared code.
  - MVVM (Model, View, View-Model) -- this design pattern was used further allowing shared logic across both mobile platforms. Source code for Models and View-Models only need to be created once, the Model being the created Part object(s) and the View-Model being the adapters created to format and display the parts. The View is the actual UI created from the View-Models on each mobile platform, unfortunately this is platform specific and cannot be shared.

        - A “Portable” namespace was created that holds all shared logic for both mobile platforms. This namespace holds all logic for Web API calls, model for parts, adapters for view-model, and any business logic for payments and/or business/client interaction.
    - Web API -- created using the same codebase, C#, on the ASP.NET framework.
        - Communicates through HTTP requests
        - Serializes information using JSON
        - Calls predefined queries on the parts inventory database
    - Mobile Applications
        - Gives an interface for users to:
            - Search for parts based on a user specified criteria
            - Make purchase on a part with a listed price
            - Make an inquiry on any given part
        - Interacts with the Web API through HTTP requests
        - Serializes requests with JSON and receives and deserializes responses that are in JSON.
- **<u>Interfaces</u>**
    - Users will interact with the applications providing parameters such as search criteria
    - Mobile applications will then communicate with the Web API passing along any provided parameters, such as a search.
    - Web API interfaces both with the mobile applications and the inventory database
        - It receives requests from the mobile applications through http requests, search criteria parameters
        - Calls queries on the inventory database using passed parameters
        - Responds to applications with data from queries using passed parameters
- **<u>Assumptions & Dependencies</u>**
    - Dependencies
        - Xamarin properly integrating cross-platform functionality
        - Mobile applications depend on a fully functioning Web API, since all of the applications functionality relies on information on the database

        - Web API relies on the inventory database, if the database has been changed or becomes corrupt and fails to provide proper information then the Web API may return false information or fail completely
        - PayPal account is required for any purchases to be made
        - Email account would be required for an inquiries to be made

5 Management Plan

5.1 Task Assignments

The following user stories were under development in this cycle:

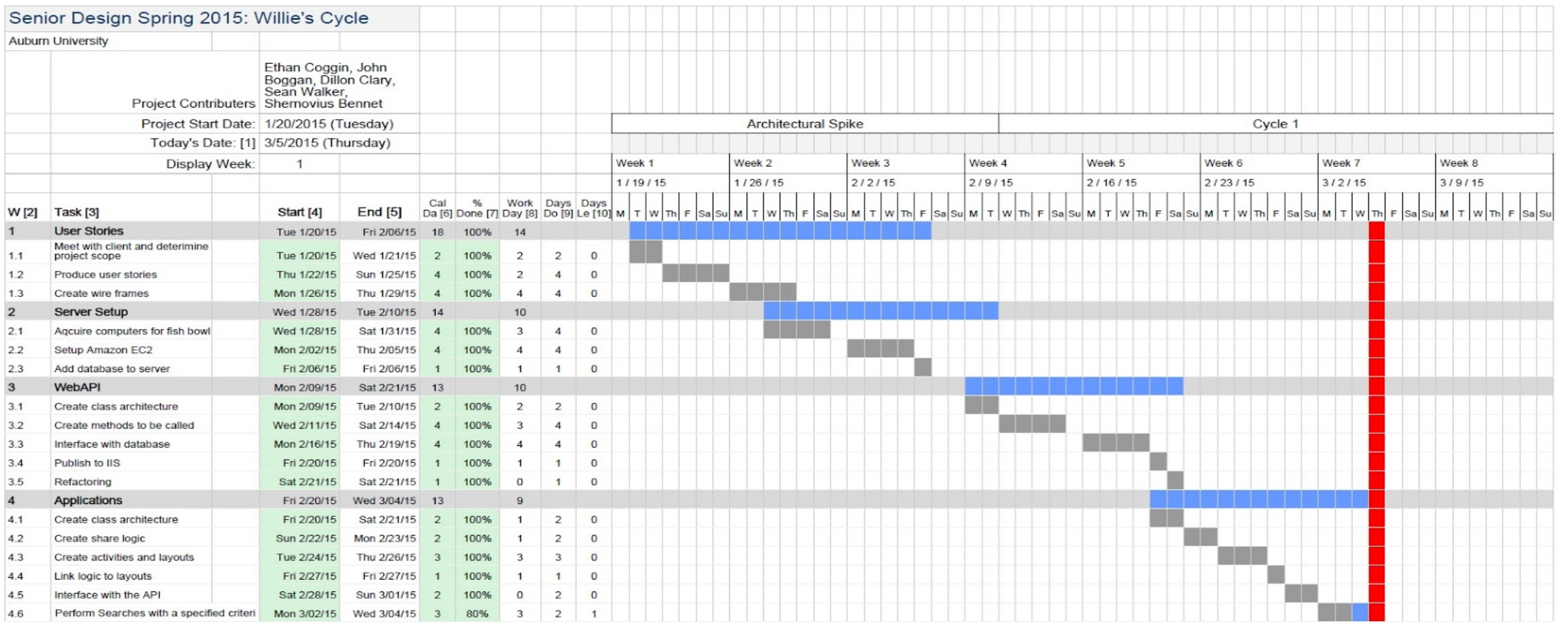
- 3.1.1 Search for Parts
    - Collaboration: Entire team
    - Code: Entire team
    - Test: John B., Sean W., Shernovius B.
    - Review: Entire team
- 3.1.2 Select a Make
    - Collaboration: John B., Dillon C., Sean W.
    - Code: John B., Sean W.
    - Test: Ethan C., John B., Sean W.
    - Review: To be determined
- 3.1.3 Select a Year
    - Collaboration: John B., Dillon C., Sean W.
    - Code: John B., Sean W.
    - Test: Ethan C., John B., Sean W.
    - Review: To be determined
- 3.1.4 Select a Part Name
    - Collaboration: John B., Dillon C., Sean W.
    - Code: John B., Sean W.
    - Test: Ethan C., John B., Sean W.
    - Review: To be determined
- 3.1.8: Pay for a Part with PayPal
    - Collaboration: Entire team

    - Code: Ethan C., John B., Shernovius B.
    - Test: Dillon C., Sean W.
    - Review: Entire team

The following tasks were under development this cycle:

- Finalize user stories
    - Assigned to: Dillon C.
- Complete wireframe mockups
    - Assigned to: Dillon C.
- Get the Web API communicating with the database (part of user story 3.1.1)
    - Assigned to: John B., Dillon C., Sean W., Shernovius B.
- Query the database with the Web API by make, year, and part name (part of user story 3.1.1)
    - Assigned to: John B., Ethan C.
- Android app: Search for parts and display list of results (part of user stories 3.1.1-3.1.4)
    - Assigned to: Sean W.
- iOS app: Search for parts and display list of results (part of user stories 3.1.1-3.1.4)
    - Assigned to: John B.
- Generate documentation for Web API
    - Assigned to: Ethan C.
- Research PayPal as an option for logins
    - Assigned to: John B.
- Research PayPal as an option for payments
    - Assigned to: Shernovius B.
- Secure Xamarin licenses
    - Assigned to: John B.

5.2 Development Schedule


Senior Design Spring 2015: Willie's Cycle

Auburn University

Project Contributers: Ethan Coggin, John Boggan, Dillon Clary, Sean Walker, Shernovius Bennet

Project Start Date: 1/20/2015 (Tuesday)

Today's Date: [1] 3/5/2015 (Thursday)

Display Week: 1

W [2]	Task [3]	Start [4]	End [5]	Cal Da [6]	% Done [7]	Work Day [8]	Days Do [9]	Days Le [10]
1	**User Stories**	Tue 1/20/15	Fri 2/06/15	18	100%	14		
1.1	Meet with client and deterimine project scope	Tue 1/20/15	Wed 1/21/15	2	100%	2	2	0
1.2	Produce user stories	Thu 1/22/15	Sun 1/25/15	4	100%	2	4	0
1.3	Create wire frames	Mon 1/26/15	Thu 1/29/15	4	100%	4	4	0
2	**Server Setup**	Wed 1/28/15	Tue 2/10/15	14		10		
2.1	Aqcuire computers for fish bowl	Wed 1/28/15	Sat 1/31/15	4	100%	3	4	0
2.2	Setup Amazon EC2	Mon 2/02/15	Thu 2/05/15	4	100%	4	4	0
2.3	Add database to server	Fri 2/06/15	Fri 2/06/15	1	100%	1	1	0
3	**WebAPI**	Mon 2/09/15	Sat 2/21/15	13		10		
3.1	Create class architecture	Mon 2/09/15	Tue 2/10/15	2	100%	2	2	0
3.2	Create methods to be called	Wed 2/11/15	Sat 2/14/15	4	100%	3	4	0
3.3	Interface with database	Mon 2/16/15	Thu 2/19/15	4	100%	4	4	0
3.4	Publish to IIS	Fri 2/20/15	Fri 2/20/15	1	100%	1	1	0
3.5	Refactoring	Sat 2/21/15	Sat 2/21/15	1	100%	0	1	0
4	**Applications**	Fri 2/20/15	Wed 3/04/15	13		9		
4.1	Create class architecture	Fri 2/20/15	Sat 2/21/15	2	100%	1	2	0
4.2	Create share logic	Sun 2/22/15	Mon 2/23/15	2	100%	1	2	0
4.3	Create activities and layouts	Tue 2/24/15	Thu 2/26/15	3	100%	3	3	0
4.4	Link logic to layouts	Fri 2/27/15	Fri 2/27/15	1	100%	1	1	0
4.5	Interface with the API	Sat 2/28/15	Sun 3/01/15	2	100%	0	2	0
4.6	Perform Searches with a specified criteri	Mon 3/02/15	Wed 3/04/15	3	80%	3	2	1

Gantt Chart (PDF of chart included with CD)

5.3 Planned Code / Feature Freeze

Our planned code freeze date for this cycle was 3/2/15.

