UX for Home Mechanics

Helping mechanics stay on track

Gustav Eriksson
Abstract

Home mechanics services their own vehicles instead of handing them to a professional mechanic. Professional mechanics have a set of recurring service items put forth by the manufacture of the vehicle. The home mechanics may follow the same schedule or may have their own way of doing things. This project created a UX for those home mechanics to keep track of what service work they have completed and when for their different vehicles.

Within the scopes of the project the type of home mechanics was ones working on their own motorcycles. The UX has nothing stopping a mechanic from using it for any other kind of vehicle, but the focus group used to test it had motorcycles as their point of reference.

The UX was created from scratch, going from hastily drawn diagrams to mockup images, and finally an interactive digital prototype built with Node.js and VueJS.

The final iteration of the prototype allows the user to set a recurring schedule and registered that an instance of a service has been carried out, without getting stuck or committing major errors by the user testing group.
# Contents

1 Introduction .................................. 1
   1.1 Unrelated Items at the Same Time .............. 1
   1.2 Limiting the Target Demographic ................. 1
   1.3 User Tests as Test Driven Design ................ 2
   1.4 Three Iterations of the Prototype ................. 2

2 Background .................................. 3
   2.1 Home Mechanics ................................. 3
   2.2 Professional Mechanics have Different Workflows .... 4
   2.3 Introducing the Project Owner .................. 4
   2.4 Why Maintenance Records are Important .......... 4

3 Purpose, Aims, and Motivation .................. 5
   3.1 Sustainability Goals .............................. 6
   3.2 Ethical Questions ................................. 6
   3.3 Delimitations .................................... 7
      3.3.1 Only Targeting Computers .................... 7
      3.3.2 Limitations of Technologies Chosen .......... 7
      3.3.3 Avoiding Dates and Times .................... 8
      3.3.4 Lacking Instructions and Tool Tips .......... 8

4 Related Works ................................ 8
   4.1 Computerized Maintenance Management Systems (CMMS) .... 9
   4.2 AUTOsist ........................................ 9
   4.3 Simply Auto .................................... 9
8 Building the Prototypes

8.1 Non-interactive Prototypes

8.1.1 Low Quality Sketches During Discussions with Project Owner

8.1.2 Designing the Layout with Images

8.1.3 From Images to HTML and CSS

8.2 Interactive Prototypes

8.2.1 Implementing the Design in Node.js

8.2.2 From HTML to Vue Components

8.2.3 Placeholders for Computed Values

8.2.4 Storing Data

8.2.5 Onto User Testing

8.2.6 Functionality beyond Minimum Requirements

9 User Testing the Prototypes

9.1 How to Users were Chosen

9.1.1 Users Excluded from the Results

9.2 The Physical Testing Environment

9.3 Establishing the Proficiency of the User

9.3.1 Computer Proficiency

9.3.2 Motorcycle Servicing Proficiency

9.4 Tasks for Users to Perform

9.4.1 Preexisting Data

9.4.2 Task List

9.5 Free to Speak Their Mind

10 Generations of Prototypes
A.2 Results from the Second Version ........................................ 45
A.3 Results from the Third Version ........................................ 45

B Users Answers to Questionnaire ........................................ 45
  B.1 Quantifiable ............................................................. 45
  B.2 Free Form - How You Currently Keep Track of Maintenance . . . . 46

C Images of Pages During All Versions ................................. 46
  C.1 Image of Pages During Version Two ................................ 46
  C.2 Image of Pages During Version Three ............................ 46
1 Introduction

Buyers of vehicles put a high level of trust in the service centers and professional mechanics to have carried out services of the vehicle correctly. Because of this, being able to prove that the work has been carried out by a trustworthy company positively affects the resale values of vehicles.[Can22] Home mechanics lack this method of proving work carried out and may compensate with receipts from buying the materials needed to complete the instance of the service item. After 20 years of ownership some receipts may be faded or lost, keeping a digital record enables easy backups which can withstand the test of time.

Home mechanics differ from professional mechanics and manufacture authorized service centers in the variety of vehicles they work on. A home mechanic is likely working on their own vehicles and possibly a limited number of friends motorcycles. As of June 30th, 2018, there was 59 669 more motorcycles than owners of motorcycles in Sweden meaning that some own more than one motorcycle.[Mot18] As such the UX must support multiple motorcycles. Unlike an authorized service center which is likely to see multiple bikes of the same model, owners are more likely to service a variety of motorcycle models. This means the workflow may differ between the two types of mechanics and as such may have different desired layouts of UI to best fit their needs. This project sole focus is on the home mechanics needs.

1.1 Unrelated Items at the Same Time

There is a lot of different recurring service items that needs to be carried out to properly maintain a vehicle. According to Kawasaki’s service manual for the ER-6f 2009, there are fifty different service items with varying frequency.[Kaw09] Some have intervals that depend on which comes first, a specific period of time or a distance traveled. There’s a variety of ways home mechanics choose to keep track of these service items. This includes receipts, date, and mileage. Some use physical papers and the receipts themselves whereas others use digital spreadsheets. There are also apps for mobile devices and online services.

1.2 Limiting the Target Demographic

This project will not be a fully featured solution for keeping track of the maintenance. The project is solely focused on creating a UX which is simple to use with a low learning curve. It does not feature any helpful tips for what services should be done or how
to do them. The home mechanics which is the target audience are those with experience of servicing their bikes and not complete novices. The distinction between these two groups for this project is their experience reading a service manual and completing service items. This program does not contain or support instructions for how to complete a service item. It does support information which is useful prior to carrying out the service work, such as lists of tools needed, torque values, and warning messages. This information is there to be a quick reference instead of having to check the service manual.

1.3 User Tests as Test Driven Design

To create this UX, first user tests were written to be used as test driven design. The UX was designed to be used with computer monitors. The initial tests started off as single focused tasks, such as “add a new motorcycle”. When the first tasks were doable in the UI, additional tasks were added with multiple steps, such as “register an instance of service item X have been carried out on motorcycle Y”. These user tests were in part created based on information provided by the project owner and in part based on a few service manuals’ periodic maintenance schedule.

When the user tests deemed to be critical were able to be completed in theory, the user tests were used for practical user testing with the target demographic. Metrics regarding their objective performance in completing the tasks were recorded and used to evaluate the UX. Problems were addressed for an improved implementation in later iterations of the prototype.

1.4 Three Iterations of the Prototype

During the project three iterations of the prototype was tested by users with the same set of tasks to complete. They were given the same tasks for all iterations to test if changes between the iterations were improvements or hindrance. The first version’s biggest issue was overload of information when trying to enter data regarding a new vehicle. The overload of information led users to saving the data before completing the task. The first version did not offer a way to correct this mistake without entering the information again as a new vehicle. The second version solved both issues, but the way users were able to modify the data after initially saving the vehicle created an issue. The issue was in a prior task, which no users had any issues completing in the first version. The third iteration addressed this, and all tasks were completed by all users without errors which could not be solved by the user.
2 Background

When a mechanic at a manufacture authorized service center that is not the sole mechanic, most commonly the person that interacted with the customer wasn’t the mechanic, but a service representative.[AUT22b] The customer might only provide the information that the X-mileage service needs to be completed. The service representative enters the information in their system and when the mechanic has time, they will start carrying out the service. The scope of the work will differ depending on mileage and age of the vehicle. The mechanic will know what to do based on the information in either a physical copy of the service manual or some digital alternative provided by the manufacture.

A non-authorized company mechanic might have a physical copy of the service manual but is more likely to complete work based on prior experience and expertise rather than a model specific manual. While more likely the mechanic may be the sole proprietor of the company, the information provided by the customer is likely to be limited to the mileage or age of the motorcycle. The same as when using an authorized service center.

2.1 Home Mechanics

As a home mechanic, the customer and mechanic are the same person and as such the information of what should be done will be based on the mechanics own preference. If the motorcycle is used for a more exhausting purpose, such as racing, compared to what the motorcycle and therefor the service schedule was designed for, the mechanic may choose to increase the frequency for certain services.[Lub20]

When an owner has a company maintain the motorcycle rather than doing it themselves it is likely that multiple service items will be completed at the same time, such as lubricating the steering stem bearing and changing brake fluid. They are unrelated service items, but they have the same service interval.[Kaw09] Changing the brake fluid takes very little effort and time, and the home mechanic may opt to do it during the evening of a workday in the middle of the summer. In contrast, lubricating the steering stem bearing requires a fair amount of setup. As such, a home mechanic might opt to do it while taking apart the motorcycle to do multiple service items at the same time during winter.
2 Background

2.2 Professional Mechanics have Different Workflows

Because of how the professional mechanic and home mechanic are likely to do work, they have different needs for keeping track of when instances of service items were completed. In addition, the variety of motorcycles they work on is likely different. A mechanic at an authorized service center is likely to see a large quantity of the same model of motorcycles almost daily as certain motorcycle models sells in significantly larger quantities than other. It would be a lot of repetitive work if each time the mechanic would have to set up a new motorcycle in a program from scratch every time a new customer came in. Finding the exact motorcycle in a list of motorcycles from different customers or multiple for the same customer could be done using the Vehicle Identification Number (VIN).[Sha21]

For a home mechanic it’s different as they are likely to set up the motorcycle in the service tracking system once, whether it be analog or digital. The motorcycles to choose from are likely to be different models meaning that knowing the model itself would likely be all the information that is needed to find it in a list of motorcycles instead of using the vehicle’s VIN.

