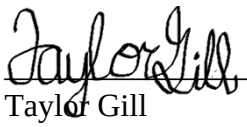
iMechanic

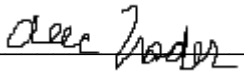
by

Taylor Gill, Alec Nader, Ryan Gigliotti

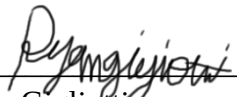
Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology



Taylor Gill _____4/13/20____
Date



Alec Nader _____4/13/20____
Date



Ryan Gigliotti _____4/13/20___
Date

_____ _____
Abdou Fall, Faculty Advisor Date

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

April 2019

TABLE OF CONTENTS

List of Illustrations

TABLES

FIGURES

ACRONYMS AND ABBREVIATIONS

API – Application programming interface

ABSTRACT

Car maintenance management can be difficult due the different components that are commonly repaired and replaced. Even if an owner has vast knowledge of their vehicle's maintenance routine, it is common to forget when a component was replaced. Consumer Reports advises that the best way to save money at the mechanic is to do proper research on specific components, and fix/replace components at the right time. iMechanic is a multi-platform web application that allows car owners to take maintenance into their own hands by keeping detailed work logs and allowing owners to efficiently organize their car's data. The user can input simple vehicle information to create their dashboard, where they can get familiar with common maintenance routines, costs, and local mechanics. When a car owner uses a proper organization management tool, they can increase the well-being of their vehicle, while also saving money from avoiding unnecessary repairs.

1. PROBLEM STATEMENT

1.1 Introduction

Driving a car is often a daily routine performed in order to get to work, school, or for leisure. Cars go through maintenance several times during their lifespan, and these maintenance stops can include repairing or replacing several components of the car. Unfortunately, it is not common for a car owner to know exactly what needs to be replaced, when, how often, or the last time work was performed. Mechanics use this to their advantage to attempt to sell unnecessary work on unknowing customers. There have been car maintenance applications that help with organization, but lack a free, multiplatform reach. We have created iMechanic, an application that will make car maintenance simplified and easy for everyone.

1.2 Project Description

iMechanic will help car owners/drivers have a better idea of their car care by tracking all maintenance work related to mileage and inform the user when a routine maintenance check is approaching. Included in that information will be other car items that will possibly need checked or replaced. It will also feature a logbook that will help the user keep track of any performed maintenance, which will also help the user not be sold on unnecessary work. iMechanic will have a modern and easy to use design that will allow users to access their data quickly and securely. The main dashboard of iMechanic will hold most of the information, making it the center of the design. Unfortunately, some of our initial features that we had considered were unable to be implemented. We had wanted to utilize two-factor authentication, but with the use of a reverse proxy instead of a web server, it made it difficult to implement.

1.3 Problem

Consumer Reports paints a story on how a mechanic can easily try and sell unnecessary work to a consumer by preying on their ignorance. According to CR, unethical mechanics will pressure car owners during safety recalls or routine maintenance to have extra work done on their car, which increases their sales [1]. To avoid spending more than needed at a mechanic, it is advised to stay educated and check the vehicle's manual for maintenance schedules. From what we have gathered so far, there is no online service to help organize and plan car maintenance that is completely user oriented. There are many parts of a car that can be inspected and replaced, and it is easy to forget previous work performed. An online feature that logs information while also informing users about future car maintenance can help a consumer be more prepared when headed to the mechanic.

1.4 Solution

iMechanic is a multi-platform web application that funnels all car maintenance into one easy to use dashboard and logbook, which lays out a maintenance plan as well as logs any and all work done to the car. The user will input information about their car: year, make, and model, and then will be given maintenance report. The user can modify their reports in the logbook with any past or recent work done on the car. When a user has expected maintenance coming up, the application will give price estimates and help the user find a local mechanic. There are car maintenance applications out there, but they do not have the large scope of iMechanic. aCar is a maintenance application that has a large focus on mileage and gas usage and is only available on Android. Openbay is a free, IOS only, application also helps organize and plan maintenance, but it is large focus is allocating quotes and helping user's compare price.

1.5 User Profile

Table 1: User profile is the user profile that was referenced during the creation of the application. This helps put into perspective how frequently our product will be used and also the different ways users may be assisted from iMechanic.

PROJECT: iMechanic, a car maintenance application.
POTENTIAL USERS: - Vehicle Owners - Anyone looking for assistance in organizing their car maintenance. - Anyone looking to increase their knowledge about routine maintenance
SOFTWARE, INTERFACE, AND RELATED EXPERIENCE: This project will be aimed towards anyone who is a car owner and is looking for help with their car maintenance. People who use iMechanic will come from all backgrounds and areas of expertise when it comes to car maintenance and using technology. The end user will not have to have any technology or maintenance skills to use iMechanic, as basic maintenance will be laid out for them. The end user may have to reference their car's manual for additional maintenance.
EXPERIENCE WITH SIMILAR APPLICATIONS: - aCar - Maintenance Reminder - OpenBay

TASK EXPERIENCE:

- Navigating a web application
- Data entry and reading maintenance logs
- Keep entries updated to reflect maintenance

FREQUENCY OF USE:

Users can use iMechanic on a daily/weekly basis if they own multiple cars and need to keep track of a large amount of maintenance. Some users may have one car and only log the oil changes every few months.

KEY PROJECT DESIGN REQUIREMENTS THAT THE PROFILE SUGGESTS:

-Modern and sleek design

-Quick and easy data access

-Interface that is friendly and appealing to all users

Table 1: User Profile

1.6 Use Case Diagram

Figure 1: Use Case Diagram presents how a standard user would interact with the application, and the application's response from the user's inputs. This is subject to change if more features are added.