6 Risk Mitigation

Developing for Willie's Cycle has been challenging simply because most of the systems are so far outdated; for example, their local computers are still running windows 2000. Therefore, ensuring software was backwards compatible was a huge concern. To mitigate this risk we created an Amazon EC2 instance so that we could avoid directly accessing their systems until absolutely necessary.

Another source of risk has been through Willie's Cycles' private contractor, Mr. Terry Odell. During the architectural spike, the project essentially reached a standstill while we waited on the contractor. Mr. Odell planned to move Willie's Cycles' current system, which was based on text files, over to a system which would utilize a Microsoft Access database. We were able to mitigate the risk imposed by the transition by requesting a sample database with the same format that the final database would be in. Because of this, we were able to continue developing even while we were waiting on the new database to be completed.

A prime source of risk to the project is due to the nature of handling payments and the inherent security risks therein. To avoid having to deal with payment information directly, we plan to use PayPal for payments.

Our final issue resides in teaching the employees how to use the system so that it adds value. Our goal is to make the system straightforward and user-friendly, but there are still some components that will require the employees at Willie's to perform some actions. We have made sure to let the client know what these procedures are up-front so that they are aware of and willing to carry out these procedures.

7 Test Plan and Test Procedures

7.1 Test Plan

The Willie's Cycles project group will test every component of software that it ends up shipping, from the web service to the mobile applications. Depending on the type of software involved, the testing strategy may change.

Mobile Application Test Plan

All mobile applications will be tested using both blackbox and whitebox testing methodologies. Because of the architecture we have chosen for our application, we can write unit tests for around 50-60% of our desired functionality. This includes things like standard business logic, payments, and even our view models (which provide functionality to our views via the MVVM pattern). In addition, we are writing blackbox UI tests using Xamarin Test Cloud. This allows us to test our application using thousands of devices. While this certainly provides value for iOS, it provides even more for Android due to the fragmented nature of the OS. Between the two methods, we are getting significant test coverage, both of the UX of the app, as well as the internal functionality that powers it.

Powering all of these tests is our continuous integration server, TeamCity. Following completion of a component, developers will write tests (if they haven't already using something like test-driven development). This will build on an existing suite of tests that have already been written for pre-existing functionality. Before the developer commits his code to source control, he will run these tests locally. If they pass, he will commit his code to source control (in our case GitHub). Once the code is commited to GitHub, TeamCity will automatically execute the existing test suite (for both the blackbox and whitebox tests). If any of the tests fail, the build will be rejected, the previous commit reverted, and an email will be sent to all developers . This allows us to know that our master branch is always tested and ready for new additions.

Web Service Test Plan

The web service will be tested using unit tests. While we may expand our test coverage to include full system tests of the web service as a whole, for now the focus is only on the database components that the web service uses to return data. This is because Web API is an extremely well-tested framework, and the only additions required by users are to provide data to the service. For our group, this means using the database component to interact with a Microsoft Access database. If those tests pass, we have a high confidence that our web service will not encounter service issues.

7.2 Test Procedures

7.2.1 Procedure 1: Mobile Test Procedure

Action Number	Required Actions	Expected Results	Pass / Fail?
1	Develop a new component to the software.	A new component is created and added to the existing code for the project.	
2	Run the existing test suite to ensure that your changes did not cause regressions in the software.	All tests should pass (unless you have altered previous functionality). If they don't, return to Action Number 1.	
3	Write new tests for the new component that was created in Action Number 1, and add them to the existing test suite.	All of the new tests should cover both base and edge cases and should all pass.	
4	Commit code to GitHub.	TeamCity will be triggered and run all tests again, including the UI tests that you could not run locally. If all	

		tests pass, testing procedure complete. If failure, developer will be notified via email and should return to Action Number 1.	

7.2.2 Procedure 2: Web Service Test Procedure

Action Number	Required Actions	Expected Results	Pass / Fail?
1	Develop a new component to the software.	A new component is created and added to the existing code for the project.	
2	Run the existing test suite to ensure that your changes did not cause regressions in the software.	All tests should pass (unless you have altered previous functionality). If they don’t, return to Action Number 1.	
3	Write new tests for the new component that was created in Action Number 1, and add them to the existing test suite.	All of the new tests should cover both base and edge cases and should all pass.	
4	Commit code to GitHub.	TeamCity will be triggered and run all tests again. Because all tests can be executed locally, if you tested on your machine they should all pass. If all tests pass, testing procedure complete. If failure, developer will be notified via email and should return to Action Number 1.	

8 Lessons Learned

Currently credit card information is stored as raw data on a local database for Willie's. This actually turned out to be illegal in some states (although not Alabama), so doing business with those states presents some legality issues. Moving forward any issues of this nature should be addressed immediately. Eventually we decided to include a warning in the terms of agreement which simply informs the customer of the potential hazard and avoids lawsuits. We also plan to clear out all the data every 7 days in order to minimize the security risk.

Some of our other issues were with Microsoft Access. If a system is developed using an Access database, the system hosting the database requires a license. Without the license, even if you have the actual Access file you will not be able to query it. This issue is really hard to troubleshoot since the system will not throw an error. Google was also of no assistance. This problem was literally solved with plug and play; eventually we just plugged in the right components. Another issue we ran into with Access is that if the Access database is open it cannot be read, which means our Web API is essentially non-functional if the database is open.

One idea that we attempted but had to scrap was logins through PayPal. The issue we ran into was that it would be effectively impossible to keep track of accounts outside of the app, so we could not use PayPal as an account system in the way that we wanted. This forced us to abandon the bid system that we planned to implement due to an inability to maintain account identities and status. As a replacement, we are incorporating the ability for users to contact Willie's Cycles about specific parts. This would generate a pre-formatted email that would be used to initiate an email conversation between the user and Willie's Cycles.

Finally after receiving bills we realized EC2 is not completely free, even though we have signed up under the free tier. We have yet to get to the root cost of these charges, but charges have occurred.

9 Appendix A Supporting Documents

9.1 Status Reports

9.1.1 Cycle 1 Week 1

Senior Design Status Report (Page 1)

Project Name:	Willie's Cycles		
Team Members:	Ethan Coggin, John Boggan, Dillon Clary, Sean Walker, Shernovius Bennett		
Date:	2/16/2015	**Cycle:**	1
System Metaphor:	"Everybody needs a little Willie's" - A mobile application with which users can search for and purchase parts from Willie's Cycles.		
Cycle Intent:	Establish the underlying structure and direction necessary for the mobile aspect of development in cycles 2 and 3. Complete the web service to allow interaction with the current inventory.		

		Planned			Actual		
#	User Story	Cycle planned for completion	Total planned hours	Planned hours this cycle	Status	Actual hours this cycle	Total hours
1	Finalize and approve user stories	1	20	20	Collaboration	5	5
2	Implement web interface to allow interaction with inventory	1	60	60	Collaboration	9	9
3	Skeletonize structure of mobile applications	1	20	20	Collaboration	0	0
4	Interface with PayPal	2	30	20	Unstarted	0	0
5							
		Planned Total	130	120	**Actual Total**	14	14

Senior Design Status Report (Page 2)

Team Name: **Willie's Cycles**

	User Stories				Planned	Actual									
					Planned hours this cycle	Refactor Hours		Process Hours		Customer Hours		Product Hours		Total Hours	
Member Name	Collab	Code	Test	Review		Week	Cycle	Week	Cycle	Week	Cycle	Week	Cycle	Week	Cycle
Ethan Coggin	1; 2	3; 4			24			2	2					2	2
John Boggan	1; 2	3	3		24			7.5	7.5			5	5	12.5	12.5
Dillon Clary	1; 2	3			24					2	2	2	2	4	4
Sean Walker	1; 2	3			24	1.5	1.5							1.5	1.5
Shernovius Bennett	1; 2				24	1.5	1.5	2	2	3	3			6.5	6.5
				Totals	120	3	3	11.5	11.5	5	5	7	7	26.5	26.5

Accomplishments since last status report:

We have set up our Web Service, and it is communicating with a non-live version of the MS Access database that we will be using for the parts inventory. We have created a GitHub project and set up issues to delegate the tasks that we need to work on.

Objectives for the next week:

Get Willie's Cycles to sign off on a list of functionalites and deliverables. Use this list to finalize our User Stories. Reconfigure Firewall on our server. Start working on interfacing with PayPal. Work on the app skeletons.