A home mechanic is not necessarily going to use whatever maintenance tracking system they use every day, as they are not likely going to service their motorcycle every day. As such they need to be able to recognize features and how to use them without having to relearn shortcuts. Because of this, the UX needs to be easy to use and more explicit (not using acronyms) compared to a UX designed for someone to use every day at work.

2.3 Introducing the Project Owner

The project owner is a home mechanic that has been servicing their own motorcycle for decades and is well versed in the subject matter. Beyond servicing their own motorcycle, they also help teach new home mechanics basic maintenance tasks as well as giving advice to other experienced home mechanics. Having been a home mechanic for decades they’ve seen the switch from fully analog to fully electronic systems that is common on modern motorcycles.

2.4 Why Maintenance Records are Important

There are two key reasons to keep track of what services have been done and when they were made. The first reason is to keep track of when the next service needs to be
done based on the service interval and the last time it was done. The second reason is to show evidence that the vehicle has been taken care of and not neglected. When you buy a new vehicle, you’ll receive a booklet where the company that services the vehicle can provide their stamp to show that a certain service has been made. Many people will ask for this booklet when looking to buy the vehicle from you as a peace of mind that the vehicle will most likely not suffer premature failure. Being able to provide evidence to support the claim that the vehicle has been maintained in accordance with the maintenance schedule increases the value of the vehicle.[Can22]

Modern vehicles are getting more digital and with this a shift from physical paper booklets to track the service records to digital systems. In a BMW car this can be accessed in the vehicles infotainment system. BMW service centers can pull up the record stored by BMW outside the car.[Ana22]

3 Purpose, Aims, and Motivation

The end goal of this project was to create an interactive prototype of an interface that home mechanics would find easy to use without instructions. The primary goal should be for the user to be able to track when the last instance of a service item was carried out and register that a new instance of the service item has been completed. To be more useful than writing it down on a piece of paper or in a calendar, there are some additional functionalities aiming to help the home mechanic. These functionalities include providing a quick reference for tools, torque values and similar which is relevant to the particular service item.

The purpose of this interface is to provide useful information to the home mechanic that is the current owner of the vehicle. This regard keeping track of when recurring service items are due by providing an easy-to-understand history of previous instances of service items. In addition to knowing when services items are due, an abnormal frequent need for services can indicate mechanical issues that requires addressing. One such concern could stem from frequently needing to top off the engine oil as this could indicate that the motor is worn internally or is leaking oil.[Car21]

According to Carfax Canada (a subsidiary of Carfax, a global company that provides vehicle history reports) one factor in the resale value of a vehicle is the existence of service history.[Can22] Carfax Canada includes the service history when ordering a vehicle history report from them on a vehicle, but this service history only pulls data from service centers partnered with Carfax Canada.[Can20]

BMW has a digital service history in the cars entertainment system which shows when
services are due and when they were last done. This system is locked such that only au-
thorized service centers can enter information into the system. As with Carfax Canada,
the home mechanic is not able to utilize these systems to record services they have
carried out themselves.[Ana22]

Before digital systems there was, and still exists, service history in the form of paper.
Some owners keep a service history in the form the receipts from the work done by a
service center, or by the service center signing the service booklet that is included when
purchasing a new vehicle.[Mas19] As with any paper trail it risks disappearing from be-
ing lost or fires. They are also not searchable unless properly organized and the number
of records from 30 years can make it time-consuming to find any specific information.
In the case of selling a vehicle the buyer is unable to inspect the records without being
there in person or having the records physically transported to them without creating a
digital replica, such as scans.

The user TheStrifist posted a forum post on the subreddit "r/motorcycles", that has 1,3
million users in early 2020, if other users could recommend an Android app to keep
track on the maintenance of their motorcycle. They received three different suggestions
of maintenance tracker apps. One of them were intended for cars, one was intended
for fleet owners, and one was intended for any type of vehicle. Beyond these actual
suggestions for apps there was also the suggestion to use a calendar app and make an
entry there. Another user suggested using a spreadsheet program.[CB320]

More detailed exploration of existing competing solutions is discussed under the header
Related works.

### 3.1 Sustainability Goals

Target 12.5 of the UN’s global goals is to “By 2030, substantially reduce waste genera-
tion through prevention, reduction, recycling and reuse.”.[Goa22] Maintaining vehicles
prolongs their lifespan which reduces the emissions compared to manufacturing a new
vehicle to replace the old one if there are not substantial gains in efficiency (such as
going from gasoline to electric) and the distance traveled is sufficiently long.[Wis22]

### 3.2 Ethical Questions

The prototypes are built to not require communication with the server except when
fetching the webpage itself. There is no information regarding the user saved server
side when a fetch request is received. Nor is there any information sent from the user to
the server to store the data the user enters in any text field.

During this project there has been no particular focus on being inclusive for disabilities. As the target users are mechanics, they are presumed to have vision and do not require read-out-loud features. There is no audio in the prototype, as such auditory senses are not required to utilize the prototypes.

3.3 Delimitations

During the initial meeting with the project owner a dream version of what a final product could contain was discussed. This dream version had no limitations and was a free-flowing forum for ideas. Delivering the dream version within the scope of this project was unfeasible and as such some ideas was shelved.

3.3.1 Only Targeting Computers

Some home mechanics may prefer to use a smartphone whereas some prefer to use a laptop or desktop computer. This interface is intended to be used on the latter, but the underlying technologies used to create the interface is commonly used to also create interfaces for phones and tablets. But it is not within the scope of this project to design, develop, and test for phones and tablets.

3.3.2 Limitations of Technologies Chosen

The interactive prototypes developed to test the UX have limitations which could negatively affect functionality if they were used for keeping track of the full maintenance of a motorcycle over multiple years. Some of the reasons for this is covered under the chapter System structure, where technical limitations affect long term performance of the prototype. Beyond technical limitations there are limitations or functionality which is present in appearance but not in functionality to decrease development time. This includes systems which may be technically complex which does not affect the limited scope of the user testing described in 5. These features include things such as sorting items in descending or ascending order, computing the distance or time until the next service is due based on current mileage or date.
3.3.3 Avoiding Dates and Times

One technical challenge, which balloons development time, is how different people choose to enter data. This includes building functionality to recognize if the user wants to enter distance in imperial or metric miles, or kilometers. Service manuals generally lists periods in both imperial miles and kilometers, but the user might copy the value directly from the odometer of the motorcycle which might include one tenth resolution.[Mot22] This means that the user might enter the values for the periods as kilometers and copy the value from the odometer as hundreds of meters, creating a disconnect and values which are not directly comparable.

Dates are written in different ways globally which complicates entering and computing difference between two dates even before taking leap years and similar into account.[Map15] A study from the University of Basel in 2011 identified a drop-down menu to reduce the number of errors whereas a single input field (which is the type used in these prototypes) is faster and provide higher user satisfaction.[BABT+11]

3.3.4 Lacking Instructions and Tool Tips

Instructions, helping texts or tool tips are also not included in any of the prototypes as the test user always had the author to ask questions if stuck. Part of the instructions not included is install instructions. The prototype is not packaged in a single package but rather contained in a git repository and requires Node.JS to be installed and basic knowledge how to install npm packages is required to set up the prototype.

4 Related Works

During user testing the users were asked how they tracked the maintenance schedule currently. The answers make the cut-off for what is a related work fuzzy. Some users reported that they memorized what work they had done which year and from that extrapolated what work was due this year. Some users had varying levels of structure to where they recorded service work in paper or digital form. Those who owned a motorcycle which was still covered by warranty used the booklet included when they bought the motorcycle. In the booklet the service center puts their stamp when a service has been completed.
4.1 Computerized Maintenance Management Systems (CMMS)

At larger companies Computerized Maintenance Management Systems (CMMS) is used to keep track of both vehicle and facilities regarding their maintenance.[IBM22] There is a large selection of software to fill this role within companies, but they offer functionality which is not necessary for home mechanics. Taking MaintainX as one example, a lot of functionality is designed for large corporations such as functionality to have a pool of mechanics which can address many items of equipment and is not focused on motorcycles or vehicles in general.[Mai22]

4.2 AUTOsist

There are applications and services dedicated to vehicles specifically such as AUTOsist but this is to track fleets of vehicles, with the free version being limited to a single vehicle. AUTOsist is in large extent a CMMS but tailored to vehicles with features which is not needed by the home mechanic such as being able to assign vehicles to specific users in a pool of multiple users.[AUT22c]

4.3 Simply Auto

Some applications are more focused on vehicle owners such as Simply Auto but it focuses more on keeping track of costs and fuel. It’s also limited to Apple and Android devices.[Aut22a]

4.4 The Missing Thing from Alternatives

The focus which these alternatives miss is an owner doing their own work and wanting to quickly be reminded of what tools and torque values the service work requires. To get the information of tools and torques at the same time as knowing when to do them, the service manual must be used in addition to these digital systems.