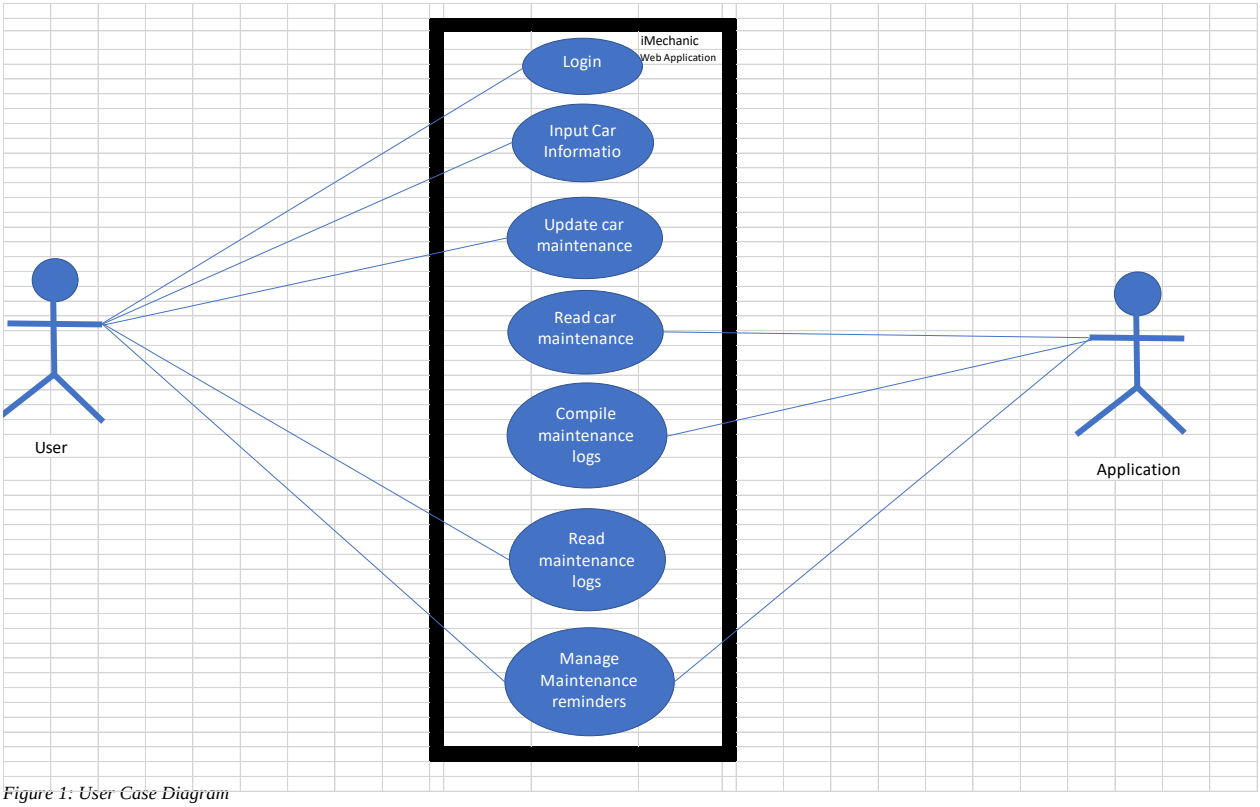


Figure 1: User Case Diagram

2. PROJECT MANAGEMENT

2.1 Budget

Figure 2: Project Budget lays out the real-word cost for application development. It would cost 12,000$ for research and development. This is all due to labor, at the estimate of 20/hrs of work on the project for 23 weeks. There is little to no other costs due to not needing to purchase a web service. Our original budget included the hours worked on the application until the end of the Spring semester. During the Spring and Fall semester we did not accumulate any other costs besides direct labor hours, at 20/hr.

Risk Identification (See Risk Types tab)						**Project Stakeholder(s)**
	Risk Rating 1-5 (5 is high)*	*Comments*	*Weight*	*Score*		All team members of the project are stakeholders
Work Effort (days)	1	Approximately 600 hours of work with research	40%	0.40		
Complexity	3	Low to no technology needed for application development	60%	1.80		
Project Risk Score:				**2.20**		

Estimate of Benefits		
If project will generate revenue, estimate 1 year here:	$	-
Select other benefits the project may bring a customer or user:		
Risk Avoidance		☑
Improved customer satisfaction		☑
Increased system availability		☐
Productivity or process improvement		☐
Reduced costs		☑

Estimated Cost Rough Order of Magnitude:							**Comments: About 20 hours per week of work for every week of the 23 weeks of senior design. Ongoing costs include any updates, bug patches, new cars end the market, etc.**
	Rate Per/Hr	**Work Effort (Hours)**	**1 X Costs**	**Ongoing Annual**			
				Rate Per/Hr	**Work Effort (Hours)**	**1 X Support Cost**	
Labor - IT	20	600	$ 12,000.00	20	150	$ 3,000.00	
Labor - External	0		$ -		0	$ -	
Software - External							
Hardware - External							
Misc.							
TOTAL			$ 12,000.00			$ 3,000.00	

5-Year ROI Analysis		
Description	**5- Year Expected**	**Conservative (1.5)**
Total Costs	**$ 27,000.00**	**$ 40,500.00**
Total Benefit	**$ -**	$0
Total Costs/Benefit Differential	**$ (27,000.00)**	
Conservative Costs/Benefit Differential	**$ (40,500.00)**	

Figure 2: Final Project Budget

2.2 Objectives/Deliverables

Table 2: Project Objectives/Deliverables shows important milestones, assignments, and deliverables. The team will regularly check and update the timeframes as needed. The dates include when the deliverables became actively worked on

MAJOR PROJECT MILESTONES (DELIVERABLES)			
FALL OF 2019 MILESTONES			
Initiation Phase	8/26/19	Team Contract Written	9/2/19
Research Phase	9/9/19	Project Abstract Drafted	9/23/19
Design Phase	9/23/19	User Profile Drafted	10/7/19
Environment Setup	9/16/19	Elevator Speech	10/7/19
Development	10/7/19	Fall Presentation	12/2/19
SPRING OF 2020 MILESTONES			
Alpha Test Phase	1/13/20	Spring Presentation	3/31/20
Regression Test	2/17/20	Tech Expo	4/14/20
Security Testing	3/5/20		
Beta Test Phase	3/9/20		
Redesign/Software Update Phase	3/16/20		

Table 2: Project Objectives/Deliverables

2.3 Project Schedule

Figure 3: Project Schedule and Gantt Chart is used as our guide throughout the project to ensure we stack on track.