Obstacles encountered since last status report:

Some security measures were blocking communication with the Web Server. After T/S the system we realized the firewall settings were hindering access to the server. By adjusting the rules to allow HTTP and TCP connections, the server now allows the app to access the data.

Notes:

Risks facing the project:

We are still waiting for the third party developer licenses. Still waiting for third party developer to move Willie's Cycles' current file-based inventory system over to the new system that will utilize Microsoft Access. Also establishing the software on the local server at Willie's. The firewall configuration will have to be duplicated on their server.

COMP4710 Status Report - Member Timesheet

Project Name:	Willie's Cycles
Member:	Ethan Coggin
Week Ending:	16-Feb-15
Cycle:	Cycle 1

Team Member Work Summary

Monday	**Tuesday**	Task(s) performed:	
Date:	**2/10/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Wednesday**	Task(s) performed:	
Date:	**2/11/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Thursday**	Task(s) performed:	
Date:	**2/12/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Friday**	Task(s) performed:	
Date:	**2/13/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Saturday**	Task(s) performed:	
Date:	**2/14/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Sunday**	Task(s) performed:	
Date:	**2/15/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Monday**	Task(s) performed:	Created Status Report for this week and hosted on Google Docs.
Date:	**2/16/2015**	Result:	Status Report created and emailed.
Hours Worked:	**2**	Problems encountered:	None

COMP4710 Status Report - Member Timesheet

Project Name:	Willie's Cycles
Member:	John Boggan
Week Ending:	16-Feb-15
Cycle:	Cycle 1

Team Member Work Summary

Monday	**Tuesday**	Task(s) performed:	
Date:	**2/10/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Wednesday**	Task(s) performed:	App to Web Server Communication
Date:	**2/11/2015**	Result:	Working app to web server communication. Web API, IIS, Windows Server, and EC2 all had to be properly configured to allow incoming HTTP connections.
Hours Worked:	**5**	Problems encountered:	For some reason, the Access database refuses connections remotely via the web service, but works fine locally. Probably a security or permissions issue.
Day:	**Thursday**	Task(s) performed:	Senior Design Mac Setup
Date:	**2/12/2015**	Result:	Install all needed applications for development on the senior design Mac for our group.
Hours Worked:	**1.5**	Problems encountered:	Did not have access to this until Thursday evening, which really hurt the amount of time we could be working on our iOS app as a group.
Day:	**Friday**	Task(s) performed:	GitHub setup (.5 hours) and Web Server Deployment documentation video (.5)
Date:	**2/13/2015**	Result:	Add issues for this week to GitHub issues. Record a video explaining how to deploy a new, updated instance of the web server on our Windows Server Instance
Hours Worked:	**1**	Problems encountered:	
Day:	**Saturday**	Task(s) performed:	
Date:	**2/14/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Sunday**	Task(s) performed:	
Date:	**2/15/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Monday**	Task(s) performed:	
Date:	**2/16/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	

COMP4710 Status Report - Member Timesheet

Project Name:	Willie's Cycles
Member:	Dillon Clary
Week Ending:	16-Feb-15
Cycle:	Cycle 1

Team Member Work Summary

Monday	**Tuesday**	Task(s) performed:	Meeting with Willie's employee to discuss User Stories and user experience
Date:	**2/10/2015**	Result:	Better instruction on what Willie's wants for user interaction
Hours Worked:	**2**	Problems encountered:	
Day:	**Wednesday**	Task(s) performed:	Working on Web Server
Date:	**2/11/2015**	Result:	Web Server is communicating with app
Hours Worked:	**2**	Problems encountered:	
Day:	**Thursday**	Task(s) performed:	
Date:	**2/12/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Friday**	Task(s) performed:	
Date:	**2/13/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Saturday**	Task(s) performed:	
Date:	**2/14/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Sunday**	Task(s) performed:	
Date:	**2/15/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Monday**	Task(s) performed:	
Date:	**2/16/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	

COMP4710 Status Report - Member Timesheet

Project Name:	Willie's Cycles
Member:	Sean Walker
Week Ending:	16-Feb-15
Cycle:	Cycle 1

Team Member Work Summary

Monday	**Tuesday**	Task(s) performed:	
Date:	**2/10/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Wednesday**	Task(s) performed:	
Date:	**2/11/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Thursday**	Task(s) performed:	
Date:	**2/12/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Friday**	Task(s) performed:	
Date:	**2/13/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Saturday**	Task(s) performed:	
Date:	**2/14/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Sunday**	Task(s) performed:	
Date:	**2/15/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Monday**	Task(s) performed:	Added Documentation to existing code. Cleaned up auto generated site and added project details.
Date:	**2/16/2015**	Result:	Site shows project description and links to GitHub and google docs. Some documentation added to WebAPI
Hours Worked:	**1.5**	Problems encountered:	Not a problem just forgot some parts so had to do some more refactoring

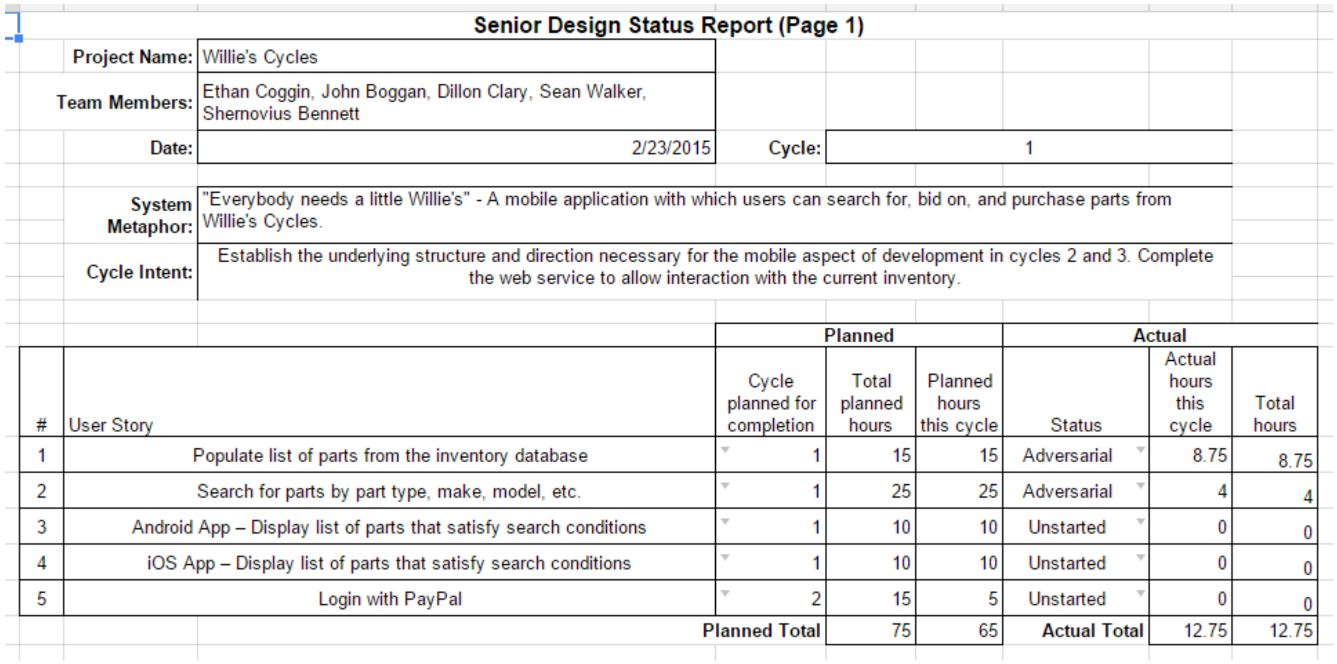
COMP4710 Status Report - Member Timesheet

Project Name:	Willie's Cycles
Member:	Shernovius Bennett
Week Ending:	16-Feb-15
Cycle:	Cycle 1

Team Member Work Summary

Monday	**Tuesday**	Task(s) performed:	
Date:	**2/10/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Wednesday**	Task(s) performed:	Meet with Dan to adjust requirments and update team changes
Date:	**2/11/2015**	Result:	Adjustment of project goals
Hours Worked:	**3**	Problems encountered:	
Day:	**Thursday**	Task(s) performed:	Reconfigure server firewall settings
Date:	**2/12/2015**	Result:	Allow app to talk directly to server
Hours Worked:	**1**	Problems encountered:	Intially app wasn't able to interact with server because of firewall settings
Day:	**Friday**	Task(s) performed:	
Date:	**2/13/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Saturday**	Task(s) performed:	
Date:	**2/14/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Sunday**	Task(s) performed:	
Date:	**2/15/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Monday**	Task(s) performed:	
Date:	**2/16/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	

9.1.2 Cycle 1 Week 2



Senior Design Status Report (Page 1)

Project Name:	Willie's Cycles
Team Members:	Ethan Coggin, John Boggan, Dillon Clary, Sean Walker, Shernovius Bennett
Date:	2/23/2015
Cycle:	1
System Metaphor:	"Everybody needs a little Willie's" - A mobile application with which users can search for, bid on, and purchase parts from Willie's Cycles.
Cycle Intent:	Establish the underlying structure and direction necessary for the mobile aspect of development in cycles 2 and 3. Complete the web service to allow interaction with the current inventory.