The service manual informs the reader in detail how to complete a specific item and it also contains a table of items and their torque values, if thread glue must be used, and if O-rings or similar should be replaced when doing the work. But they don’t write the tools needed in the same list, nor does it include the service intervals which leads to a lot of flipping back and forth. They are intended to be used in service centers on multiple vehicles which means they do not contain a location to enter the service history.[Kaw09]
5 Method and Approach

The workflow of the project is based on the book “Interaction design and UX” (project’s author’s translation) by M. Arvola. The book breaks down the development of UX in three phases: conceptual design, work phase, and detail phase, listed in order of operation.[Arv14, p. 7] Through the phases different aspects are covered such as how to conduct user testing, both important concerns regarding if one is to record video or audio during the interview.[Arv14, p. 46] But it also covers how to design questions to ask during the user testing.[Arv14, p. 41-45] Some aspects of the book are more suitable for projects larger than this one but most of the aspects from the book is applicable. What the book does not cover is a particular design language, rather this is the aspect where the reader is forced to be creative on their own. This report will not be written as a step-by-step result from following the book, but when aspects from the book has been implemented it will be discussed.

This book was chosen as it was the course book the project’s author was assigned for the course ”System Design with a User Perspective” at Uppsala University.[oIT19]

5.1 User Testing

To verify that the UI was simple to use, it was tested by a spread of users of varying level of computer, and mechanical proficiency. Arvola describes methods for how to conduct user testing and user interviews in the different stages of development.[Arv14] Because of the small scope of this project there was no initial interviews to establish needs of the userbase, instead this relied on the expertise of the project owner.

During the detail phase the initial user testing was performed. The first prototype which met the minimum requirements was tested by a couple of users after which issues found by the testing was addressed. After the issues being addressed a second round of user testing was conducted with new users. This was done a third time, which was the final version.

The user testing was carried out in large extents in accordance with the format presented by Arvola. This included asking some questions to establish their grouping, if they were to be classified as an experience home mechanic or novice, and if they were comfortable using the hardware platform on which the user testing was conducted. Also informing the users that they would be anonymous, and the fact they were not the ones being tested but rather the UI was being tested.[Arv14, p. 134-138]
5.2 Choosing Technology

The technology chosen to build the interactive prototype was chosen based on two aspects. The first aspect comes from the dream version of what this project could build to in the future beyond this project. The vision was an interface which was platform agnostic (Windows, Mac, and Linux) and could also be adopted to both phones and tablets. There are many options and to narrow it down the second aspect came in to play which was the prior experience of the project’s author. This left three options; Avalonia, Electron, or a webpage delivered locally.

5.2.1 Avalonia

Avalonia is an open-source project with XAML syntax for the graphics side which is based on Windows Presentation Format (WPF) and Universal Windows Platform (UWP) which are two GUI systems developed by Microsoft. The backend is written in C# and uses .NET. Avalonia is compiled into a program which runs locally. The program can be compiled for the three big desktop operating systems, both Android and iOS, and web assembly.

5.2.2 Deliver a Webpage Locally

Regarding delivering a webpage locally, there are multiple options for both frontend and backend. One option is to host a local server using one of many available platforms such as XAMPP, .NET, or Node.js. With these as backends the end user would run their own server on the device they would use for the program and then they would access the web page on a local address. These three options offer different backend languages, XAMPP offers backend languages in PHP and Pearl whereas .NET offers backend in any of the .NET languages such as C# or F#. Node.js is a large platform for hosting JavaScript servers.

5.2.3 Electron

Electron is a competing product with Avalonia, built on Node.js. It bridges the gap between a local application, a server application, and a webpage. Like Node.js, Electron is an application written in JavaScript and is essentially a Node.js server bundled with an instance of the Chromium web browser. The Electron application is compiled and distributed in the same way as an Avalonia application.
The project’s author had experience with writing both WPF applications and Node.js applications for locally hosting websites. The disadvantages of Avalonia as compared to Node.js for this project is the significantly smaller user base and the fact that Avalonia is still in beta whereas Node.js is a production ready solution. Because of these disadvantages, Node.js was deemed to be the better option for this project.

5.2.4 Node.js

The Node.js server serves the user with a web page when the user requests a page associated with the server. This webpage could be anything and could be written in any language but common languages for web pages are a combination of HTML, CSS, and JavaScript. This response to the user’s requests is known as the frontend and to build the frontend there are many solutions available. These includes using the basics of HTML CSS and JavaScript but there’s also a wide variety of frontend frameworks which are popular to use to build your web page upon. There is a multitude of available frontend frameworks to choose from including React by Meta (previously known as Facebook), AngularJS by Google, and Vue.js by Evan You. In this case Vue.js was chosen since the project’s author had prior experience with this framework. Vue.js will from this point in the report be referred to as Vue.

5.2.5 Vue

Vue can be implemented in a variety of ways including being imported as a script in the header tags of the HTML file, but this does limit certain functionality of Vue. Vue can also be used to build single page applications (SPA) in which Vue control large part of what would normally be considered backend work, such as handling routing on user’s request. Controlling the routing means that when you go to a subpage of a website such as webpage.com/profile where the subpage is /profile. Routing would control what is presented when the user goes from webpage.com to webpage.com/profile. In this project Vue was utilized in the latter possibility, implemented as an SPA.

One of the features which makes Vue preferable to use compared to not using a framework at all is the control of the visual elements of the web page. Variables in the instance of Vue are reactive which means that if a list of visual elements is to be rendered based on a list of data elements contained in the Vue instance, when an item is added or removed from the list of data elements the list of visual elements will be modified to reflect the change automatically.
6 System Structure

This program is self-contained and does not rely on external sources of information nor does it rely on being able to communicate with external systems. While the interface is presented to the user in their web browser as a web page, they can access the interface as a traditional desktop application rather than a web service as no internet connection is needed.

There is no form of administrations or superusers in the program, nor is there any kind of logging in for information. If two separate users are using the same computer, it is going to depend on their web browser if they see different or the same data as each other.

6.1 User

The user is free to use any physical device with a web browser which follows modern standards for CSS, JS, and Web API. Support for all three of these technologies are required to render and use all functionality of the frontend. The prototype is only tested on one desktop and one 15” laptop, but there is nothing stopping the user from opening the program on a phone. The backend (Node.js) does not support executing on Android or iOS, but the program could be accessed via the browser to a backend like any website.

6.2 Frontend

The entire frontend is controlled by Vue. Vue router handles navigation requests from the user regardless of if the user clicks a button or link inside the UI to navigate or if they use the web browsers features for navigating, such as going backwards.

The website is built as Vue components which are compiled at the backend to HTML, JS, and CSS. This compiled code is sent to the user when they request the website from the backend. The web browser of the user renders the package, and the user can interact with it as with any website.

The Vue components were built in TypeScript rather than JavaScript to enable type warnings prior to runtime. During compilation of the Vue components the TypeScript code is compiled into JavaScript. Using TypeScript was an option selected during the initial setup of Vue during development.
Figure 1 The system structure of the interactive prototype. Vue.js logo courtesy of logo.wine, Node.js logo courtesy of logos-download.com.
6.2.1 Content

Content is rendered by Vue based on current data and the user’s interactions. The content is what is displayed to the user in their web browser. When a page is requested, it is handled locally in the frontend by the router which switches the content.

6.2.2 Router

Routing is handled by the Vue Router which is an option in Vue applications.[Vue22] This enables the page to be a SPA instead of a new request going to the server each time the user loads a new page. The entire website is downloaded on the first request which increases the initial load time of the landing page but subsequent loads of any of the pages is faster since the data is cached locally.[Ach21]

With exception of load times between page switches and not requiring an internet connection (if the prototype was hosted online), the user sees no difference between this solution instead of traditional separate pages. Note that in this prototype the backend is hosted locally so no internet connection is needed anyway.

6.2.3 Web Browser and localStorage

localStorage is part of the Web Storage API supported by all major web browsers. The Vue instance which handles the content access localStorage through this API to store all data saved by the user. No cookies or other methods of storage is used. localStorage is on a per website basis, allowing access regardless of current page. localStorage stores strings in key value pairs.[con22] When saving the data which are JavaScript objects in the Vue instance it’s converted to string using JSON.stringify using custom logic to manage the keys.