Task Name	Duration (Days)	Start Date	End Date	Responsible	Status
1.0 Project Management and Deliverables	248	8/26/2019	4/27/2020	ALL	Complete
1.1 Team Building	7	8/26/2019	9/2/2019	ALL	Complete
1.2 Ideas and Brainstorming	7	8/26/2019	9/2/2019	ALL	Complete
1.3 Fall Semester Assignment 0: Team Members & Project Name	7	8/26/2019	9/2/2019	TG	Complete
1.3.1 Project Name	7	8/26/2019	9/2/2019	TG	Complete
1.3.2 Project Logo and Branding	7	8/26/2019	9/2/2019	TG	Complete
1.4 Fall Semester Assignment 1: Team Contract	21	9/2/2019	9/23/2019	ALL	Complete
1.4.1 Project Approval	7	9/9/2019	9/16/2019	ALL	Complete
1.4.2 Gantt Chart	14	9/9/2019	9/23/2019	TG	Complete
1.4.3 Work Breakdown Structure	14	9/9/2019	9/23/2019	ALL	Complete
1.5 Fall Semester Assignment 2: Project Abstract for Tech Expo	21	9/23/2019	10/14/2019	TG	Complete
1.6 Fall Semester Assignment 3: Team Contract Resubmission	7	10/7/2019	10/14/2019	TG	Complete
1.7 Fall Semester Assignment 4: User Profile	14	10/7/2019	10/21/2019	TG	Complete
1.8 Fall Semester Assignment 5: Use Case Diagram	14	10/7/2019	10/21/2019	TG	Complete
1.9 Fall Semester Assignment "3 Minute Elevator Speech"	14	10/7/2019	10/21/2019	TG	Complete
1.10 Fall Semester Assignment 6: Draft Report	14	10/21/2019	11/4/2019	TG	Complete
1.11 Fall Semester Assignment 7: Final Fall Semester Report	14	11/4/2019	11/18/2019	TG	Complete
1.12 Fall Semester Oral Presentation	14	11/4/2019	11/18/2019	TG	Complete
1.12.1 Presentation Practice	14	11/4/2019	11/18/2019	TG	Complete
2.0 Research	42	9/9/2019	10/21/2019	ALL	Complete
2.1 Software Requirements	7	9/9/2019	9/16/2019	ALL	Complete
2.1.1 Determine Front End Development Languages	14	9/9/2019	9/23/2019	RG+AN	Complete
2.1.2 Determine Back End Development Languages	14	9/9/2019	9/23/2019	RG+AN	Complete
2.2 Network Requirements	42	9/9/2019	10/21/2019	RG+AN	Complete
2.2.1 Determine Hosting Environment	14	9/9/2019	9/23/2019	RG+AN	Complete
2.2.2 Determine Database Environment	7	9/9/2019	9/16/2019	RG+AN	Complete
2.2.3 Research IP Mapping	28	9/23/2019	10/21/2019	TG	Complete
2.2.4 Research Load Balancing Systems	14	10/7/2019	10/21/2019	TG	Complete
2.3 Security Requirements	28	9/23/2019	10/21/2019	ALL	Complete
2.3.1 Determine Account Security and Authentication	21	9/23/2019	10/14/2019	RG+AN	Complete
2.3.2 Research Common Malicious Injection Methods	7	9/23/2019	9/30/2019	RG+AN	Complete
2.3.3 Research Custom Port Numbers (TCP/Layer 4)	7	9/30/2019	10/7/2019	RG+AN	Complete
2.3.4 Research Legal Disclaimers	14	10/7/2019	10/21/2019	TG	Complete
2.3.5 Research Login/Credential Storage/MFA	14	10/7/2019	10/21/2019	RG+AN	Complete
2.4 Car Research	28	9/16/2019	10/14/2019	TG	Complete
2.4.1 Determine fundamental car parts	7	9/16/2019	9/23/2019	TG	Complete
2.4.2 Research common maintenance logs	14	9/16/2019	9/30/2019	TG	Complete
2.4.3 Determine car research breadth or depth	21	9/23/2019	10/14/2019	ALL	Complete
2.5 Miscellaneous Research	28	9/23/2019	10/21/2019	ALL	Complete
2.5.1 Determine Analytics	7	9/23/2019	9/30/2019	RG+AN	Complete
2.5.2 Determine features	14	9/23/2019	10/7/2019	ALL	Complete
2.5.3 Budget Analysis	14	9/30/2019	10/21/2019	ALL	Complete
3.0 System Design	42	9/23/2019	11/4/2019	ALL	Complete
3.1 Create System Diagrams	35	9/23/2019	10/28/2019	TG	Complete
3.1.1 Create Network Diagrams	14	9/30/2019	10/14/2019	TG	Complete
3.1.2 Create Database Diagrams	21	9/30/2019	10/21/2019	TG	Complete
3.1.3 Create Interaction Diagrams	35	9/30/2019	11/4/2019	ALL	Complete
3.1.4 Create Wireframe Diagrams	35	9/30/2019	11/4/2019	ALL	Complete
3.2 Create Legal Documentation	21	10/7/2019	10/28/2019	TG	Complete
3.2.1 Draft Legal Disclaimers and Privacy Policy	21	10/7/2019	10/28/2019	TG	Complete
