Report #1

For

“Better Automobile Inventory Management”

CSCI441\_VA

Software Engineering

Fall 2019

<https://github.com/gculver/SoftwareEngineering_FinalProject>

Team:

Paul Whitely

James Cox

Grant Culver

September 9, 2019

Revision History:

|  |  |
| --- | --- |
| Version No. | Date of Revision |
| v.1 | 9/7/2019 |
| v.2 | 9/23/2019 |
| v.3 | 9/23/2019 |

**Contents**

Contents

1 Customer Statement of Requirements

1.1 Problem Statement…………………………………………………………………. 4

1.2 Glossary of Terms…………………………………………………………………... 7

2 System Requirements

2.1 Enumerated Functional Requirements…………………………………………... 8

2.2 Enumerated Non-Functional Requirements……………………………………… 9

2.3 On-Screen Appearance Requirements………………………………………….. 10

3 Functional Requirements

3.1 Automotive Professionals and Administrative Staff ..…………………………... 11

3.2 Actors and goals…………………………….……………………………………… 12

3.3 Use Cases……………………………….………………………………………….. 13

3.3.1 Use Case Diagram..………………….………………………………………….. 15

3.3.2 Traceability Matrix...………………….………………………………………….. 16

3.3.3 System Sequence Diagrams….…….………………………………………….. 16

4 User Interface Specification

4 User Interface Specification……………..…………………………………………... 17

5 Domain Analysis

5.1 Domain Model……………………………………….. ..…………………………... 21

5.1.2 Concept Definitions……………………….……………………………………… 21

5.1.3 Boundary Concepts.………………….………………………………………….. 21

5.1.4 Association Definitions..…………….………………………………………….. 22

5.1.5 Attribute Definitions..………………….………………………………………….. 22

5.1.6 Traceability Matrix……………….…….………………………………………….. 22

6 Project Size Estimation

6.1 Use Case Points………………………..…………………………………………... 23

7 Plan of work

6 Plan of work………..……………………..…………………………………………... 23

8 References

8 References………………………………..…………………………………………... 24

1. **Customer Statement of Requirements**
   1. **Problem Statement**

Throughout the history of civilization and business, there has been a continuous trend towards increased sophistication and efficiency. Sometimes these advancements are small and seemingly insignificant and other times they are drastic and life altering. One can look back at the Gutenberg Press and see that it was a major advancement in education and society. The industrial revolution changed the way products were manufactured and produced. It gave way to the automobile which fundamentally changed society and business by effectively enlarging the geographic area that businesses could compete in. Today, businesses face another revolution and that is the technological revolution that is completely altering the landscape for society and business. It has given way to self-driving cars, space shuttles that can save enormous sums of money by reusing booster rockets and in the retail business, given rise to the “Goliath” of Amazon. Amazon is fundamentally altering the rules and the playing field that small business must compete in. It does no good to debate the equality or the goodness or badness to society of such a mammoth company. Amazon will force, through survival of the fittest, businesses to adapt and modify the way they engage and profit in today's business climate. It is both exciting and perplexing to be a part of. Fortunately, the same technology that has enabled amazon to flourish and prosper will also enable small businesses to perhaps gain an advantage and compete toe-to-toe with a seemingly unbeatable “Goliath”. This has given rise to companies such as salesforce that allow small business to compete on cost with many larger competitors. Open source software has driven down the cost for many businesses. Our company reduced our CRM expense by over $150,000 per year due to these developments. Our proposed solution plans to start the process by enabling our company to economically develop a new car inventory management system with the goal of maximizing shareholder profit by reducing expense and providing relevant, convenient and up-to date information.

The goal of a new car inventory management system is to understand quickly and easily where in relation to a model inventory our current inventory stands. A model inventory is based 3 variables. First, the dealership must be informed as to what the past 3 months sales history looks like regarding total new vehicles sold by brand and then broken down by model. Industry standard is to have a four month’s supply of new car inventory. Second, the dealership must realize what the current inventory levels are by brand and by model. Third, the dealership must be informed as-to whether it is short or long in inventory by model and brand. Currently all of this information is scattered over 3 or more systems and leads to frustration and indigestion. Our customer centric solution is to consolidate these 3 systems into 1 system that will ultimately save the customer money and time.

The monetary savings will largely be due to three main factors. First, the dealership will no longer be stocking inventory it doesn’t need, or that it is too long in. When considering 10,000,000 dollars in inventory at 4.5% interest, stocking the right inventory is not a matter of just increased profitability but ultimately of long-term survivability. Second, savings will be increased by the lessening of time it takes to tackle implementing a model inventory. There will no longer be a need for logging into 3 different systems to obtain information. This will allow user to focus on other business-related tasks and ultimately be more efficient. Thirdly, this will save money due to the unnecessary expense of paying for 3 complimentary but uniquely different software solutions. With the combination of all three features of creating a model inventory software solution, additional or unneeded software can be cancelled.