There is no formal specification for how much data can be stored in localStorage, leading people to develop scripts and browser extensions to find the number. The user gkucmierz wrote in 2020 that, according to his testing, several the biggest web browsers on the market had approximately 5 MB of storage but Safari had approximately 2 MB.[gku20]
6.3 Backend

The backend is managed by Node.js. It was setup using the command `npm init vue@latest`. No other npm packages is used in the project. The backend responds to http requests and similar network communication with users. The Node.js instance is executed on the server.

6.4 Server

The code is platform agnostic and can run on any host system which is able to run a Node.js project. This includes the three big desktop operating systems and additionally some server specific solutions and container solutions such as Docker.

7 Requirements and Evaluation Methods

The goal of the project was to deliver a prototype interface which was interactable by the user. In this user interface the user had to be able to complete certain tasks. The quality of the interface is determined by the number of errors the user made trying to complete each task. An error was defined as the following things:

- A miss click caused by the interface (not caused by misuse of the trackpad on the testing computer)
- Incorrect navigation, going to a location where the task cannot be completed or going to the correct place but before saving changes.
- Misunderstanding and communicating the wrong information. E.g., asked when the last time a service item was completed and reporting the time until the next time the service should be completed.

7.1 What Was Not Considered an Error

Some issues or errors were not quantifiable by the user test results for some users, as certain users would spend a significant amount of time trying to figure out how not to commit an error rather than quickly clicking on what they thought to be the correct answer. Some users would voice their confusion or concern whereas others would keep
quiet, maybe interact with the computer maybe not for an extended period of time, and eventually ask or committing to an action. In the results this was not listed as an error but rather its own category named *extended pauses*. The definition of an extended pause was not set to a specific amount of time but rather if the pause was extensive compared to their other pauses or delays in actions for things they believed or felt were easy to complete. An example of this could be if they had taken very little time to identify the manufacturer of the first motorcycle in the list of motorcycles, but then took a significant amount of time to start entering information to record a service item for a new motorcycle as part of a user task. If the user was writing slowly or writing for an extended period of time, this was not considered a pause as the user was still committing actions.

### 7.2 Setting the Bar

The quantity of errors deemed to be an acceptable number of errors or number of extended pauses for a certain task was left undefined. If completing a certain task or doing a certain action during the completion of a certain task stood out from the rest, then this feature of the program was deemed below acceptable quality and was iterated on for the next prototype.

Each prototype was tested by a limited number of users to increase the turnover of the number of prototypes, to enable testing of more ideas with solutions if one task had a high number of errors.

### 7.3 Speed Was Not a Metric

While speed may be interesting, the portion of completing a service item which relates to recording this information is disappearingly small compared to the amount of time needed to physically complete the service item. In a service center where time is money the time would be a greater concern but for home mechanics this has a very limited impact.

### 7.4 How Errors were Recorded

To record the number of errors the user had two options. Either being video recorded from behind or the project’s author would record the number of errors using a tally counter. If the user accepted being video recorded then later, away from the user, the
project’s author watched the video and wrote down the number of errors and extended pauses and which task was set as the goal when the error was made.

In practice all users agreed to be video recorded.

7.5 Picking Minimum Functionality

The basic requirements were picked from the large pool of things which could be thought of as being useful for doing maintenance work on the same vehicle over multiple years, and multiple vehicles, from the dream version. Some of the ideas which were cut was functionality to keep track of stocks of materials at home. Stock would be anything which were distinct for a service item and was constant. This could be an oil filter but not the paper towel which may be required to clean up any spillage.

The minimum functionality required was filtered based on the requirement to be able to define a service item with materials needed, torque values for nuts and bolts, and settings for things such as carburetors plus some related items. These service items would need to be able to be associated with a distinct motorcycle. Functionality for creating a record of this service having been made and being able to discern when the last time a particular service item was carried out.

8 Building the Prototypes

Arvola talks about prototype driven development and describes the design being driven forward through both mockups and prototypes. The definition of a prototype is fluid and contains both low detail mockups on paper as well as high detailed interactive digital versions.[Arv14, p. 11-13]

8.1 Non-interactive Prototypes

The very first ideas for the UI were non-interactive to minimize the amount of work needed to create something visual. As the ideas became clearer, they moved to a higher level of detail with more structure. With the level of detail changing, the most appropriate tool for designing the UI changed.
8.1.1 Low Quality Sketches During Discussions with Project Owner

The initial layouts were drawn on paper during this discussion with the project owner regarding the possible features of a final dream version. These sketches were little more than groupings of information which were related and would be useful to have close to each other when using the program by the end user. In these drawings things such as links were grouped together. Only the type of link (link to the service manual, link to bike forums etc.) was drawn, the text field where the user could enter the link was not.

From the LoFi drawings a mental image with an idea for how the user interface could look appeared in the project author’s head. The mental image was then converted into digital images using a vector illustration program.

8.1.2 Designing the Layout with Images

Before the application was created in a digital interactive medium the layout and design choices were made using illustrations created in a digital image creator called Affinity Designer which is a competitor to the more known Adobe illustrator.[Ser22][Ado22]

The images were created based on a vague feature-set from the dream version. The design language is minimalistic, based on a single color, a muted cyan color to be used on interactive elements. Static elements have three options, pure black, pure white and a light grey. The colors are shown in Figure 2. The first page designed was the page to enter information regarding the motorcycle and the associated service items are to be entered by the user. The second image to be created was the landing page which would list all motorcycles created by the user from where the user could navigate to any of them or create a new motorcycle.
8.1.3 From Images to HTML and CSS

After creating these two pages which contained a lot of reused elements the project’s author came to the realization that it would be faster to create the interface layouts using HTML and CSS as much of the elements could be reused in quick successions once the element had been created.

The first and second image was remade as static web pages with HTML and CSS. This generated most of the assets needed for the project with only minor modifications needed each time a slightly different version was needed. The assets were used to create the two last pages, the page for viewing information regarding the specific motorcycle, and the page to register information regarding an instance of a service item having been carried out.

Many of the assets were reused when creating a third view which was the page the user could see the details of the motorcycle and choose which service item to register that a service had been carried out.

8.2 Interactive Prototypes

After the three pages had been created in HTML it was time to make them interactive to see if design choices worked as expected. The goal of the first iteration of the interactive prototypes was to be able to fulfill the minimum requirements before user testing.

8.2.1 Implementing the Design in Node.js

The project was created using the command `npm init vue@latest` (which was version 3.2.31). This opened a menu for the official scaffolding tool, which prompts you with several options. For this project two options where chosen. First was to include TypeScript and the second was to use the Vue Router to create a SPA. Enabling TypeScript enables a level of type checking during development which JavaScript does not offer. When the code is compiled by the Vue package, TypeScript is converted into JavaScript. During this project, in development and user testing, dev mode has been used which recompiles every time the source files are saved.
8.2.2 From HTML to Vue Components

The HTML pages with their associated CSS files were converted into Vue components which only required minimal modification. Elements of the page design, such as one entry in the list of motorcycles to choose from on the landing page, were broken down into a subcomponent which was combined with other subcomponents into a larger component. The larger components were used as separate pages that the user would navigate to as different pages. Because Vue handles the routing, the page would not be reloaded when switching between these pages, rather the content would be swapped based on the current URL. Vue router handled the routing because the website is a SPA.

The HTML pages did not include any interactivity, nor did it use any JavaScript to control or send data in between different elements on the pages. The interactivity was added during the process of turning these static HTML pages into the Vue components which would be used for the prototype.

8.2.3 Placeholders for Computed Values

Computed values such as days until service is due, distance until service is due, and total price of a service item are not implemented in the initial prototype, rather static placeholders are in their place. This is to reduce development time and was not considered as part of the minimum requirements. In the last prototype a solution is used to compute days and distance to the next service. But the algorithm of the solution is incomplete and was just implemented out of curiosity.

8.2.4 Storing Data

Data which was recorded by the user regarding their motorcycle in any capacity, either when making the motorcycle or when entering service information is stored in localStorage in the web browser. localStorage allows for more data than cookies but it’s still somewhat restricted in terms of storage of any information larger than text. The maximum allowed storage per domain is 5 megabytes. localStorage is not the same thing as files stored on the user’s computer which the user might chose to access, but rather is a part of the web browser specifications in accordance to web storage API. The localStorage has no expiration date unlike cookies and can only be accessed from the domain which stored the data.[Dig22]

In this project the information is stored with in the Vue object as data, that data is then converted to JSON which is then converted to a string of characters using the
8 Building the Prototypes

Using the JSON.stringify function built into JavaScript. This means that the information can be extracted and put into the localStorage which only stores string data with a key. To read the data from local storage when opening a new session, some custom functionality was created for reading general data about the application from localStorage. There is also custom functionality for computing keys for the stored motorcycles and their associated service history.

Adding the periodic maintenance schedule for the Kawasaki ER-6f 2009 EX650C9F without adding all materials needed, tools, and torque values, for items that needs to be removed to gain access in order to perform all the service items totals 16500 characters. These characters are not limited to ASCII but a larger pool of characters leading to possible storage beyond 16500 bytes required to store the information.