		Planned			Actual		
#	User Story	Cycle planned for completion	Total planned hours	Planned hours this cycle	Status	Actual hours this cycle	Total hours
1	Populate list of parts from the inventory database	1	15	15	Adversarial	8.75	8.75
2	Search for parts by part type, make, model, etc.	1	25	25	Adversarial	4	4
3	Android App – Display list of parts that satisfy search conditions	1	10	10	Unstarted	0	0
4	iOS App – Display list of parts that satisfy search conditions	1	10	10	Unstarted	0	0
5	Login with PayPal	2	15	5	Unstarted	0	0
		Planned Total	75	65	**Actual Total**	12.75	12.75

Senior Design Status Report (Page 2)

Team Name: **Willie's Cycles**

	User Stories				Planned	Actual									
						Refactor Hours		Process Hours		Customer Hours		Product Hours		Total Hours	
Member Name	Collab	Code	Test	Review	Planned hours this cycle	Week	Cycle	Week	Cycle	Week	Cycle	Week	Cycle	Week	Cycle
Ethan Coggin	2; 3; 5	2; 5	4	2	24			1.25	3.25					1.25	3.25
John Boggan	1; 2; 4; 5	1; 5	3	2	24			1	8.5			2.75	7.75	3.75	16.25
Dillon Clary	1; 3; 4	1; 3; 4		2	24			4.5	4.5		2		2	4.5	8.5
Sean Walker	1; 2	2	5	2	24		1.5	1.5	1.5			1	1	2.5	4
Shernovius Bennett	1; 2	1; 2		2	24		1.5		2	2	5	2	2	4	10.5
				Totals	120	0	3	8.25	19.75	2	7	5.75	12.75	16	42.5

Accomplishments since last status report:

Web server and MS Access database are now successfully communicating, previously we were not able to access remotely. We've done some preliminary wireframing for the iOS and Android apps. We have written some of the database queries for the Web API. We now have Xamarin licenses, so we have the tools necessary for our Android and iOS app development.

Objectives for the next week:

Finish setting up the Web API for parts searching. Have one screen of the Android and iOS apps done where the user can search for a part (through the Web API) and then populate a list of parts returned from that search.

Obstacles encountered since last status report:

Willie's Cycles is storing customer information in a non-secure manner that may present some security and legality issues. We all had busy schedules this week, so we were not able to get much work done this week.

Notes:

We modified our Status Report based on Dr. Chapman's comments, so our Page 1 and Planned Hours have been adjusted.

Risks facing the project:

We are still waiting for their third party developer to move Willie's Cycles' current file-based inventory system over to the new system that will utilize Microsoft Access. Also establishing the software on the local server at Willie's. The firewall configuration will have to be duplicated on their server.

COMP4710 Status Report - Member Timesheet

Project Name:	Willie's Cycles
Member:	Ethan Coggin
Week Ending:	23-Feb-15
Cycle:	Cycle 1

Team Member Work Summary

Monday	**Tuesday**	Task(s) performed:	
Date:	**2/17/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Wednesday**	Task(s) performed:	
Date:	**2/18/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Thursday**	Task(s) performed:	
Date:	**2/19/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Friday**	Task(s) performed:	
Date:	**2/20/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Saturday**	Task(s) performed:	
Date:	**2/21/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Sunday**	Task(s) performed:	Watched Pierce's video on updating the server. Downloaded Visual Studio on my new computer.
Date:	**2/22/2015**	Result:	Visual Studio downloaded and set up.
Hours Worked:	**0.5**	Problems encountered:	None
Day:	**Monday**	Task(s) performed:	Set up Status Report for this week so that each team member could enter their information. Modified Status Report based on Dr. Chapman's comments.
Date:	**2/23/2015**	Result:	Status Report set up and hosted on Google Sheets. Created a template for future Status Reports.
Hours Worked:	**0.75**	Problems encountered:	None

COMP4710 Status Report - Member Timesheet

Project Name:	Willie's Cycles
Member:	John Boggan
Week Ending:	23-Feb-15
Cycle:	Cycle 1

Team Member Work Summary

Monday	**Tuesday**	Task(s) performed:	Senior Design Windows Setup
Date:	**2/17/2015**	Result:	Succesfully installed necessary software for working with our Android app, as well as our web service.
Hours Worked:	**1**	Problems encountered:	
Day:	**Wednesday**	Task(s) performed:	Server to Access Communication
Date:	**2/18/2015**	Result:	Successfully have our web server and access database communication. Before, we got a permissions exception when trying to access remotely. Only step now is to refactor database logic!
Hours Worked:	**2.75**	Problems encountered:	
Day:	**Thursday**	Task(s) performed:	
Date:	**2/19/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Friday**	Task(s) performed:	
Date:	**2/20/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Saturday**	Task(s) performed:	
Date:	**2/21/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Sunday**	Task(s) performed:	
Date:	**2/22/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Monday**	Task(s) performed:	
Date:	**2/23/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	

COMP4710 Status Report - Member Timesheet

Project Name:	Willie's Cycles
Member:	Dillon Clary
Week Ending:	23-Feb-15
Cycle:	Cycle 1

Team Member Work Summary

Monday	**Tuesday**	Task(s) performed:	
Date:	**2/17/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Wednesday**	Task(s) performed:	Wireframing the Android and iOS apps. Updated User Stories.
Date:	**2/18/2015**	Result:	User Stories finalized.
Hours Worked:	**1**	Problems encountered:	None
Day:	**Thursday**	Task(s) performed:	
Date:	**2/19/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Friday**	Task(s) performed:	
Date:	**2/20/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Saturday**	Task(s) performed:	
Date:	**2/21/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Sunday**	Task(s) performed:	
Date:	**2/22/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Monday**	Task(s) performed:	Wireframes for iOS and Andriod.
Date:	**2/23/2015**	Result:	Wireframes finished.
Hours Worked:	**3.5**	Problems encountered:	

COMP4710 Status Report - Member Timesheet

Project Name:	Willie's Cycles
Member:	Sean Walker
Week Ending:	23-Feb-15
Cycle:	Cycle 1

Team Member Work Summary

Monday	**Tuesday**	Task(s) performed:	
Date:	**2/17/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Wednesday**	Task(s) performed:	Process on upcoming tasks
Date:	**2/18/2015**	Result:	Discussed Payment methods to use and routes to take (Paypal login) Discussed testing for current build
Hours Worked:	**1.5**	Problems encountered:	
Day:	**Thursday**	Task(s) performed:	
Date:	**2/19/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Friday**	Task(s) performed:	
Date:	**2/20/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Saturday**	Task(s) performed:	Started writing some test cases
Date:	**2/21/2015**	Result:	No tests written, but better understanding of what needs to be done and what direction to go
Hours Worked:	**1**	Problems encountered:	New to Web API's and C#. Visual Studio on my local machine wouldn't build project as a Web API to test
Day:	**Sunday**	Task(s) performed:	
Date:	**2/22/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Monday**	Task(s) performed:	
Date:	**2/23/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	

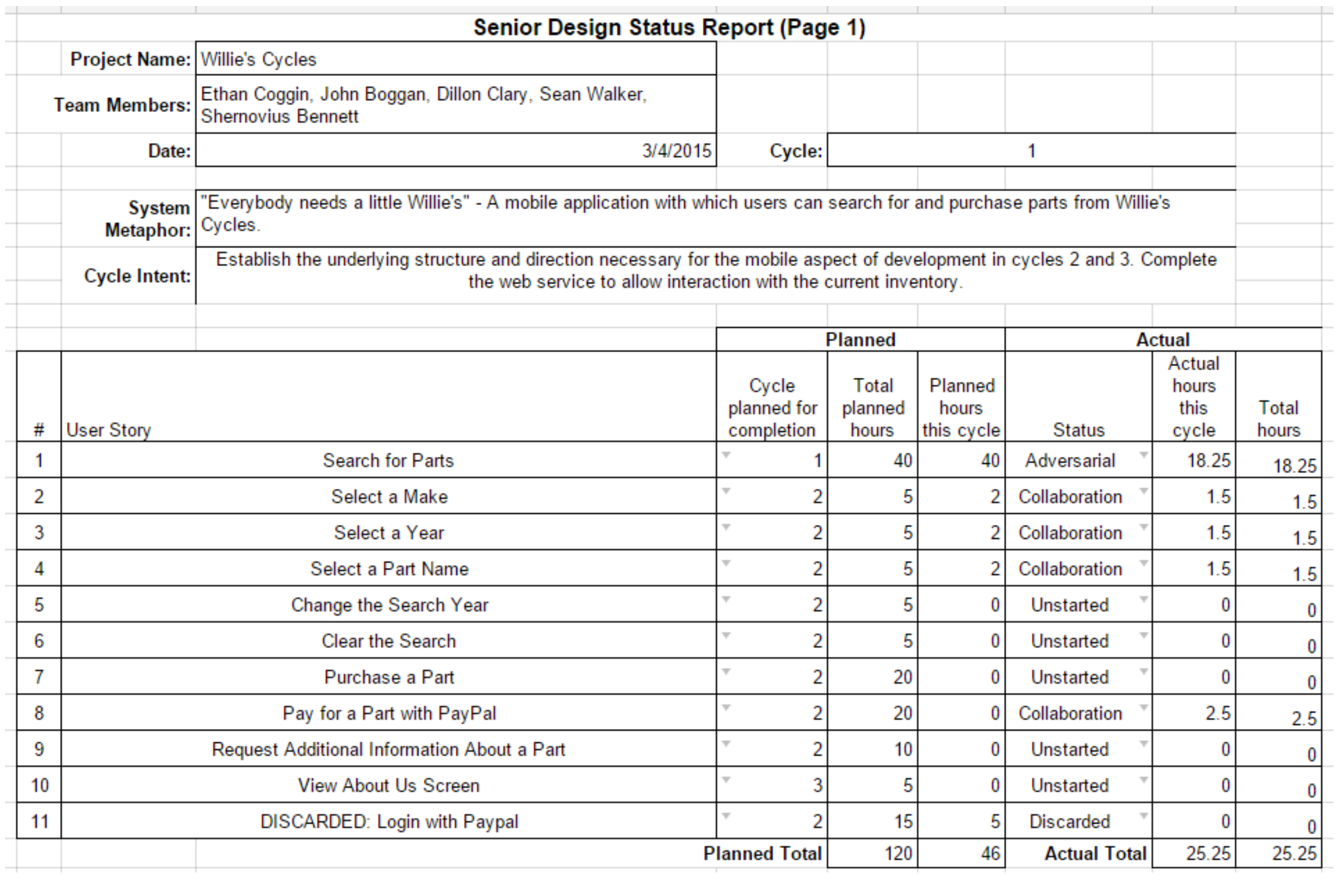
COMP4710 Status Report - Member Timesheet