Additional features of the software solution will include authentication to access the system. System will allow additional users to register or sign-up according to dealership policy. The system will require dealership setting up the recommended months supply with a degree of granularity for specifying months supply not only for vehicle make but also by model. The user interface will be user friendly and will focus heavily on keeping system user friendly with clutter kept to a minimum. The system will utilize a combination of API feeds and database feeds to gather required information for system to calculate model inventory.

* 1. **Glossary of Terms**

To provide better understanding of the contents of this report, listed below are important terms of our proposed software solution:

**Month’s Supply** – Units in stock (at month end) divided by Sales history (in months).

**Model Inventory** – Guideline for stocking new car inventory that takes into account past sales history, user defined month’s supply and the output is recommended number of units in stock.

**Long Inventory** – Inventory in-stock that is above model inventory suggestion.

**Short Inventory** – Inventory in-stock that is less than model inventory suggestion.

**Make** – Manufacturer of vehicle. Ex. Jeep, Dodge, Chevrolet, Toyota, etc.

**Model** – Particular model of vehicle made by a respective manufacturer.

Ex. Wrangler, Silverado, 1500, 4-Runner, Land Cruiser, etc.

**Inventory Manager** – (AKA New Car Inventory Manager) – is principle person concerned with managing state of inventory and forecasting sales in the coming months.

1. **System Requirements**

**2.1 Enumerated Functional Requirements**

|  |  |  |
| --- | --- | --- |
| Identifier | Priority  Weight (Low 1 - 5 High) | Requirement Description |
| REQ-1 | **3** | System shall support user login and authentication |
| REQ-2 | 1 | System shall support registering new users. |
| REQ-3 | 4 | System shall support users to define month supply of inventory by both make and model |
| REQ-4 | 5 | System shall pull in-stock inventory from dealership CRM |
| REQ-5 | 2 | System shall calculate month’s supply of inventory |
| REQ-6 | 4 | System shall compute long or short position of inventory |

**2.2 Enumerated Non-Functional Requirements**

A model called FURPS+ will be used here to qualify software attributes, which stand for functionality, usability, reliability, performance, supportability and the + stands for other possible attributes needed. We will be focusing on the non-functional requirements which cover FURPS+.

Functionality: Our tool should satisfy the needs of the customer such that they no longer have the need to subscribe to multiple inventory tools for their dealership. The software should use an API to gather the information that it needs to calculate model inventory.

Usability: The system should be easy to use and simple to understand. The software to consistently give accurate inventory model information so that the customer can rely on it when making decisions regarding their inventory.

Reliability: The system should give accurate and reliable results regarding the needs of the dealership and the inventory on hand.

Performance: The system to be should have high performance. The process of signing in and receiving data should take no more than 5 minutes.

Supportability: The system is benign and user friendly. The learning curve will be modest and user documentation will introduce designed program usage and will contain a software demonstration video or text. Support will all be available via email to handle software bugs and imperfection.

**2.3 On-Screen Appearance Requirements**

|  |  |  |
| --- | --- | --- |
| Identifier | Priority  Weight (Low 1 – 5 High) | Requirement Description |
| REQ-7 | 3 | User-friendly: The operation of system should be easy to operate with limited training for new users. |
| REQ-8 | 2 | Sign-In Button: Allow authorized users to gain access to system. |
| REQ-9 | 5 | Inventory Selection Drop Down: Users can select make and model of inventory to compare to model. |
| REQ-10 | 3 | UI Appearance: solution should be both functional and aesthetically pleasing to user. |
| REQ-11 | 4 | Inventory Pull Button: Allows user to pull necessary vehicle information based on selection. |
| REQ-12 | 1 | Registration Button: Allow new users to register for system. |

1. **Functional Requirements**

3.1 Automotive Professionals and Administrative Staff

This software is principally designed for senior level, mid-level automotive managers and administrative staff (i.e. accounting department).

In the current business environment and as business evolves to a more data driven approach to inventory management, people are going to make decisions based on previous data, and therefore is a defining characteristic of this software solution. Hence, in this software, all agents make decisions according to previous sales results. The principle stakeholders of this system are mentioned below.

The principle stakeholder of this software proposal is the new car inventory manager or inventory manager. This person’s job exists to minimize carrying costs associated with stocking inventory and maximizing dealership profit by stocking the right make and models based off prior selling history and future market conditions.