8.2.5 Onto User Testing

Once the first prototype met the minimum functionality requirements, the prototype moved onto user testing. Once user testing had been completed, the prototype was developed to version two to address issues and concerns found during user testing.

When version two had addressed the issues of the predecessor it moved on to user testing. Once again issues which arose during user testing was addressed in version three which was the final version.

8.2.6 Functionality beyond Minimum Requirements

There are two additional blocks of information where the user can enter data when creating a new motorcycle which can also be read on the information page for the motorcycle. The first of these are required to enter enough information to discern which motorcycle is which. The second is not needed to meet the minimum requirements and contains all links related to the motorcycle. These are links to forums and bike related documents. These blocks are the same elements with the labels changed and are linked to different data values. The second, unrequired information block, turned out to be helpful during user testing to exaggerate a problem which was found regarding information overload in later data fields.

When servicing one’s own motorcycle a home mechanic may opt to user third party material for several reasons, including price and higher quality. The program caters for this by allowing the user to enter the original equipment manufacturer’s (OEM) name and part number as well as a third-party option as their preferred option. On the page for recording the service having been done, the user can select to use the OEM parts,
their preferred option, or they can choose to enter different values for each item in the materials list. The option to select material is only available if there are materials associated with the service item, if there are no materials, this menu will be hidden.

The user can create settings for each service item, both what the original setting is but also their preferred setting. When recording an instance of a service item having been carried out, the user has the same options as regarding materials. If there are no settings this menu will be hidden.

9 User Testing the Prototypes

The users were motorcyclists of varying level of experience with both computers and servicing motorcycles. Some had extensive knowledge of computers such as software engineers, in contrast to some which only use their computers for gaming and daily life things such as banking, reading, and writing documents. In the same manner the experience on servicing motorcycles goes from never having done anything beyond greasing the drive chain to some users which have completely rebuilt multiple engines on both motorcycles and other types of vehicles.

The tasks were used to evaluate the performance of the interface in the number of mistakes and how many extended pauses were made by each user while trying to complete the tasks.

9.1 How to Users were Chosen

One of the users was the project owner which tested both prototypes’ versions one and two. The rest of the users were friends of the projects author and project owner which is why subjective opinions without pointing to specific elements of the UI were ignored as they might be skewed.

The friend groups of the projects author and project owner heavily overlap and as such the number of available users were limited if friends of the project’s author were to be excluded. Both are also involved in post drivers license education for motorcyclists with hundreds of participants yearly where both parties have some form of authority role. Because of this, finding non-biased users would pose a significant hurdle and instead the choice was made to focus on objective data.
9.1.1 Users Excluded from the Results

Two user testing results are excluded from the report. They both tested the third iteration of the prototype and had severe issues completing the tasks. The first of the two had issues with the part of the UI to register a service item that had been serviced which no other user had issues with. The second user had misunderstood the target demographic and was trying to complete the tasks from the view of a service center rather than a home mechanic. The reason why their results were ignored is because they were the only two users which completed the testing late at night after consuming some quantity of alcohol.

9.2 The Physical Testing Environment

Three of the users tested the program at a clubhouse, the rest were tested at the users’ homes. None of the users had seen anyone else use the program prior to completing the tasks themselves. The users sat in chairs and the laptop was placed on a table in front of them, with the project author standing behind them at an angle recording the monitor of the laptop.

9.3 Establishing the Proficiency of the User

Before the user was allowed to see the prototype, they were verbally asked questions from a questionnaire designed to establish their computer proficiency and their proficiency as home mechanics. The questions were to self-assess on a scale from 0 to 10 on how comfortable they were with certain concepts or tasks.

9.3.1 Computer Proficiency

The user testing was carried out on the project author’s laptop with a trackpad. And there are no hotkeys available in the program, but because the program was a website shown in the browser the hotkeys of the web browser was still available.

1. General proficiency with computers
   0: No prior experience using desktops, laptops, tablets, or smartphones.
   10: Have worked with designing software or hardware for any of the aforementioned platforms.
2. Proficiency with laptops with trackpads
   0: No prior experience with trackpads.
   10: Uses multiple laptops with trackpads which you are not the sole user on every week.

3. Reliance on hotkeys
   0: Only clicks or taps (depending on device) and only using keyboard to enter text.
   10: Immediately trying out hotkeys or searching the user manual, or similar, to find if a program offers hotkeys when first using an unfamiliar piece of software.

4. Experience designing digital interfaces
   0: Never considered how to make a digital interface.
   10: Designing digital interfaces is my profession.

### 9.3.2 Motorcycle Servicing Proficiency

Beyond the two questions with answers on a scale between 0 and 10 there were to additional questions. The third question was to quantify the number of motorcycles ever serviced by the user. The fourth question was free form to describe how they currently kept track of the service history of their motorcycle.

1. Comfortable servicing their motorcycle
   0: Never done any service work and no theoretical knowledge.
   10: Having done services to both computer systems and internals of the engine on multiple motorcycles.

2. General servicing mechanic appliances and other vehicle types
   0: Never try to fix broken mechanical items.
   10: Serviced both computer systems and complex mechanical items, such as other types of vehicles.

### 9.4 Tasks for Users to Perform

The initial tasks were simple, single step, to find and read out loud specific information. This information included such as “what is the manufacture of the first motorcycle in the list of motorcycles”. This information is not labeled and requires prior knowledge of makes of motorcycles, but this manufacture (BMW) is one of the largest vehicles
manufactures in the world and would be recognized by any motorcyclist as one of their models was the most sold motorcycle in Sweden 2013.[Bik14]

Then tests moved on to entering information into the interface. One such tasks were to record that an oil change had been made to the previously mentioned BMW on the date of the user test with the odometer at 3340km. The last types of tasks were multistep with parts to complete in multiple places of the interface.

### 9.4.1 Preexisting Data

In normal use, the user would come to an empty program without preexisting data. But to test the program preexisting data was used during the user testing. If the user had created a motorcycle and provided the information of which manufacture had made the motorcycle, and then being asked to identify the manufacture of the motorcycle it would be impossible to know if they had just said what they entered or if they had read it from the interface as intended.

This preexisting data was reset between each user. The data wasn’t a full service-schedule with lots of entries for dates when service items had been carried out, but instead was a limited subset.

### 9.4.2 Task List

The list of tasks the users had to complete was the same for all versions of the prototype. There was the idea to skip tasks which no one had any issues within previous iterations, but this was unintentionally not implemented which was lucky. To version two the ability to modify existing service items or adding more service items was added. The implementation made it so that all users struggled with task 3 which had not been any issue in version one.

1. What’s the manufacture of the first motorcycle in the list of motorcycles?
   
   Note: On the landing page which the user was presented the manufacture is named but not labeled. By clicking on the motorcycle there is more information but the manufacture is labeled there.

2. When was the last oil change on the motorcycle with the nickname ”Fairing”?
   
   Note: What the nickname of a motorcycle isn’t labelled on the motorcycle, but as with manufacture is labelled if one clicks on a motorcycle.
3. Create a record that an oil change has been made on the first motorcycle in the list of motorcycles with 3340km on the odometer and today’s date. 
*Note: The user was not told to navigate out of the previous page and if they had not done it already they had to do it now without instructions. The ”today’s date” was the date of the user testing.*

4. Add a new motorcycle with the nickname ”Targus” and the manufacture, model and model year of one of your own motorcycles. 
*Note: The nickname was not important. The user was free to add any other information they wanted to.*

5. Add a new motorcycle with two service items, each with two materials. 
*Note: The user was free to add any other information they wanted to. In the first prototype one could not edit an existing motorcycle which is why this was a task to add a new motorcycle instead of modifying an existing one.*

### 9.5 Free to Speak Their Mind

After completing the list of tasks, the users were invited to freely talk about their experience. If the user did not talk about things which the project’s author had noted as being interesting regarding the user’s performance, they were asked an open-ended question related to it. If they struggled with finding a certain piece of information they were asked to explain where they expected to find it. This is part of what Arvola describes as the “after test”.[Arv14, p. 137] Asking opened ended questions rather than leading questions is something Arvola talks about when describing how to design interview questions.[Arv14, p. 48-49]

### 10 Generations of Prototypes

After completing user testing on the first prototype, changes were made based on the issues users had when trying to complete the tasks. When changes had been made to correct these issues this new version, version two, was tested. After user testing version two, changes were made and a version three, which is the final version, was tested by new users.
10.1 Results from the First Prototype

The first prototype was presented to two individuals at separate times in addition to the product owner also at a separate time. These three people were subject to the user testing as described in the previous chapter.