Project Name:	Willie's Cycles
Member:	Shernovius Bennett
Week Ending:	23-Feb-15
Cycle:	Cycle 1

Team Member Work Summary

Monday	**Tuesday**	Task(s) performed:	Discuss Pending Security Issues with Willie's Employees
Date:	**2/17/2015**	Result:	Determine to simply include disclosure agreeement
Hours Worked:	**2**	Problems encountered:	
Day:	**Wednesday**	Task(s) performed:	Explore query options
Date:	**2/18/2015**	Result:	Determine Query statement to search database
Hours Worked:	**2**	Problems encountered:	Finding access query builder
Day:	**Thursday**	Task(s) performed:	
Date:	**2/19/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Friday**	Task(s) performed:	
Date:	**2/20/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Saturday**	Task(s) performed:	
Date:	**2/21/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Sunday**	Task(s) performed:	
Date:	**2/22/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Monday**	Task(s) performed:	
Date:	**2/23/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	

9.1.3 Cycle 1 Week 3



Senior Design Status Report (Page 1)

Project Name:	Willie's Cycles		
Team Members:	Ethan Coggin, John Boggan, Dillon Clary, Sean Walker, Shernovius Bennett		
Date:	3/4/2015	**Cycle:**	1
System Metaphor:	"Everybody needs a little Willie's" - A mobile application with which users can search for and purchase parts from Willie's Cycles.		
Cycle Intent:	Establish the underlying structure and direction necessary for the mobile aspect of development in cycles 2 and 3. Complete the web service to allow interaction with the current inventory.		

		Planned			**Actual**		
#	User Story	Cycle planned for completion	Total planned hours	Planned hours this cycle	Status	Actual hours this cycle	Total hours
1	Search for Parts	1	40	40	Adversarial	18.25	18.25
2	Select a Make	2	5	2	Collaboration	1.5	1.5
3	Select a Year	2	5	2	Collaboration	1.5	1.5
4	Select a Part Name	2	5	2	Collaboration	1.5	1.5
5	Change the Search Year	2	5	0	Unstarted	0	0
6	Clear the Search	2	5	0	Unstarted	0	0
7	Purchase a Part	2	20	0	Unstarted	0	0
8	Pay for a Part with PayPal	2	20	0	Collaboration	2.5	2.5
9	Request Additional Information About a Part	2	10	0	Unstarted	0	0
10	View About Us Screen	3	5	0	Unstarted	0	0
11	DISCARDED: Login with Paypal	2	15	5	Discarded	0	0
		Planned Total	120	46	**Actual Total**	25.25	25.25

Senior Design Status Report (Page 2)

Team Name: **Willie's Cycles**

	User Stories				**Planned**	**Actual**									
						Refactor Hours		Process Hours		Customer Hours		Product Hours		Total Hours	
Member Name	Collab	Code	Test	Review	Planned hours this cycle	Week	Cycle	Week	Cycle	Week	Cycle	Week	Cycle	Week	Cycle
Ethan Coggin	1; 7; 8	1; 7; 8	2; 3; 4	1; 7; 8	24	0.5	0.5	17.25	20.5			2	2	19.75	23
John Boggan	1; 2; 3; 4; 7; 8	1; 2; 3; 4; 7; 8	1; 2; 3; 4	1; 7; 8	24			4.5	13			2.5	10.25	7	23.25
Dillon Clary	1; 2; 3; 4; 7; 8	1	7; 8	1; 7; 8	24			8.5	13		2		2	8.5	17
Sean Walker	1; 2; 3; 4; 7; 8	1; 2; 3; 4	1; 2; 3; 4; 7; 8	1; 7; 8	24		1.5	9.5	11			5.5	6.5	15	19
Shernovius Bennett	1; 7; 8	1; 7; 8	1	1; 7; 8	24	0.5	2	13	15	2	7	2.5	4.5	18	28.5
				Totals	120	1	4	52.75	72.5	2	9	12.5	25.25	68.25	110.75

Accomplishments since last status report:

WebAPI is now returning a list of Parts with all necessary fields. We can now use the Web API to Query the database and search by Year, Make, and Model. We've added XML Documentation to the Web API web page. We have completed our UI wireframe mockups. The apps are now successfully populating a list of items from the database based upon search parameters that are passed to the API.

Objectives for the next week:

Complete the UI design from the wireframes mockups. Incorproate the new UI design into both apps to improve the user experience.

Obstacles encountered since last status report:

For some reason, the XML document used to populate the Web API documentation was removed from the project (even though it was still being generated properly) which caused the Web API to stop functioning. Upon further research we realized that using PayPal to login would not work for what we were trying to use it for, so we've had to adjust our approach accordingly.

Notes:

We updated the User Stories on Page 1 and 2 of the Status Report to match our finalized User Stories.

Risks facing the project:

We are still waiting for their third party developer to move Willie's Cycles' current file-based inventory system over to the new system that will utilize Microsoft Access. Also establishing the software on the local server at Willie's. The firewall configuration will have to be duplicated on their server.

COMP4710 Status Report - Member Timesheet

Project Name:	Willie's Cycles
Member:	Ethan Coggin
Week Ending:	4-Mar-15
Cycle:	Cycle 1

Team Member Work Summary

Monday	**Tuesday**	Task(s) performed:	Working on Web API database accessing. Adding Web API XML documentation. Prepped Status Report for this week.
Date:	**2/24/2015**	Result:	Encapsulated Database interactions in a separate class for ease of testing. Web API now returns all Part information and list of parts. Can now query by year, make, and part name. XML documentation now appears on WebAPI webpage. Status Report prepped and on Google Sheets for this week.
Hours Worked:	**2.5**	Problems encountered:	Prices appear to be doubles, but they are returned from the database as strings. XML documentation will not appear without uncommenting a specific line in the project settings.
Day:	**Wednesday**	Task(s) performed:	
Date:	**2/25/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Thursday**	Task(s) performed:	
Date:	**2/26/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Friday**	Task(s) performed:	
Date:	**2/27/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Saturday**	Task(s) performed:	
Date:	**2/28/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Sunday**	Task(s) performed:	Corrected Parts query syntax. Fixed issue with XmlDocument.xml being removed from the project. Working on Cycle Report. Working on Presentation.
Date:	**3/1/2015**	Result:	Parts query is now working.
Hours Worked:	**2.75**	Problems encountered:	For some reason the XmlDocument.xml used for creating the documentation on the WebAPI website wasn't being included in the project (even though it was being created), so the WebAPi wasn't working. I'm not sure what caused this issue as it was working previously.
Day:	**Monday**	Task(s) performed:	Working on presentation.
Date:	**3/2/2015**	Result:	Presentation completed.
Hours Worked:	**4**	Problems encountered:	None
Day:	**Tuesday**	Task(s) performed:	Working on Cycle Report.
Date:	**3/3/2015**	Result:	Progress made on Cycle Report.
Hours Worked:	**2**	Problems encountered:	None
Day:	**Wednesday**	Task(s) performed:	Working on Cycle Report. Updating Status Report with finalized User Stories.
Date:	**3/4/2015**	Result:	Cycle report completet. Page 1 and 2 of Status Report updated with new finalized User Stories.
Hours Worked:	**8.5**	Problems encountered:	None