4.0 Environment Set-Up	140	9/16/2019	2/3/2020	AN+RG	Complete
4.1 Import Libraries for Development	7	9/23/2019	9/30/2019	AN	Complete
4.2 Setup GitHub	7	9/16/2019	9/23/2019	AN	Complete
4.3 Determine Front-End Framework	7	9/23/2019	9/30/2019	AN	Complete
4.4 Set up Database Server	28	10/7/2019	11/4/2019	AN+RG	Complete
4.4.1 Dual role as domain controller	14	10/7/2019	10/21/2019	ALL	Complete
4.4.2 Install MYSQL Server	7	10/14/2019	10/21/2019	AN+RG	Complete
4.4.3 Write Database Creation Scripts	14	10/21/2019	11/4/2019	RG	Complete
4.4.4 Link Database Tables	7	10/21/2019	11/28/2019	RG	Complete
4.4.5 Configure User Groups and Access	7	10/28/2019	11/4/2019	RG	Complete
4.4.6 Implement Audit Logging	14	10/21/2019	11/4/2019	RG	Complete
4.5 Set up Web Application Servers	21	11/4/2019	11/25/2019	RG+AN	Complete
4.6 Set up Web Application Firewall	21	11/4/2019	11/25/2019	TG	Complete
4.6.1 Create Managed Rules	21	11/4/2019	11/25/2019	TG	Complete
4.7 Set up Load Balancer	7	1/13/2020	1/20/2020	TG	Complete
4.8 Purchase Domain Name	7	1/13/2020	1/20/2020	RG+AN	Complete
4.8.1 Install Site Certificates	7	1/13/2020	1/20/2020	RG+AN	Complete
5.0 Development (Back End and Front End)	175	10/7/2019	4/6/2020	AN+RG	Complete
5.1 CreateBootstrap View Pages	14	10/7/2019	10/21/2019	AN	Complete
5.2 Create and Configure backend controllers (AJAX)	14	10/7/2019	10/21/2019	RG	Complete
5.3 Create Navigation Bar	14	10/14/2019	10/21/2019	AN	Complete
5.4 Set up API Routers	21	10/14/2019	11/4/2019	RG	Complete
5.4.1 Configure Backend APIs	14	10/21/2019	11/4/2019	RG	Complete
5.5 Design and Develop Dashboard	14	10/28/2019	11/11/2019	AN	Complete
5.6 Create Login and Set Up Authentication	14	10/14/2019	10/28/2019	AN+RG	Complete
5.6.1 Create User Registration and Confirmation	7	10/14/2019	10/21/2019	AN+RG	Complete
5.6.2 Create Forgot Password	7	10/14/2019	10/21/2019	AN+RG	Complete
5.7 Develop Features	77	10/21/2019	2/24/2019	AN+RG	Complete
5.7.1 Design and Develop Initial Report (Car Added)	21	10/21/2019	11/11/2019	AN+RG	Complete
5.7.2 Design and Develop Future Recommended Maintenance	14	10/28/2019	11/11/2019	AN+RG	Complete
5.7.3 Design and Develop Alerts for Future Maintenance	21	11/4/2019	11/25/2019	AN+RG	Complete
5.7.4 Design and Develop Information on Prices and Mechanics	21	11/4/2019	11/25/2019	AN+RG	Complete
5.7.5 Design and Develop Recommendations and Notes	14	1/13/2020	1/27/2020	AN+RG	Complete
5.7.6 Design and Develop Replacement Indicators	21	2/3/2020	2/24/2020	AN+RG	Complete
5.8 Release Bugfixes	28	2/17/2020	3/16/2020	AN+RG	Complete
5.8.1 Patches for Alpha (Regression) Testing	14	2/17/2020	3/2/2020	AN+RG	Complete
5.8.2 Patches for Beta (User Acceptance) Testing	14	3/2/2020	3/16/2020	AN+RG	Complete
6.0 Testing	77	1/13/2020	4/6/2020	ALL	Complete
6.1 Set Up Mock Clients	21	1/13/2020	2/4/2020	AN+RG	Complete
6.1.1 Set Up Desktop Clients	14	1/13/2020	1/27/2020	AN+RG	Complete
6.1.2 Set Up Mobile Clients	14	1/20/2020	2/3/2020	AN+RG	Complete
6.2 Full Regression Test	14	2/17/2020	3/2/2020	ALL	Complete
6.2.1 Regression Test for User	14	2/17/2020	3/2/2020	ALL	Complete
6.2.2 Regression Test for Administrator	14	2/17/2020	3/2/2020	ALL	Complete
6.3 Security Test	21	2/17/2020	3/9/2019	TG	Complete
6.3.1 Firewall Test	7	2/17/2020	2/24/2020	TG	Complete
6.3.2 Penetration Test	7	2/24/2020	3/2/2020	TG	Complete
6.3.3 SQL Injection Test	7	3/2/2020	3/9/2020	TG	Complete
6.4 User Acceptance Test	21	3/9/2020	3/30/2020	ALL	Complete
6.4.1 User Acceptance Test for User	14	3/9/2020	3/23/2020	ALL	Complete
6.4.2 User Acceptance Test for Administrator	14	3/16/2020	3/30/2020	ALL	Complete
6.5 Final Certification Test	7	3/30/2020	4/6/2020	TG	Complete