The General Manager or Owner of the Automotive dealership is another stakeholder. Their interaction with the system to be would be more monitoring and ensuring utilization of the software system.

The controller would ensure that billing of the software solution is completed within agreed upon terms and would likely use the portal to ensure correct billing and payment processing.

3.2 Actors and Goals

|  |  |  |
| --- | --- | --- |
| **Actor** | **Actor’s Goal** | **Use Case Name** |
| User | Fetch Database Records | FetchData(UC – 3) |
| Admin | Set up user | UserAdmin (UC – 2) |
| User | Set Inventory Settings | InvSettings (UC – 1) |
| User | Calculate Month’s Supply | CalcSupply (UC – 4) |
| Admin | Pay Bill | BillPay(UC – 5) |

3.3 Use Cases

|  |  |  |
| --- | --- | --- |
| **UC – 1** | | **InvSettings** |
| Related requirements | |  |
| Initiating actor | | User |
| Actor’s goal | | Set initial settings for evaluation |
| Participating actors | | System, User |
| Preconditions | | User is active and initial screen is showing |
| Post conditions | | Initial settings are set and evaluation is ready to start |
| **Flow of events for main success scenario** | | |
|  | User (a) selects the menu item “Settings” (b) types in Month’s supply | |
|  | User (a) clicks save | |
|  | System verifies values and stores them in database | |
| **Flow of events for alternate scenario** | | |
|  | User (a) submits invalid Month’s Supply | |
|  | System prompts user for valid Month’s supply | |
|  | User (a) enters valid Month’s supply | |
|  | System verifies values and stores them in database | |

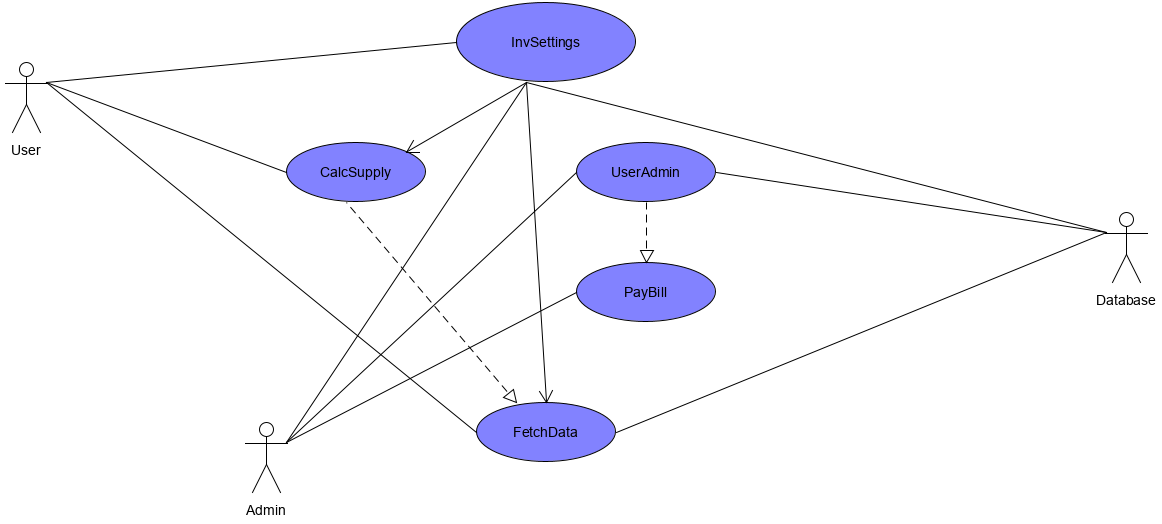
|  |  |  |
| --- | --- | --- |
| UC – 2 | | UserAdmin |
| Related requirements | |  |
| Initiating actor | | Admin |
| Actor’s Goal | | Set up an authorized user account |
| Participating actor | | Database, User |
| Preconditions | |  |
| Post condidtions | | User account is active |
| **Flow of events for main success scenario** | | |
|  | Admin user clicks “Admin” icon on home page | |
|  | Admin user submits required information | |
|  | System stores user in database and shows “Success” to Admin user | |
| **Flow of events for alternate scenario** | | |
|  | Admin user submits user info | |
|  | System notifies Admin user is already active | |