COMP4710 Status Report - Member Timesheet

Project Name:	Willie's Cycles
Member:	John Boggan
Week Ending:	4-Mar-15
Cycle:	Cycle 1

Team Member Work Summary

Monday	**Tuesday**	Task(s) performed:	
Date:	**2/24/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Wednesday**	Task(s) performed:	
Date:	**2/25/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Thursday**	Task(s) performed:	
Date:	**2/26/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Friday**	Task(s) performed:	
Date:	**2/27/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Saturday**	Task(s) performed:	
Date:	**2/28/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Sunday**	Task(s) performed:	Added search UI to Willie's iOS client.
Date:	**3/1/2015**	Result:	Users can now query using part name, make, and year on the iOS client, and a table will display the results.
Hours Worked:	**2.5**	Problems encountered:	
Day:	**Monday**	Task(s) performed:	Presentation prep. Pull together slides and practice presentation.
Date:	**3/2/2015**	Result:	Presentation completed. Dr. Chapman said one of the best in the class!
Hours Worked:	**3**	Problems encountered:	
Day:	**Tuesday**	Task(s) performed:	
Date:	**3/3/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Wednesday**	Task(s) performed:	Working on cycle report
Date:	**3/4/2015**	Result:	Updated Testing section
Hours Worked:	**1.5**	Problems encountered:	

COMP4710 Status Report - Member Timesheet

Project Name:	Willie's Cycles
Member:	Dillon Clary
Week Ending:	4-Mar-15
Cycle:	Cycle 1

Team Member Work Summary

Monday	Tuesday	Task(s) performed:	Changed User Stories
Date:	2/24/2015	Result:	Number of user stories reduced, wireframes affected.
Hours Worked:	1	Problems encountered:	
Day:	Wednesday	Task(s) performed:	
Date:	2/25/2015	Result:	
Hours Worked:	0	Problems encountered:	
Day:	Thursday	Task(s) performed:	
Date:	2/26/2015	Result:	
Hours Worked:	0	Problems encountered:	
Day:	Friday	Task(s) performed:	
Date:	2/27/2015	Result:	
Hours Worked:	0	Problems encountered:	
Day:	Saturday	Task(s) performed:	
Date:	2/28/2015	Result:	
Hours Worked:	0	Problems encountered:	
Day:	Sunday	Task(s) performed:	Working on wireframe mockups
Date:	3/1/2015	Result:	Wireframe mockups complete
Hours Worked:	3.5	Problems encountered:	
Day:	Monday	Task(s) performed:	Working on presentation
Date:	3/2/2015	Result:	Presentation done
Hours Worked:	4	Problems encountered:	
Day:	Tuesday	Task(s) performed:	
Date:	3/3/2015	Result:	
Hours Worked:	0	Problems encountered:	
Day:	Wednesday	Task(s) performed:	
Date:	3/4/2015	Result:	
Hours Worked:	0	Problems encountered:	

COMP4710 Status Report - Member Timesheet

Project Name:	Willie's Cycles
Member:	Sean Walker
Week Ending:	4-Mar-15
Cycle:	Cycle 1

Team Member Work Summary

Monday	**Tuesday**	Task(s) performed:	
Date:	**2/24/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Wednesday**	Task(s) performed:	
Date:	**2/25/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Thursday**	Task(s) performed:	
Date:	**2/26/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Friday**	Task(s) performed:	
Date:	**2/27/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Saturday**	Task(s) performed:	
Date:	**2/28/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Sunday**	Task(s) performed:	Worked on search functionality for Android application
Date:	**3/1/2015**	Result:	Created MainActivity, which holds a simple basic unpolished UI for performing searched
Hours Worked:	**3**	Problems encountered:	Niether computers in the Senior Design room have the SDK's for Android. Had to work from personal machine, which required registering of Xamarin trial license.
Day:	**Monday**	Task(s) performed:	Worked more on Android application
Date:	**3/2/2015**	Result:	Finshed up populating spinners for search options and using parameters from spinners to perfmor searches. Met with group and created presentation for end of Cycle 1
Hours Worked:	**6**	Problems encountered:	Not skilled in C#, had to figure out some issues along the way
Day:	**Tuesday**	Task(s) performed:	
Date:	**3/3/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Wednesday**	Task(s) performed:	Worked on end of Cycle report
Date:	**3/4/2015**	Result:	Filled in design documentation, source code, and gantt chart
Hours Worked:	**6**	Problems encountered:	

COMP4710 Status Report - Member Timesheet

Project Name:	Willie's Cycles
Member:	Shernovius Bennett
Week Ending:	4-Mar-15
Cycle:	Cycle 1

Team Member Work Summary

Monday	**Tuesday**	Task(s) performed:	Meeting with Dan. Group meeting to set goals for the week
Date:	**2/24/2015**	Result:	Discussed credit card issue
Hours Worked:	**6**	Problems encountered:	
Day:	**Wednesday**	Task(s) performed:	
Date:	**2/25/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Thursday**	Task(s) performed:	Examine C# Xamarin SDK for PayPal
Date:	**2/26/2015**	Result:	Determine SDK doesn't work for both OS
Hours Worked:	**2.5**	Problems encountered:	SDK didn't work across platforms
Day:	**Friday**	Task(s) performed:	
Date:	**2/27/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Saturday**	Task(s) performed:	
Date:	**2/28/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Sunday**	Task(s) performed:	Group meeting to work on design presentation
Date:	**3/1/2015**	Result:	Design presentation
Hours Worked:	**3.5**	Problems encountered:	
Day:	**Monday**	Task(s) performed:	Group meeting to work on design presentation
Date:	**3/2/2015**	Result:	Design presentation
Hours Worked:	**4**	Problems encountered:	
Day:	**Tuesday**	Task(s) performed:	
Date:	**3/3/2015**	Result:	
Hours Worked:	**0**	Problems encountered:	
Day:	**Wednesday**	Task(s) performed:	Group meeting
Date:	**3/4/2015**	Result:	Cycle Report
Hours Worked:	**2**	Problems encountered:	

9.2 Correspondence

- Email to Dan of Willie’s Cycles:
  Dan,

  I have attached the User Stories document, which includes all the functionality that will be provided to the users of the application. Please confirm these cover the desired functionality of the app. Upon your confirmation, these user stories will be used to determine that the delivered application covers all requirements.

  ***If** at some point changes are proposed and **accepted by both parties**, then we can update the user stories and use the updated document to determine complete delivery.*

  Note: The user stories are each of the numbered sentences. The additional bullet points are extra information concerning each user story. Also, the user stories are designed to describe the user interaction requirements, and therefore do not cover the other development we're doing (concerning the server, etc.).

  Thank you!
  -Dillon Clary

  Contents of attached document:

  User Stories

  1. As a customer, I want to search the available inventory of Willie’s Cycles so that I can find certain parts.
  - *Only the items matching the search criteria’s arguments are displayed in the search results.*
  - *Search is attempted only if the search criteria for model, year, and part type have been selected.*

  2. As a customer, I can select a model to search so that I can narrow down potential items.

- *Valid models are Honda, Kawasaki, Suzuki, and Yamaha.*

3. As a customer, I can select a year to search so that I can narrow down potential items.

- *Valid options include the range of years from the earliest year of a listed part in the inventory to the current year.*

4. As a customer, I can select a part type to search so that I can narrow down potential items.

- *Valid options include the listed parts in the inventory.*

5. As a customer, I can change the year of the search results so that I can view similar results without having to clear the search.

- *Items matching the previously searched model and part type, and updated year, are displayed.*

6. As a customer, I can clear the search arguments so that I can search for different criteria.

- *Focus returned to search screen.*

7. As a customer, I want to purchase an item with a listed price so that I can buy it directly.

- *Customer is prompted to enter payment information via PayPal.*

8. As a customer, I can use PayPal to pay for an item so that I have a secure payment method.

9. As a customer, I want to request additional information for an item without a listed price so that I can receive its price.