All = Taylor Gill, Alec Nader, Ryan Gigliotti
TG = Taylor Gill
AN = Alec Nader
RG = Ryan Gigliotti

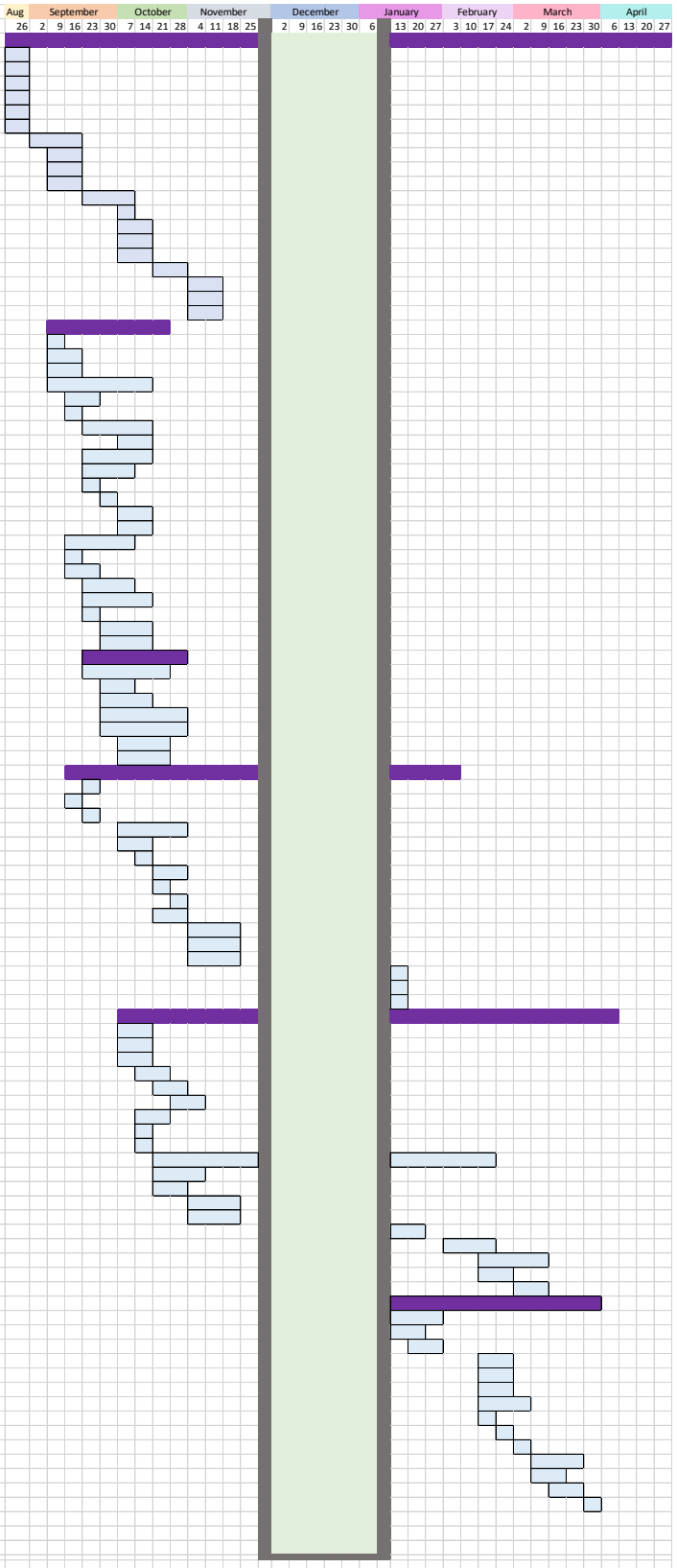


Figure 4: Project Schedule and Gantt Chart

3. TECHNICAL DISCUSSION

3.1 Network overview and discussion

Figure 4, Network Diagram represents our network structure. iMechanic is a web application is hosted on a Digital Ocean Droplet. This virtual server is running Ubuntu 18.04.3 LTS, which houses our API web server, reverse proxy, and MySQL database. We are leveraging the built in Linux Uncomplicated Firewall (UFW) as well as CertBot to ensure secure connections to our web server. Our network is simple due to our use of leveraging APIs, allowing us to only need a firewall, router, and server for full service of our application. If we want to add a DMZ for more security, it would be added to any external web application or mailing service that would be deemed necessary under future growth. Our plan is to operate solely off of a single server that hosts our database and our application. This will make it cost nothing in terms of hosting the application.

iMechanic is built on an ASP.NET Core API which runs on a Kestrel web server to complete client-to-server requests. Kestrel is not a fully featured web server but is designed to make ASP.NET requests as performant and as fast as possible. Thus, in order to fully utilize the capabilities of CertBot we have configured Nginx as a reverse proxy to handle our client requests and pass them to our Kestrel server for processing. This configuration allows us to pass all traffic as secured HTTPS requests from the client through Nginx to our Kestrel web server.

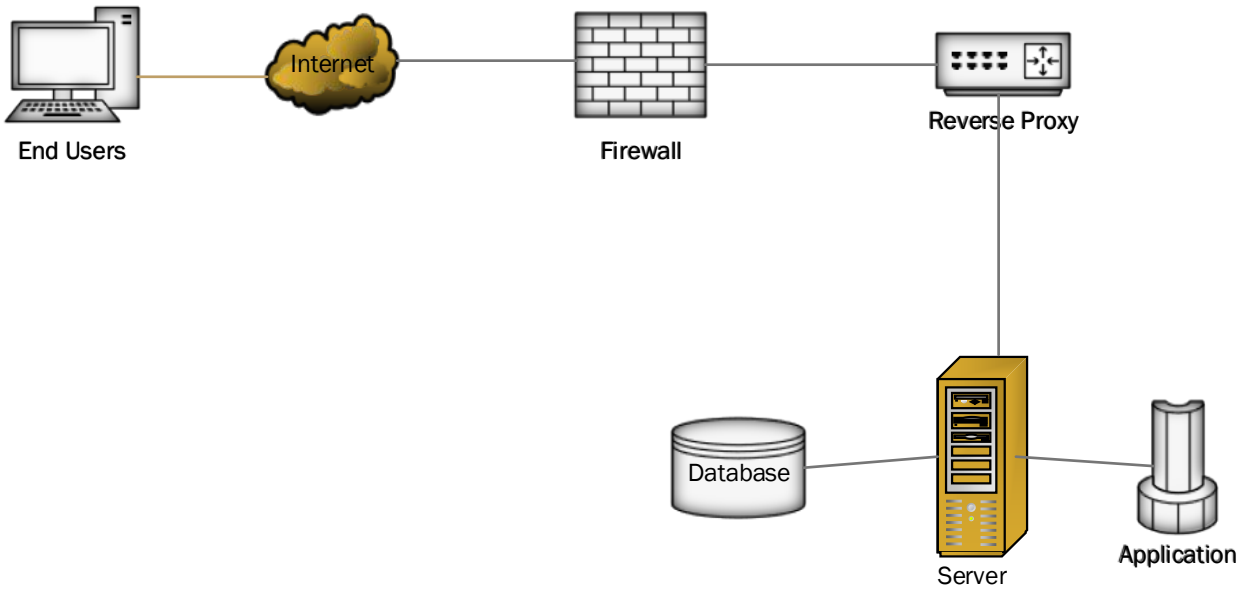


Figure 4: Network Diagram

3.2 Application overview and discussion

iMechanic is a fully responsive web application built with ASP.NET Core 2.1, MySQL, Bootstrap CSS and jQuery. iMechanic supports a variety of screen sizes and is mobile friendly. Our services layer is built out using the Entity Framework. This enables the connection between our API and our database to have Object-Relational Mapping (ORM). This simply means that when we query items from our database, they are mapped into .NET objects that can be manipulated on the server side. This also means that we can efficiently and effectively create JavaScript Object Notation (JSON) responses to send to our front-end for display. Our backend is fully fledged out to handle the tracking of multiple user owned vehicles and their current and future maintenance progress. With our backend we can track multiple vehicle components and maintenance log events in order to provide users with a detailed understanding on when their vehicle needs to go for maintenance or have parts replaced.

iMechanic's front-end layer is served by our API. Our front-end related HTML/CSS/JavaScript files are housed in our API project. Our front-end is built out using Bootstrap CSS and jQuery. Since we are using Bootstrap CSS, we can take a mobile-first approach. This means that our application will be built for mobile and will scale to larger sizes as needed. Regarding communication between client and server, our main HTTP request/response method is handled through jQuery AJAX. AJAX allows us to seamlessly integrate front and backend systems asynchronously while being able to quickly and easily handle HTTP errors. Since we are taking an asynchronous approach, our application can be used to edit database records and display the changes on the page in real time. Our front-end is also making use of some open and closed source JavaScript libraries. These include Google Maps API, jQuery DataTables, ChartJS, Full Calendar, and Font Awesome. The libraries are being used for finding mechanics in your area, data visualization, a full functioning calendar for your vehicle and high-quality icons for visual appeal.