Images of the different pages as they appeared in all versions one can be seen in appendix Images of Pages During All Versions

10.1.1 How the Users Performed

Task 1: Name the Manufacture. All three of the users were able to identify the manufacturer of the first motorcycle in the list of motorcycles without the manufacturer being labeled. None of the users were mistaken or took more than a few seconds. The user taking the longest did not read the information from the list of motorcycles but rather clicked on the first motorcycle in the list which took them to the bike’s information page. On this page the manufacture is presented with a label.

See Landing Page for version one and two, and View Bike Page for version one.

Task 2: Last Oil Change. The second task involves identifying information regarding a motorcycle with a nickname. The nickname is not labeled but all users were able to identify the correct motorcycle from the list. They all clicked on it without being prompted, to find more information regarding the motorcycle and its service history. When it came to finding the last instance of a certain service being performed, all three first looked at the top of the page where the list of service items is listed as to record a new instance of the service item. Two of the users quickly identified this list as not providing the information they were looking for and looked further down on the page where they found the service history and was able to provide the information requested by the task.

The third user was distracted by the record recording list and clicked on the service item for which they were asked to identify the last record for. After not finding the information they were looking for, they clicked “save record” rather than the “go back” button. This created a service record without information as there is no protection against this mistake in the first prototype. They did not accidentally click “save record” rather than “go back” but did it intentionally. Saving a record puts the user back to the bike information page. The user scrolled down, found the service history section but noted they were not able to identify the date as the record was not clickable. They then went back to the page to create a new record of the service, clicked “save bike” yet again to go
back. Took a bit more time and then looked at the service history and identified the information the task requested.

**Task 3: Register an Oil Change** All users were able to record that the motor oil on the first motorcycle in the list with the requested mileage and date without issues. One of the users asked about the options for selecting which materials they should use and were told to use preselected option.

All users clicked to save the record rather than the “go back” button which is good.

See Record Instance Page for version one.

**Task 4: A new Simple Motorcycle** None of the users had any issues with this prompt regarding the UI itself. One of the users found the scrollbar in Firefox (the web browser the test was performed on) to be hard to control.

**Task 5: A new Complex Motorcycle** This task exposed several interpretation issues where all the users found one or more ways for straying from the intended path.

The first user added an empty service item while trying to figure out how to add a service item. After which the user added two service items as requested but added the materials separately instead of as being part of any of the service items.

The second user added a material to a service item but did not save the service item before saving the motorcycle which means that the material and the last service item was not saved.

The third user started doing things while still receiving instructions (as they later explained as to not forget the instructions). They created the motorcycle without any service items and then opened the new motorcycle from the list of motorcycles and tried to add service items from the bike’s information page. After a couple of seconds trying to find how to back up, they explained that they thought they should add them from the list of motorcycles. They were told they needed to create a new motorcycle and add the service items to it during the making of the motorcycle. The user created a service item but because there was no visible change, they clicked the button to add a service item again, adding an empty service item. They then proceeded to scroll up where they identified that the service item had been added and started creating the second service item. They then saved the motorcycle without saving the second service item.

See Create Bike Page for version one.
10.1.2 Evaluating the Users Performance

Arvola describes how sometimes the specific actions of the user doesn’t always provide the full information. That observations of the entire context may clarify what may be the actual issue rather than what the user perceives to be the issue.[Arv14, p. 50-51] Following is observations and perspectives from the project’s author on the user’s performance and as such is necessarily opinionated.

Task 1: Name the Manufacture. While one of the users did not identify the manufacture from the list of motorcycles, they were still able to find the manufacture quickly. This user generally liked to click things and expected to be provided explicit information from the program.

None of the users reflected on this feature when freely talking about the program after they had completed all the tasks.

Task 2: Last Oil Change. All users were distracted by having the place to record instances of service items at the very top. The reason this feature is at the very top is to create the least amount of friction when recording instances of services, as one might need to record many instances of services one after another if the user has completed a selection of service items at the same time.

The user who had trouble finding the information enjoyed clicking things. Later when speaking freely they said that they wanted to click on the service history item to see the information there. This was even though they could already see the information without the extra click. Regardless, it would be good if the last time an instance of the service item was completed would be listed on the page when recording that a new instance of the service has been carried out. After many years there could be a long list of the same service item having been completed and finding the last time it was made should still be easy to find.

There should be some form of protection against making an empty record of a service item having been completed as it’s not helpful, there would be no reason for a user to not enter any information about mileage or date. This could be achieved with either greying out the save button or informing the user that they had to enter information before being able to save. The “go back” button would still be useable as it does not save a record.

Task 3: Register an Oil Change All the users chose the intended path to record an instance of a service item. Only one of the users reflected over the ability to choose
which material was used.

**Task 4: A new Simple Motorcycle**  The only thing of note is that the user spent a bit of time to figure out they needed to scroll down to the bottom of the page to save the motorcycle. They also took a bit of time to scroll down as they were glancing over the list of things they could add to a service item.

**Task 5: A new Complex Motorcycle**  The long list of options to add to a service item was a significant distraction for all the users. This may be alleviated if all option things (materials, consumables and so on) where check box options to include and when they were check the block for entering the options were displayed. If they were not checked, they would not be visible.

All users had some issue with clicking the button which would save their entry. Both with options such as materials but also with saving the last service item before saving the motorcycle. Currently the language used is “add” rather than “save”, this may prompt a misunderstanding. For the second prototype the decision needs to be made whether to prompt the user that they have unsaved data before clicking certain buttons at different stages or to automatically saved what they have entered.

When adding a service item to a bike it doesn’t take the user to the top which made users confused about what had happened when they clicked the button.

In the first prototype there is no options to edit a motorcycle or the associated service items once the motorcycle has been saved. This meant the one of the users had to create a new motorcycle to complete the task. This would be inconvenient once the program was used by end users.

### 10.2 Turning Version One into Version Two

Nothing regarding the presentation of the manufacture was changed as the results from the first prototype showed this not to be an issue for any of the testers.

**Additional Place to See Last Service Instance.**  When clicking on an item in the location to record an instance of a service having been carried out, it now shows the last time this item was serviced. If the item has never been serviced before then it informs the user of this.
See Record Instance Page for versions two and three.

**No More Empty Service Records.** The button to save the record is greyed out until either a date or mileage for the service has been entered as protection against empty entries being recorded.

**New Button to Modify or Add Service Items.** Below the last entry in the list of services to record an instance of the service having been carried out, a new button was added. Clicking the button shows a new block on the page which allows the user to edit existing service items or add a new one. This is to allow modifications of an already existing motorcycle without having to copy or manipulate the raw text from the export string and then reimport the motorcycle.

See View Bike Page for version two.

**Hiding Options until Selected.** On the page where the user creates a new motorcycle, all sub items to a service item (materials, consumables, tools, torque values, settings, and warnings) are now hidden until the user checks an associated checkbox which displays the sub item. This is to decrease the distraction of the user and to decrease the amount of scrolling needed to navigate the page. When a service item is saved, these check boxes are unchecked which turns the sub items invisible.

See Create Bike Page for version two and three.

**Changing ”Add” to ”Save”.** The wording on buttons saying “add . . . ” was checked to say “save . . . ” to nudge the user to click it before saving the motorcycle.

**Automatically Saving Unsaved Service Items.** When the “save motorcycle” button is clicked the currently unsaved service item and all sub items are saved. This does not include text which is written in textboxes but not saved for the sub items.

### 10.3 Results from the Second Version

Version two was tested by two new users and the project owner.
10.3.1 How the Users Performed

**Task 1: Name the Manufacture.** The interface for reading the manufacture was not changed from the first to the second version of the prototype and none of the users had any issues identifying the correct answer.

**Task 2: Last Oil Change.** An extended pause was recorded for one of the users where they looked at the landing page at different places but was unable to find the information and needed to be prompted that the bikes where clickable. Once they clicked the motorcycle, they had no issues finding the information. As they did not click on anything incorrectly this was recorded as an extended pause instead of an error.

**Task 3: Register an Oil Change** This task went from being completed without errors or pauses by all users in the first version to all three users making the same mistake which was a major one. Adding the button to allow adding and editing of service items after a motorcycle had already been created misled all the users. When asked to record that an oil change had been carried out, they all clicked on this new button rather than the item in the list of existing service items.

They all entered the requested mileage in the interval distance text box but stopped when trying to find where to enter the date as there is no place to enter a date when creating a new service item. It was at this point the users realized their mistake, until then they all thought they were doing it correctly.

Only the last user was able to find the correct way of doing it after clicking around for a minute without being prompted. The other two users needed prompting to find the correct way of doing it.

**Task 4: A new Simple Motorcycle** As with version one, this was not a difficult task for any of the users.

**Task 5: A new Complex Motorcycle** The first user to test the second prototype struggled a lot but was able to complete the task eventually. The struggle likely stemmed from clicking “save motorcycle” instead of clicking “save service” which put the user back to the landing page. The user navigated to the correct motorcycle but instead of using the button to add or modify the list of service items they clicked on the one service they had recorded for the motorcycle and were taken to the page to record an instance of the service. After this the user navigated backwards to the landing page and entered
the wrong motorcycle. But on this motorcycle, they did click the correct button to add
more service items to the motorcycle and entered the information requested. They did
struggle to click the correct combination of save buttons to actually save the information
entered. The combination of save buttons needed to be clicked changed after this user
test to save if any of the “save changes”-buttons were clicked.