10. As a customer, I can view an "About Us" screen so that I can view contact information.

- *Address, phone number, and email are displayed.*

9.3 Source Code

- **Web API**

HomeControllers.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace WebAPI.Server.Controllers
{
   public class HomeController : Controller
   {
        public ActionResult Index()
        {
            ViewBag.Title = "Home Page";

            return View();
        }
    }
}
```

PartsController.cs

```
using System;
using System.Collections.Generic;
using System.Data.OleDb;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Threading.Tasks;
using System.Web.Http;
using WebAPI.Server.Models;
using WebAPI.Server.Database;

namespace WebAPI.Server.Controllers
{
    public class PartsController : ApiController
    {
        // api/Parts
        /// <summary>
        /// Use Connector to connect to DB.
        /// </summary>
        /// <returns>A list of parts that was created from the DB on Willie's Server.</returns>
        public IEnumerable<Part> Get ()
        {
            Connector connector = new Connector();
            return connector.Get("SELECT * FROM Parts");
        }

        // api/Parts
        /// <summary>
        /// Use Connector to connect to DB. Formulate query to pass to DB.
        /// </summary>
        /// <param name="year">The year of the part.</param>
        /// <param name="make">The make of the part.</param>
        /// <param name="partName">The name of the part.</param>
        /// <returns>A list of parts satisfying the query conditions that was
        /// created from the DB on Willie's Server.</returns>
        public IEnumerable<Part> Get(string year, string make, string partName)
        {
            Connector connector = new Connector();
            // Query below has not been tested
            return connector.Get("SELECT * FROM Parts WHERE YR = \'" + year + "\' AND "
                + "Make = \'" + make + "\' AND " + "PartName = \'" + partName + "\'");
        }
    }
}
```

Connector.cs

```
using System;
using System.Collections.Generic;
using System.Data.OleDb;
using System.Linq;
using System.Web;
using WebAPI.Server.Models;

namespace WebAPI.Server.Database
{
    public class Connector
    {
        // api/Parts
        /// <summary>
        /// Makes connection to DB. Iterates through the DB file creating parts objects
        /// that satisfy the query conditions and adding them to a list then returns this list.
        /// </summary>
        /// <returns>A list of parts that was created from the DB on Willie's Server</returns>
        public List<Part> Get(String query)
        {
            var list = new List<Part>();
            try
            {
                var connectionString = @"Provider=Microsoft.ACE.OLEDB.12.0; Data
Source=C:\PartsDatabase.mdb;";

                using (var connection = new OleDbConnection(connectionString))
                {
                    connection.Open();
                    var command = new OleDbCommand(query, connection);
                    using (var reader = command.ExecuteReader())
                    {
```

```
                    var count = 0;
                    while (reader.Read() && count < 25)
                    {
                        var partName = reader.GetString(reader.GetOrdinal("PartName"));
                        var year = reader.GetString(reader.GetOrdinal("YR"));
                        var make = reader.GetString(reader.GetOrdinal("Make"));
                        var price = reader.GetString(reader.GetOrdinal("Price"));
                        list.Add(new Part { PartName = partName, Year = year,
                            Make = make, Price = price });

                        count++;
                    }
                }

                return list;

            }

            Console.WriteLine("It worked!");
        }
        catch (Exception ex)
        {
            Console.WriteLine("It didn't work!");
            Console.WriteLine(ex.Message);

            return new List<Part> { new Part { PartName = ex.Message, Make = ex.Source } };
        }
    }
  }
}
```

Part.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace WebAPI.Server.Models
{
    public class Part
    {
        public string Year { get; set; }
        public string Make { get; set; }
        public string PartName { get; set; }
        public string PartNumber { get; set; }
        public string Interchange { get; set; }
        public string Price { get; set; }
    }
}
```
Index.cshtml
```
<div class="jumbotron">
    <h1>Willie's Cycle</h1>
    <p class="lead">Willie's Cycle is a motorcycle salvage yard based out of Camphill, AL.
This is a project for Senior Design at Auburn University.
    The project consists of creating 2 foward facing apps to give customers the ability to
interact and search the yard's inventory and make inquiries
    and purchases.</p>
</div>
<div class="row">
    <div class="col-md-4">
        <h2>GitHuB</h2>
        <p>GitHub is being used to allow easier collaboration and allow the use of version
control.</p>
        <p><a class="btn btn-default"
href="https://github.com/pierceboggan/WilliesCycles">Learn more &raquo;</a></p>
    </div>
    <div class="col-md-4">
        <h2>Google Drive</h2>
        <p>Google Drive is being used for joint sharing of documentation and any source
materials.</p>
        <p><a class="btn btn-default"
href="https://drive.google.com/open?id=0B7mmrPPtuHx3fmhub2RtRVNCc1BYR0JhdldieTB
HUnJKcGwtSVZvVkdVdjY3NGdqR0Y1b3M&authuser=0">Learn more &raquo;</a></p>
    </div>
</div>
```
_Layout.cshtml
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>@ViewBag.Title</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
```

```
                <button type="button" class="navbar-toggle" data-toggle="collapse"
data-target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("Willie's Cycle", "Index", "Home", new { area = "" }, new {
@class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home", new { area = "" }, null)</li>
                    <li>@Html.ActionLink("API", "Index", "Help", new { area = "" }, null)</li>
                </ul>
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - Willie's Cycle</p>
        </footer>
    </div>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>
```

- **App.Portable**

API.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Net;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Threading.Tasks;
using Newtonsoft.Json;

namespace App.Portable
{
        public class API
```

```
	{
		private const string BASE_URL =
"http://ec2-54-213-92-252.us-west-2.compute.amazonaws.com:80/";

		public static async Task<List<Part>> GetTestParts ()
		{
			var client = new HttpClient () {
				BaseAddress = new Uri (BASE_URL),
			};
			client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));

			var json = await client.GetStringAsync ("api/Parts");

			return JsonConvert.DeserializeObject <List<Part>> (json);
		}

		public static async Task<List<Part>> GetParts (string partName, string make,
string year)
		{
			var request = string.Format
("api/Parts?year={0}&make={1}&partName={2}", year, make, partName);

			var client = new HttpClient () {
				BaseAddress = new Uri (BASE_URL),
			};
			client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));

			var json = await client.GetStringAsync (request);
			if(json == null){
				Part part = new Part{ Make = "No matches found." };
				List<Part> noresult = new List<Part>(){part};
				return noresult;
			} else {
				return JsonConvert.DeserializeObject <List<Part>> (json);
			}
		}
	}
}
```
Part.cs
```
using System;

namespace App.Portable
```

```
{
        public class Part
        {
                public string Year { get; set; }
                public string Make { get; set; }
                public string PartName { get; set; }
                public string PartNumber { get; set; }
                public string Interchange { get; set; }
                public string Price { get; set; }

                public override string ToString ()
                {
                        return string.Format ("{0} {1} {2}", Year, Make, PartName);
                }
        }
}
```

- **App.Android**

  Main.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">
   <Spinner
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/yearSpinner" />
   <Spinner
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/makeSpinner" />
   <Spinner
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/partNameSpinner" />
   <Button
      android:text="Search"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/searchButton" />
</LinearLayout>
```

  MainActivity.cs

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Android.App;
using Android.Content;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.OS;
using Android.Util;
using App.Portable;

namespace App.Android
{
	[Activity(Label = "Willies Cycles", MainLauncher = true, Icon = "@drawable/icon")]
	public class MainScreenActivity : Activity
	{
		Bundle bundle = new Bundle ();
		string[] searchCriteria = new string[3]; //[0] - Year; [1] - Make; [2] - Part Name

		protected override void OnCreate(Bundle bundle)
		{
			base.OnCreate (bundle);
			SetContentView (Resource.Layout.Main);
			var years = populateYears ();
			var makes = new List<string> (new string[] { "Select a Make",
"H-100R", "Honda", "Yamaha", "Suzuki", "Kawasaki" });
			var partNames = new List<string> (new string[] { "Select a Part Name",
"FRT SEAT", "FRAME", "REAR FRAME" });
			var yearAdapter = new ArrayAdapter (this,
global::Android.Resource.Layout.SimpleSpinnerItem, years);
			var makeAdapter = new ArrayAdapter (this,
global::Android.Resource.Layout.SimpleListItem1, makes);
			var partNameAdapter = new ArrayAdapter (this,
global::Android.Resource.Layout.SimpleSpinnerItem, partNames);
			bool noSearch = true;

			Spinner yearSpinner = FindViewById<Spinner>
(Resource.Id.yearSpinner);
			yearSpinner.Adapter = yearAdapter;
			Spinner makeSpinner = FindViewById<Spinner>
(Resource.Id.makeSpinner);
			makeSpinner.Adapter = makeAdapter;
			Spinner partNameSpinner = FindViewById<Spinner>
(Resource.Id.partNameSpinner);
```

```
				partNameSpinner.Adapter = partNameAdapter;
				Button searchButton = FindViewById<Button>
(Resource.Id.searchButton);
				yearSpinner.ItemSelected += (object sender,
AdapterView.ItemSelectedEventArgs e) => {
					if (/*yearSpinner.GetItemAtPosition */(e.Position) != 0) {
						searchCriteria [0] =
(string)yearSpinner.GetItemAtPosition (e.Position);
						if (searchCriteria [0] != null & searchCriteria [1] != null &
searchCriteria [2] != null) {
							noSearch = false;
						}
					}
				};
				makeSpinner.ItemSelected += (object sender,
AdapterView.ItemSelectedEventArgs e) => {
					if (/*makeSpinner.GetItemAtPosition*/ (e.Position) != 0) {
						searchCriteria [1] =
(string)makeSpinner.GetItemAtPosition (e.Position);
						if (searchCriteria [0] != null & searchCriteria [1] != null &
searchCriteria [2] != null) {
							noSearch = false;
						}
					}
				};
				partNameSpinner.ItemSelected += (object sender,
AdapterView.ItemSelectedEventArgs e) => {
					if (/*partNameSpinner.GetItemAtPosition*/ (e.Position) != 0) {
						searchCriteria [2] =
(string)partNameSpinner.GetItemAtPosition (e.Position);
						if (searchCriteria [0] != null & searchCriteria [1] != null &
searchCriteria [2] != null) {
							noSearch = false;
						}
					}
				};
				searchButton.Click += (sender, e) => {
					if (noSearch) {
						Toast.MakeText (this, "Please fill all Search Criteria",
ToastLength.Long).Show ();
					} else {
						var partsActivity = new Intent (this, typeof(PartsActivity));
						partsActivity.PutExtra ("search", searchCriteria);
						StartActivity (partsActivity);
```

```
                        }
                    };


                }
                public List<string> populateYears()
                {
                    const int yearLimit = 75;
                    int currentYear = DateTime.Now.Year + 1;
                    var years = new List<string>();
                    years.Add ("Select a Year");
                    for (int i = 1; i < yearLimit; i++) {
                        years.Add((currentYear - i).ToString());
                    }
                    return years;
                }
        }
}
```