3.3 Database overview and discussion:

Our database is built using MySQL 8 and is currently hosted on our Digital Ocean droplet running Ubuntu 18.04.3 LTS. Our database utilizes standard relational database design and includes normalization practices. As mentioned in the application overview, the services layer of our application utilizes the .NET Entity Framework. Since we have the ORM available we can dynamically build queries to our database that fill the objects defined by our API.

3.4 Security overview and discussion:

Security in our application takes basic security principles into account when users register their accounts. When a user registers with our application, they are assigned a salt value which is a randomly generated alphanumeric string 64 characters long. This salt value is hashed with the user's password using SHA256 encryption and is stored in the database.

As mentioned in the previous overviews, our backend services layer is using the .NET Entity framework which will prevent all attempts of SQL injection. Since the Entity framework heavily relies on Language-Integrated Query (LINQ), when building queries, LINQ passes all data as named parameters to the database. This means that the values are substituted server side and ensures that SQL injection can never occur.

Our login and session authentication are handled via JSON Web Tokens (JWT). When a user authenticates through our API with their credentials the HTTP response will set a cookie on the header of all requests between the current user's client and the server. This means that with every request between client and server we are able to authenticate a user's account and session to appropriately assign or reassign a session cookie to their browser.

Figure 5, SSL certification, and ***Figure 6, Open/Filtered Ports***, show the results of basic security tests completed to gauge the security level of the iMechanic application. On the main page it shows that iMechanic uses HTTPS, and our SSL certification is valid from February 2nd to March 18th. Within a Kali Linux machine, the nmap command was able to identify some open ports, with most ports being filtered through NginX. The only ports that were open were needed to allow http connections to the web server, which are then routed as HTTPS connections to the kestrel server through NginX.

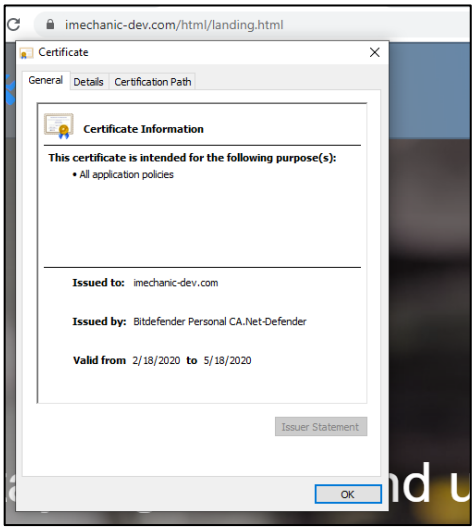


Figure 5: SSL Certification

```
Nmap done: 1 IP address (1 host up) scanned in 9.11 seconds
msf5 > db_nmap imechanic-dev.com
[*] Nmap: Starting Nmap 7.70 ( https://nmap.org ) at 2020-04-11 19:47 EDT
[*] Nmap: Nmap scan report for imechanic-dev.com (206.189.205.241)
[*] Nmap: Host is up (0.033s latency).
[*] Nmap: Not shown: 997 filtered ports
[*] Nmap: PORT    STATE SERVICE
[*] Nmap: 22/tcp  open  ssh
[*] Nmap: 80/tcp  open  http
[*] Nmap: 443/tcp open  https
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 4.65 seconds
```

Figure 6: Open/Filtered Ports

3.5 Future Recommendations

Given the opportunity to start over, I do not think we would change much of our timeline, but it is important to establish early on whether goals and features are attainable. It is important to focus on what is obtainable and test it rigorously so that the team is left with a small, but very efficient program. If we had more time, the best way to get the most benefit would be to complete more testing. From our testing we can see there's numerous factors that play into aesthetics, functionality, and usability. Our initial plan was to have a sleek, easy to use, modern application. We have a beautiful user interface but there are areas that could use final polishing with more time. We have had ideas and suggestions about certain features from others, for

example, what about a maintenance log that can be easily uploaded on the web application to bypass data entry from the user. Some of our other goals have been equally ambitious, but we took the time instead to focus on the fundamentals of our application. The future plans for iMechanic are not set in stone. The application needs more work in order to be appropriately finished, but the foundation is being used as a visual guide of our skillsets to future employees.

4. VISUALS

4.1 User interface and screenshots

4.1.1 User Dashboard

Figure 7: User Dashboard shows the main hub of iMechanic. From the dashboard the user can get a quick rundown of what needs attention, as well as be able to plan for maintenance coming up. From the dashboard the user can navigate through the rest of the application in order to find a local mechanic, see various statistics, update their logs, and view their calendar.

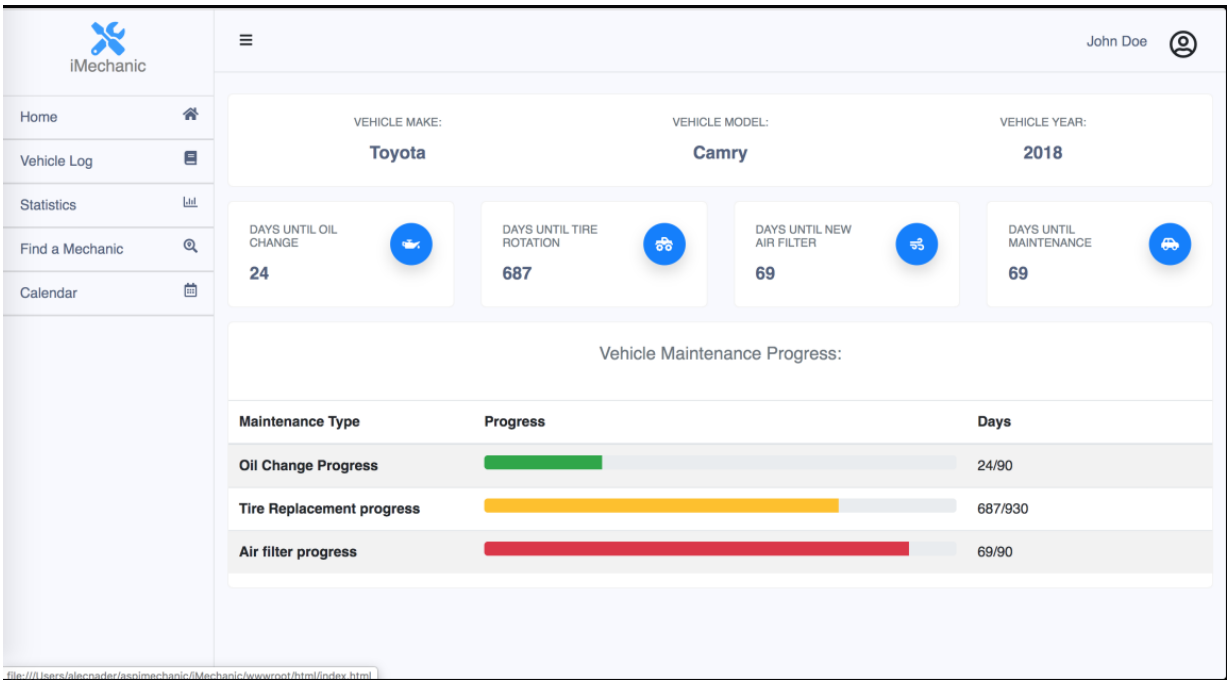


Figure 7: User dashboard

4.1.2 Calendar

Figure 8: User Calendar will allow users to schedule their maintenance easily with their reminders also showing on their calendars, helping them see maintenance in their week/month as it comes up.