The last user misunderstood the task at hand by believing the different service items
should be for two separate motorcycles and that after creating these service items they
should record an instance of the service having been carried out. As such there is no
way of scoring the actions of the user as it’s impossible to tell what the goal was of the
users’ actions.

10.3.2 Evaluating the Users Performance

There are no notes for tasks 1, 2, and 4 as they effectively were performed as intended.

Task 3: Register an Oil Change  Two of the three users all read the label of the
button out loud and still chose to click it believing it would let them record an instance
of the service having been carried out. Between the second to last and last user the text
was changed from “add or edit service items” to “modify service items” but there was
no impact on the user’s action.

In the first version, only the service items were clickable on the page to view details for
the motorcycle. In the second version the button was added to add or edit service items
beneath the list of service items, and it took the focus of all the users. Even after reading
the label which didn’t match the task they had been given they still chose to click it.

After all the tasks had been completed and the UI was explained to them, they all un-
derstood it and stated that the UI made sense.

Task 5: A new Complex Motorcycle  The first user likely struggled because of
a choice being made regarding how to save additions or modification of service items.
The choice was to be conservative and only save when very specifically being asked to
do so. Due to the struggle of the first user this choice was reversed to being liberal with
when to save. After the first user clicking the previously intended way would still save
but also clicking “save service” would save the service.

The tasks could be given in written form instead of verbal, but this would likely decrease
the freedom of the user as they would see the way the thought process of how the tasks

34
is intended to be completed.

10.4 Turning Version Two into Version Three

The changes made during user testing of version two is not covered in this section. The changes listed here were made after completing the user testing with the three users with version two.

10.4.1 Adding the Ability to Edit the Entire Motorcycle

In version two, the ability to add or modify existing service items was added. The implementation of this feature was detrimental to all users and the editing was limited to service items. To version three the button for adding or modifying service items was removed. Instead, a new button beneath the bike in the list of bikes was added for each bike which allows full editing of the motorcycle. This button takes the user to the same page as to create a new motorcycle but populates it with the existing information for the motorcycle. Saving the motorcycle overwrites the existing motorcycle and doesn’t add a new motorcycle. The associated service history is kept with the bike and is not removed.

See Landing Page for version three, View Bike Page for version three, and Edit Bike Page for version three.

10.4.2 Added some Invalid Browsing Protection

Most users had refrained from using the web browsers buttons to navigate backwards and forwards without being told to avoid using them. Instead, they used the back buttons of the program, but occasionally a user would use the web browsers buttons. Depending on the page the user was at when they clicked the back button of the web browser they might end up on an empty page as this form of navigation was not valid. The difference is in what data is sent between the pages on navigation depending on if the web browsers buttons are used or the buttons in the program.

As a quick solution to this issue, to version three some invalid browsing protection was added. This was try-catch blocks of code where if the required data was not passed along to the page, the user would be routed to the landing page. The case of the data not being passed only happens when the user uses the web browsers navigation buttons.
10.5 Results from the Third Version

This version was tested by two new users.

10.5.1 How the Users Performed

Neither user had any issues with tasks 1 and 3. User 6 informed the project’s author that they did not have their reading glasses. To compensate the UI was scaled up using the web browsers built in zoom feature.

Task 2: Last Oil Change. User 6 clicked on the item to record an instance of a service item where the requested information is available since version two, but they did not spot it and or recognize the information. They then quickly went back to the bike info page and found the information without issue. This was classified as an extended pause rather than an error since the user didn’t commit any incorrect actions.

Task 4: A new Simple Motorcycle User 6 clicked to save service item before clicking save motorcycle which saved an empty service item.

Task 5: A new Complex Motorcycle User 6 clicked an extra time on save material which saved an empty material, but beyond this had no issues completing the task.

User 7 saved the motorcycle before completing the task and was asked to add an additional service as they had only added one. They first clicked to view the bike before realizing the edit button wasn’t there. They then went back to the landing page where they found the correct button. They added the requested second service item and saved the motorcycle.

10.5.2 Evaluating the Users Performance

User 6 had some troubles navigating using the trackpad. Their zoomed in view demonstrated that without multiple elements of the page being viewable at the same time, orienting on the different pages becomes significantly more difficult.

There are no notes for task 1 and 3 as none of the users had any issues with either of them.
**Task 2: Last Oil Change.** Currently the text regarding the last time an instance of the service item was carried out on the page to register a new instance of a service item is not bolded, or in any other way made to stand out. The text is at the very top below the header and is not difficult to read once the user is aware that it’s there.

**Task 4: A new Simple Motorcycle** Before this program is ready for end users, protection and warnings regarding saving empty entries and leaving pages without saving entered information should be added.

**Task 5: A new Complex Motorcycle** User 7 was effectively looking at the place where service items could be added or modified during the previous version, when trying to add an additional service item to the motorcycle.

# Evaluation Results

In the chapter Picking Minimum Functionallity there are four features required to fulfill the threshold of passing. The user had to be able to create multiple motorcycles, each with a separate list of service items. These service items had to be able to retain certain pieces of information, such as materials needed. The user had to be able to record that an instance of a service item had been carried out. And the user had to be able to recognize when the last time an instance of a service item had been carried out.

Even the first iteration of the prototype passed these requirements, but the ease of use was defined in Setting the Bar as no task standing out from the rest in terms of errors made while trying to complete a task. The last task, where the users had to create a motorcycle with service items, stood out with all users having varying levels of difficulty.

Version two addressed some of the issues users had in version one. But the implementation of being able to add or modify service items of an existing motorcycle created massive issues for all users. Going from no errors or extended pauses to being the task with most issues of any version of the prototype.

The third version has no task with more than one error per user. By the metric of no task standing out in the error rate, they have all been completed to an acceptable standard in terms of ease of use.
12 Results and Discussions

The final product of this project is an interactive UI, tested by users of varying levels of proficiency regarding both computers, and servicing motorcycles. Through three iterations the UI has been improved and the last version passed the requirements set out in the beginning of the project.

There are features which should be included in future development, such as protection against empty entries and correctly calculating time and distance until the next service item is due. But the last version of the prototype had errors committed by the user which they could figure their way out of without intervention from the project’s author. None of the users got any instructions for how to perform the tasks, they were only given the end goal and they completed these tasks.

The presentation of the materials, torque values, and similar was not tested. This information is presented in a list format where each category has a bold header and below is the list of items for the category with its associated information. E.g., the name of the setting followed by the value of the setting. Finding the information, such as “what torque value for this thing on this service item”, was deemed trivial.

12.1 A Modified Development Route

The initial goal was to produce one prototype and test it with a larger pool of users, perhaps equaling the total number of users testing the different versions of the prototypes in this project. This would have resulted in more reliable data than the current 2-3 users per version. But this would have meant that improvements would have been left to future work instead of working through different solutions. The results for the last task were not positive in the first version. These smaller groups of users allowed for refining ideas to the point of version three being good.

The number of issues induced by the implementation of the ability to add or modify service items was a surprise. Even during testing, the fact that users read the text out loud and still chose to commit an incorrect action was not predicted. User 7 looked in the location where this button used to be to add a service item, which did feel redeeming.

12.2 Designed for Its Audience

The final prototype has some way to go until it’s ready to be used by end users, but it’s currently a unique product which fills the gap left by competing products. It’s built
for someone who manages their own motorcycles, being able to track the history of services. But it also contains a quick reference for values and materials needed, to avoid having to pull out the service manual. Based on the test results using the metrics set by this project, it is user friendly. This means that using the program a few times a year should not require reading a manual every time to relearn how to use the program.

### 12.3 Not by the Book

The workflow presented by Arvola targets, in large parts, a team of UX developers whereas this project has been carried out by a single person which also took care of development which is considered post hand-off in the book.

Most of the steps described in the book related to user testing concerns earlier steps in development than was tested in this project. If user testing was carried out in the same stage of development as in the book it would have been carried out with drawn images on paper with the people or person hosting the user test handling logic for what is presented to the user.[Arv14, p. 131-134]

One of the major differences between the book and the method of the project is the lack of pre-design interviews.[Arv14, p. 42-44] If the audience’s needs are not known by the UX team, then they need to learn. But in this project the project owner was a stand-in to provide this information. Combine the project owner’s depth of knowledge with the project’s author decent understanding of the subject and there was limited amount of information which may be found by talking to more people. There could be information specific to certain sports, such as road racing, where additional knowledge might be insightful.