PartsActivity.cs

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Android.App;
using Android.Content;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.OS;
using Android.Util;
using App.Portable;

namespace App.Android
{
        [Activity(Label = "Willies Cycles")]
        public class PartsActivity : ListActivity//, global::Android.App.Activity
        {
                string[] searchCriteria = null;
                protected override async void OnCreate(Bundle bundle)
                {
                    base.OnCreate (bundle);
                    searchCriteria = Intent.GetStringArrayExtra ("search");
                    var hasExtra = Intent.HasExtra("search");
                    var parts = await FetchPartsFromServer ();
```

```
			ListAdapter = new ArrayAdapter<String> (this,
global::Android.Resource.Layout.SimpleSelectableListItem, parts);
		}

		private async Task<string[]> FetchPartsFromServer ()
		{
			var parts = await API.GetParts (searchCriteria[2], searchCriteria[1],
searchCriteria[0]);

			var items = new string[parts.Count];
			for (int i = 0; i < parts.Count; i++) {
				items [i] = parts [i].ToString ();
				}
			return items;
			}
		}
	}
```

- **App.iOS**

PartsTableViewSource.cs

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using MonoTouch.Foundation;
using MonoTouch.UIKit;
using App.Portable;

namespace App.iOS
{
	public class PartsTableViewSource : UITableViewSource
	{
		List<Part> parts;

		public PartsTableViewSource (List<Part> parts)
		{
			this.parts = parts;
		}

		public override int NumberOfSections (UITableView tableView)
		{
			return 1;
		}

		public override int RowsInSection (UITableView tableview, int section)
		{
				return parts.Count;
			}

			public override UITableViewCell GetCell (UITableView tableView,
NSIndexPath indexPath)
			{
				var cell = tableView.DequeueReusableCell ("PART_CELL");

				if (cell == null) {
					cell = new UITableViewCell (UITableViewCellStyle.Default,
"PART_CELL");
				}

				var part = parts [indexPath.Row];
				cell.TextLabel.Text = string.Format ("{0} {1} {2}", part.Year, part.Make,
part.PartName);

				return cell;
			}
		}
}
```

PartsViewController.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Threading.Tasks;
using MonoTouch.Foundation;
using MonoTouch.UIKit;
using App.Portable;

namespace App.iOS
{
		public class PartsViewController : UIViewController
		{
			UITableView tableView;

			string partName;
			string make;
			string year;

			public PartsViewController (string partName, string make, string year)
			{
				Title = "Willie's Cycles";
```

```
                        this.partName = partName;
                        this.make = make;
                        this.year = year;
                }

                public override async void ViewDidLoad ()
                {
                        base.ViewDidLoad ();

                        View.BackgroundColor = UIColor.White;

                        UIApplication.SharedApplication.NetworkActivityIndicatorVisible = true;
                        var parts = await FetchPartsFromServer ();
                        UIApplication.SharedApplication.NetworkActivityIndicatorVisible =
false;

                        tableView = new UITableView {
                                Frame = new RectangleF (0, 64, View.Bounds.Width,
View.Bounds.Height),
                                Source = new PartsTableViewSource (parts)
                        };

                        View.Add (tableView);
                }

                private async Task<List<Part>> FetchPartsFromServer ()
                {
                        return await API.GetParts (partName, make, year);
                }
        }
}
```

AppDelegate.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using MonoTouch.Foundation;
using MonoTouch.UIKit;
using App.Portable;

namespace App.iOS
{
   [Register("AppDelegate")]
   public partial class AppDelegate : UIApplicationDelegate
```

```
    {
        UIWindow window;

        public override bool FinishedLaunching(UIApplication app, NSDictionary options)
        {
            window = new UIWindow(UIScreen.MainScreen.Bounds);

                                    window.RootViewController = new UINavigationController (new
SearchController ());
            window.MakeKeyAndVisible();

            return true;
        }
    }
}
```
Main.cs
```
using System;
using System.Collections.Generic;
using System.Linq;

using MonoTouch.Foundation;
using MonoTouch.UIKit;

namespace App.iOS
{
    public class Application
    {
        // This is the main entry point of the application.
        static void Main(string[] args)
        {
            // if you want to use a different Application Delegate class from "AppDelegate"
            // you can specify it here.
            UIApplication.Main(args, null, "AppDelegate");
        }
    }
}
```
SearchControllers.cs
```
using System;
using System.Drawing;
using MonoTouch.Foundation;
using MonoTouch.UIKit;

namespace App.iOS
{
	public class SearchController : UIViewController
	{
		public SearchController ()
		{
			Title = "Search";
		}

		public override void ViewDidLoad ()
		{
			base.ViewDidLoad ();

			View.BackgroundColor = UIColor.White;

			var partNameLabel = new UILabel {
				Text = "Part Name",
				Frame = new RectangleF (25, 75, 100, 20)
			};

			var partNameTextField = new UITextField {
				Frame = new RectangleF (25, 100, 100, 20)
			};

			var makeLabel = new UILabel {
				Text = "Make",
				Frame = new RectangleF (25, 125, 100, 20)
			};

			var makeTextField = new UITextField {
				Frame = new RectangleF (25, 150, 100, 20)
			};

			var yearLabel = new UILabel {
				Text = "Year",
				Frame = new RectangleF (25, 175, 100, 20)
			};

			var yearTextField = new UITextField {
				Frame = new RectangleF (25, 200, 100, 20)
			};

			var submitButton = new UIButton {
				Frame = new RectangleF (View.Bounds.Width / 2 - 50, 225,
100, 20)
			};
```

```
                        submitButton.SetTitle ("Search", UIControlState.Normal);
                        submitButton.SetTitleColor (UIColor.Blue, UIControlState.Normal);
                        submitButton.TouchUpInside += (sender, e) => {
                                var partName = partNameTextField.Text;
                                var make = makeTextField.Text;
                                var year = yearTextField.Text;

                                NavigationController.PushViewController (new
PartsViewController (partName, make, year), true);
                        };

                        View.Add (partNameLabel);
                        View.Add (partNameTextField);
                        View.Add (makeLabel);
                        View.Add (makeTextField);
                        View.Add (yearLabel);
                        View.Add (yearTextField);
                        View.Add (submitButton);
                }
        }
}
```

- **README,MD**

```
# Willy's Motorcycles

## Software Process
### Pre-Commit Checklist
#### General Items
* Builds on your local machine
* Debugging smoke test passes
* Passes all local unit tests
* Passes all local UI tests
* If applicable, unit and/or UI test were written for the code added in this commit

#### Structure and Form
* Conforms to established coding standards
* No unneeded or uncalled methods
* All variables, methods, and classes are descriptively named
* Most common cases are first in if-then loops
* Nullable data is checked before using it

#### Documentation
* Code is written in a self-documenting manner
```

* [XML documentation](https://msdn.microsoft.com/en-us/library/vstudio/b2s063f7(v=vs.100).aspx) is added above every class or method you have written
   * It's not necessary to compile your XML docs after individual commits. We will do this at the end of each cycle.

### Bugs & Enhancements
#### Bugs
When you encouter a bug, file an issue on the GitHub repository for the issue. Include a descriptive title, relevant tags, as well as how to reproduce as well as a test case, if applicable. If you suspect you know what the issue is, include this in the issue as well. Even if the issue is a fairly trivial fix, we should file a bug on it. When you commit a fix, say "Fix #" followed by the issue number. Example: "Fix #1".

#### Enhancement
If you suspect you have room for an enhancement (such as a performance improvement or feature idea), file an enhancement issue on the GitHub repository. Include a descriptive title, relevant tags, as well as a description of the enhancement, along with possible implementation strategies. When the enhancement is fully completed (not 1/2, 3/4, but all the way), commit "Fix #" followed by the issue number, along with the feature or enhancement's name. Example: "Fix #1 - Account System".

### Coding Standards
* [.NET Naming Guidelines](https://msdn.microsoft.com/en-us/library/ms229002(v=vs.110).aspx)
* [C# Coding Conventions](https://msdn.microsoft.com/en-us/library/ff926074.aspx)

- **Links:**

WebAPI
http://ec2-54-213-92-252.us-west-2.compute.amazonaws.com
To make http requests direct to port 80; http://ec2…...com:80/
GitHub
https://github.com/pierceboggan/WilliesCycles
GoogleDocs
https://drive.google.com/folderview?id=0B7mmrPPtuHx3fmhub2RtRVNCc1BYR0JhdldieTBHUnJKcGwtSVZvVkdVdjY3NGdqR0Y1b3M&usp=drive_web