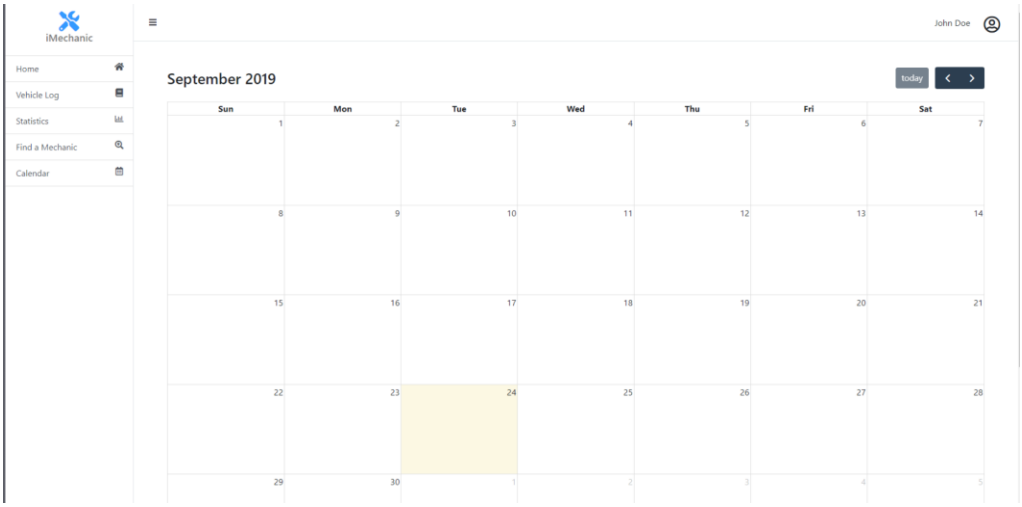
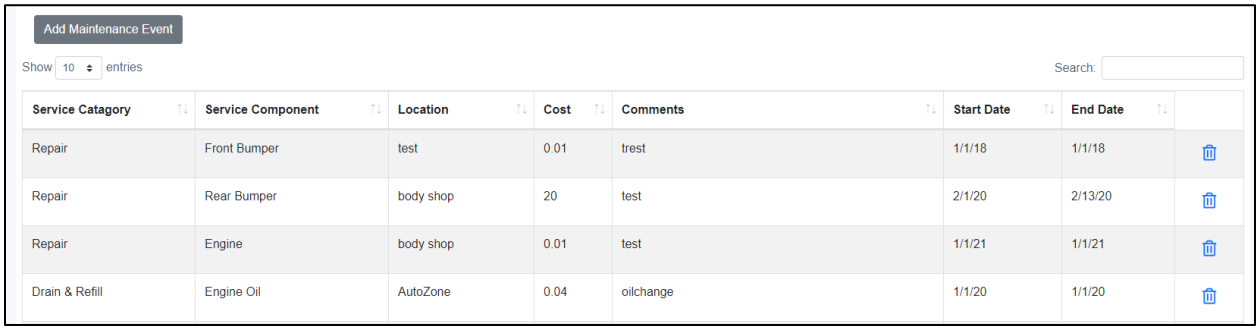


Figure 8: User Calendar

4.1.3 Maintenance Logs

Figure 9: User Maintenance Logs and ***Figure 11: User Statistics*** shows the in-depth detail of all the various components that come along with many vehicles. Maintenance logs use mileage and service history to inform users when to inspect various parts over time.



Service Catagory	Service Component	Location	Cost	Comments	Start Date	End Date
Repair	Front Bumper	test	0.01	trest	1/1/18	1/1/18
Repair	Rear Bumper	body shop	20	test	2/1/20	2/13/20
Repair	Engine	body shop	0.01	test	1/1/21	1/1/21
Drain & Refill	Engine Oil	AutoZone	0.04	oilchange	1/1/20	1/1/20