### 13 Conclusions

The prototype fills a gap in the market by the combination of features it offers. The home mechanic gets a program tailor made to their needs. The ability to have multiple motorcycles, the ability to describe the details of each service item, and the ability to record when an instance of each service item has been carried out are all available. The design of these features is intuitive according to the user testing carried out in this project.

Creating multiple versions of prototypes and testing with a small user group for each version, allowed this project to progress to the point of no task posing a much larger
This program isn’t going to replace the need for a service manual or general knowledge of the home mechanic, but it may replace other software which is used to keep track of expenses of vehicles or full CMMS solutions. Service centers and professional mechanics will not end up using this software in their work as it’s not suited for their needs. But they may opt to use it for their own personal vehicles.

14 Future Work

There is large amount of development for the prototype of this project to reach the dream version envisioned in the beginning of the project between the project owner and project author. But the features designed during this project are the core features which the other features build upon. The ability to keep a record of stock and knowing if there is material which needs to be purchased prior to starting the service item requires the ability to define materials.

The dream version doesn’t have to be achieved for the program to be useful to the target audience. With the addition of protection against empty entries, the ability to remove motorcycles, service items and so on, the program will be fully usable.

How the program will be deployed also needs to be figured out. Currently it requires a fair bit of knowledge which is unlikely to be known by the target demographic to install the setup needed. But because of the implementation as a website the work done in this project is compatible with many solutions. It could be deployed as a PWA or program for any of the currently big platforms (both desktop and mobile).

While the current design is intended for desktops, the logic behind the displayed UI could be reused in layouts designed for mobile which would increase the possible number of users. The developing world to large extent skipped desktop computers and went straight to smartphones. Combining the design of a mobile layout with the ability to select different languages could make this program usable by an audience of literal billions of users.

References

References

web/20220627122851/https://geekflare.com/single-page-applications/


References


References


A User Tasks and Their Results

1. What’s the manufacture of the first motorcycle in the list of motorcycles?

2. When was the last oil change on the motorcycle with the nickname “Fairing”?

3. Create a record that an oil change has been made on the first motorcycle in the list of motorcycles with 3340km on the odometer and today’s date.

4. Add a new motorcycle with the nickname “Targus” and the manufacture, model and model year of one of your own motorcycles.

5. Add a new motorcycle with two service items, each with two materials.

A.1 Results from the First Version

<table>
<thead>
<tr>
<th>Tasks</th>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, errors</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1, extended pauses</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2, errors</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2, extended pauses</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3, errors</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3, extended pauses</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4, errors</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4, extended pauses</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5, errors</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>5, extended pauses</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
A.2 Results from the Second Version

<table>
<thead>
<tr>
<th>Tasks</th>
<th>User 2</th>
<th>User 4</th>
<th>User 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, errors</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1, extended pauses</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2, errors</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2, extended pauses</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3, errors</td>
<td>4</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>3, extended pauses</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4, errors</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4, extended pauses</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5, errors</td>
<td>5</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td>5, extended pauses</td>
<td>1</td>
<td>0</td>
<td>N/A</td>
</tr>
</tbody>
</table>

The two N/A’s for user 5 on task 5 is because the user misunderstood the task and as such the goals of the clicks were unknown to the observer.

A.3 Results from the Third Version

<table>
<thead>
<tr>
<th>Tasks</th>
<th>User 6</th>
<th>User 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, errors</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1, extended pauses</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2, errors</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2, extended pauses</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3, errors</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3, extended pauses</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4, errors</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4, extended pauses</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5, errors</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5, extended pauses</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

B Users Answers to Questionnaire

B.1 Quantifiable

Computer proficiency Q1-4 and bike proficiency Q1-2 are on a scale on 0-10 whereas bike proficiency Q3 is any positive integer.
Table 1: Computer Proficiency and Bike Proficiency

<table>
<thead>
<tr>
<th>User</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>8</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>9</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>8</td>
<td>6</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>300</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>7</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>10</td>
<td>10</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

B.2 Free Form - How You Currently Keep Track of Maintenance

<table>
<thead>
<tr>
<th>User</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In paper form, the OEM service manual and comparing to current age and mileage.</td>
</tr>
<tr>
<td>2</td>
<td>In paper form, mostly recalling from memory when the service was last completed.</td>
</tr>
<tr>
<td>3</td>
<td>Digital spreadsheet, which they created themselves.</td>
</tr>
<tr>
<td>4</td>
<td>In paper form, writes it down in the service booklet.</td>
</tr>
<tr>
<td>5</td>
<td>In paper form, with no dedicated form of collecting the papers.</td>
</tr>
<tr>
<td>6</td>
<td>Mentally remember.</td>
</tr>
<tr>
<td>7</td>
<td>Mentally remember.</td>
</tr>
</tbody>
</table>

C Images of Pages During All Versions

Note that the majority of text is in Swedish. Text which is in English was not used in any of the user tasks.

C.1 Image of Pages During Version Two

Landing page was unchanged from version one.

C.2 Image of Pages During Version Three

The pages to record instance of service item and create a new motorcycle are the same as in version two.
### Figure 3 Version One and Two, Landing Page with Bike List

<table>
<thead>
<tr>
<th>Bike Model</th>
<th>Current km/miles</th>
<th>Km/miles to service</th>
<th>Time until service</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMW - S1000RR 2009</td>
<td>0</td>
<td>400</td>
<td>N/A</td>
</tr>
<tr>
<td>Fairing</td>
<td>0</td>
<td>400</td>
<td>N/A</td>
</tr>
<tr>
<td>Kawcan</td>
<td>0</td>
<td>400</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Export**  **Import**
C Images of Pages During All Versions

Figure 4 Version One, View Bike Page

Figure 5 Version One, Record Instance of Service Item Page
**C  Images of Pages During All Versions**

**Figure 6** Version One, Create Bike Page
### Registrera att service genomförts

<table>
<thead>
<tr>
<th>Namn på föremål</th>
<th>km till nästa service</th>
<th>Tid till nästa service</th>
<th>pris</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byta motorolja</td>
<td>NaN</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>Byta brommsäckarna</td>
<td>NaN</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>Inspektera länkage</td>
<td>NaN</td>
<td>10</td>
<td>100</td>
</tr>
</tbody>
</table>

#### Länkar
- Kart-katalog
- Service manual
- Extra katalog
- Specifikationer
- Forum

#### Motorcykel information
- Smeknamn: Faering
- Tillverkare: Suzuki
- Modell: Hayabusa
- Modelår: 2021
- Färgstil: Heritage

#### Servicehistorik

<table>
<thead>
<tr>
<th>Namn på föremål</th>
<th>km</th>
<th>datum</th>
<th>pris</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inspektera länkage</td>
<td>111</td>
<td>2022-04-05</td>
<td></td>
</tr>
<tr>
<td>Byta brommsäckarna</td>
<td>0</td>
<td>2009-05-21</td>
<td></td>
</tr>
<tr>
<td>Inspektera länkage</td>
<td>213</td>
<td>2002-04-20</td>
<td></td>
</tr>
<tr>
<td>Byta motorolja</td>
<td>555</td>
<td>2011-05-05</td>
<td></td>
</tr>
</tbody>
</table>

---

**Figure 7** Version Two, View Bike Page
**Figure 8** Version Two, Edit Service Items View

**Figure 9** Version Two and Three, Record Instance of Service Item Page
Figure 10 Version Two and Three, Create Bike Page

Figure 11 Version Three, Landing Page with Bike List
### Registrera att service genomförts

<table>
<thead>
<tr>
<th>Namn på föremål</th>
<th>km till nästa service</th>
<th>dagar till nästa service</th>
<th>pris</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte motorolja</td>
<td>NaN</td>
<td>NaN</td>
<td>100</td>
</tr>
<tr>
<td>Byte bromsprotsika</td>
<td>NaN</td>
<td>NaN</td>
<td>100</td>
</tr>
<tr>
<td>Inspektéra läckage</td>
<td>NaN</td>
<td>NaN</td>
<td>100</td>
</tr>
</tbody>
</table>

### Länkar

- Parts-katalog
- Service manual
- Extra katalog
- Specifikationer
- Forum

### Motorcykel information

- Smeknamn: Faring
- Tillverkare: Suzuki
- Modell: Hayabusa
- Modellår: 2021
- Färgklas: Heritage

### Servicehistorik

<table>
<thead>
<tr>
<th>Namn på föremål</th>
<th>km</th>
<th>datum</th>
<th>pris</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inspektéra läckage</td>
<td>111</td>
<td>2022-04-05</td>
<td></td>
</tr>
<tr>
<td>Byte bromsprotsika</td>
<td>0</td>
<td>2009-05-01</td>
<td></td>
</tr>
<tr>
<td>Inspektéra läckage</td>
<td>233</td>
<td>2022-04-20</td>
<td></td>
</tr>
<tr>
<td>Byte motorolja</td>
<td>300</td>
<td>2022-05-09</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 12** Version Three, View Bike Page
Figure 13 Version Three, Edit Existing Motorcycle Page