Figure 9: User Maintenance Logs

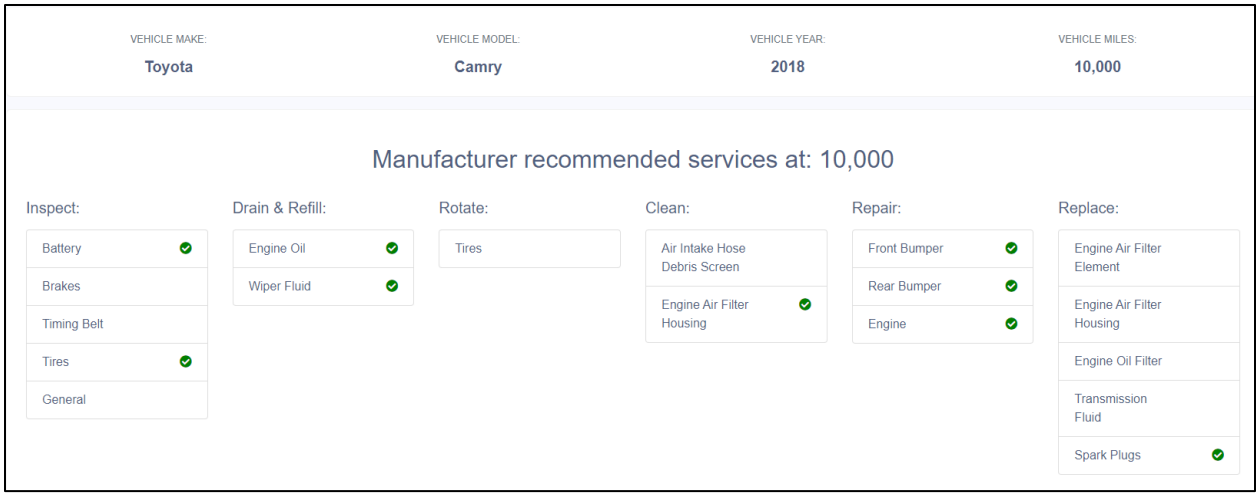


Figure 10: User Maintenance Logs Continued

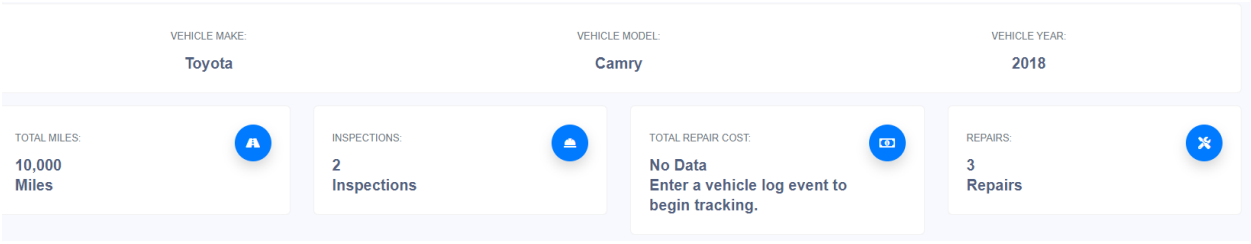


Figure 11: User Statistics

5. TEST PLAN

5.1 Overview and Methodology

The team will use JIRA to create test cases for iMechanic for a clean and easy way to log our testing and changes. Using JIRA issue tracker, the entire team can create defect issues with associated test cases. These completed test cases will become our logs for properly documented testing. The main steps that we anticipate user's taking within our application will be explored by all team members to ensure there is no bias-based blindness within our features.

5.2 Objective

We are ensuring that everyone on the team thoroughly and repeatedly tests all parts of the application to ensure the web application shows continuous integrity. Front end's functionality will be verified through smoke testing, exploratory findings, as well as working through our test cases located on our Jira dashboard. Our main functionalities will be tested with various inputs to ensure proper throwback and data retention. We are going to ensure the application functions with a continuously sleek and easy-to-use design. Vulnerability scanning and penetration testing of the web application is anticipated later in the semester.

5.3 Test Cases

1. Testing Proper Account Functionality/Test Cases
    a. User's first and last names have no special characters
    b. Passwords should be at least 6 characters.
    c. Email in use cannot be reused
    d. User should receive email and register account to log in.

e. Unauthorized users cannot login

f. User should be greeted with a prompt to add vehicle information

2. Testing Proper Visuals and Functionality/Interface

a. User can enter car information and it will be displayed on home page

b. The iMechanic logo and "select a vehicle" should show on all pages

c. The left hand of all pages should show the list of other pages.

d. Clicking on the calendar prompts for entry

e. Entry entered on calendar should be reflected on the main calendar page

f. Logging in should prompt user to update their vehicle mileage.

5.4 Testing Report

Req no:	Item #	Test case #	Input	Expected output	Actual output	Pass/ Fail	Reason for failure/success	Date
1	1.a	1	A!	Error Message	Success	F	Special Character	3/5/20
		2	F+	Error message	Success	F	Special Character	3/5/20
		3	JohnDoe	Success	Success	P	Proper name	3/5/20
1	1.b 1.c	4	0	Error Message	Success	F	Password is 1 character	3/5/20
		5	(	Error Message	Success	F	Password is 1 character	3/5/20
		6	a@test.com	Error message	Error message	P	Email cannot be reused	3/5/20
1	1.d	7	Login w/ registered	Login Success	Login Success	P	Registered login	3/13/20
		8	Login w/ unregistered	Error Message	Login Success	F	Unregistered login	3/13/20
1	1.e	9	Incorrect pw	Error Message	Error Message	P	Incorrect PW rejected	3/5/20
		10	Incorrect pw	Error Message	Error Message	P		3/5/20
		11	Incorrect pw	Error message	Error message	P		3/5/20
1	1.f	12	Login	Entry Prompt	No prompt	F	No prompt	1/14/20
		13	Login	Entry Prompt	Entry Prompt	P	Prompt on login	3/5/20
		14	Login	Entry Prompt	Entry Prompt	P	Prompt on login	3/5/20

2	2.a	15	Enter Car information	Volkswagen Jetta on Home page	Page reflects proper car info	P	Proper make and model reflected	3/13/20
2	2.b	16 17 18	Stats Page Log Page Calendar	iMechanic Logo shows	iMechanic Logo shows	P P P	Proper visuals	3/13/20 3/13/20 3/13/20
2	2.c	19 20 21	Stats Page Log Page Calendar	Access to all other pages listed on left side of page	Access to all other pages listed on left side of page	P P P	Easy access to other pages, visually appealing	3/13/20 3/13/20 3/13/20
2	2.d	22	Select a day on the calendar	Prompt shows up to enter info	Prompt shows up as expected	P	Functionality of calendar	3/5/20
2	2.e	23	Tire rotation March 30th	Calendar shows plan of maintenance	Properly reflected on Calendar	P	Functionality of calendar	3/5/20
2	2.f	24	Login	Prompt shows up to update mileage/work	Prompt on top right corner prompts for update	P	Functionality of prompts on login	3/5/20

6. CONCLUSION

6.1 Fall Semester 2019

The main progress of our product so far is creating a basis of everything we have envisioned that iMechanic will be capable of. We have determined the features that we want to include and have started bringing various features to life. We are working on integrating our backend and frontend, as well as setting up our SQL server. We will continue to work on our development as the project continues. More focus will turn to security and testing during the Spring semester.

6.2 Spring Semester 2020

We have moved passed the design phase and already set in place the features that we would like our application to have. During the Spring, the main focus has been completing the back-end and polishing up the front-end interface. The functionality of the program has been completed, allowing us to test our visuals and see if the program works the way we intended. The application is able to be tested on its security, functionality, and accessibility, which is the main goal until the end of the semester.

APPENDIX A. ADDITIONAL INFORMATION

APPENDIX B. REFERENCES

[1] Consumer Reports. (2016, February 9). Can You Trust Your Mechanic? Retrieved November 10, 2019, from https://www.consumerreports.org/car-maintenance/can-you-trust-your-mechanic/.