|  |  |  |
| --- | --- | --- |
| UC – 3 | | FetchData |
| Related requirements | |  |
| Initiating actor | | User |
| Actor’s goal | | Pull sales history from database to allow comparison. |
| Participating actor’s | | Database |
| Preconditions | | System is connected to database |
| Post conditions | | Data is pulled from database |
| **Flow of events for main success scenario** | | |
| C | User clicks button to initiate download of data | |
|  | Software solution pulls data from database | |
|  | Software sends success message to users browser | |
| **Flow of events for alternate scenario** | | |
|  | User clicks button to initiate download of data | |
|  | System notifies user connection cannot be established | |

|  |  |  |
| --- | --- | --- |
| UC – 4 | | CalcSupply |
| Related requirements | |  |
| Initiating actor | | User |
| Actor’s goal | | To calculate month’s supply of inventory based off previous sales data |
| Participating actor | | Database, User |
| Preconditions | | Month supply calculation and settings entered correctly |
| Postconditions | | Data is displayed to user |
| Flow of events for main success scenario | | |
|  | User clicks calculate supply button | |
|  | Software calculates Months supply and displays to user | |
| Flow of events for alternate scenario | | |
|  | User clicks calculate supply button | |
|  | Settings are incorrect or not defined software sends error message to interface | |

|  |  |  |
| --- | --- | --- |
| UC - 5 | | Admin |
| Related requirements | |  |
| Initiating actor | | Admin |
| Actor’s goal | | Pay bill |
| Participating actor | | Vendor System |
| Preconditions | | Admin is logged in and can access pay bill page |
| Postconditions | | Admin successfully pays vendor |
| Flow of events for main success scenario | | |
|  | | Admin logs in and access pay bill page |
|  | | Admin enters payment information and clicks submit |
|  | | System successfully process payment and displays success |
| Flow of events for alternate scenario | | |
|  | Admin logs in and access pay bill page | |
|  | Error processing payments and sends Admin user error message | |

3.3.1 Use Case Diagram

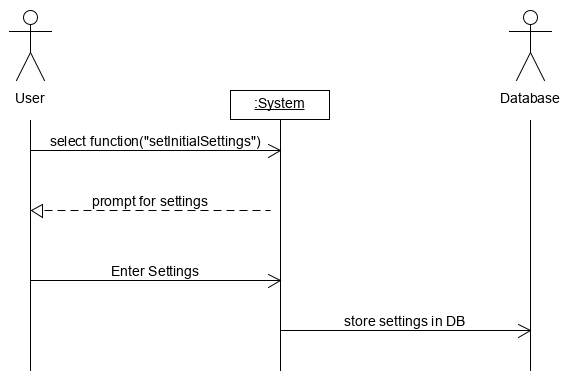


3.3.2 Traceability Matrix

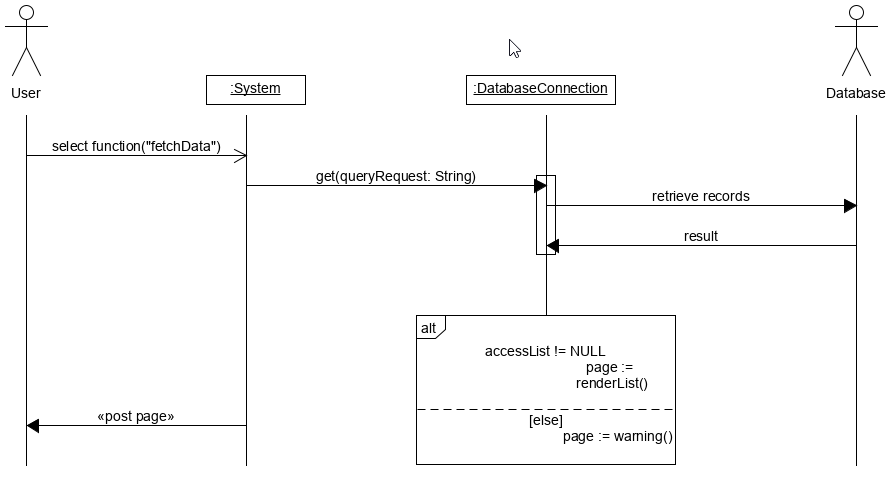
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Req’t | PW | UC1 | UC2 | UC3 | UC4 | UC5 |
| REQ1 | 3 | X |  |  |  |  |
| REQ2 | 1 | X |  |  |  |  |
| REQ3 | 4 | X |  | X |  |  |
| REQ4 | 5 |  |  | X |  |  |
| REQ5 | 2 |  |  |  | X |  |
| REQ6 | 4 |  |  |  | X |  |
| MAX PW |  |  |  |  |  |  |
| TOTAL PW |  |  |  |  |  |  |

3.3.3 System Sequence Diagrams

UC-1: setInvSettings



UC-2: FetchData



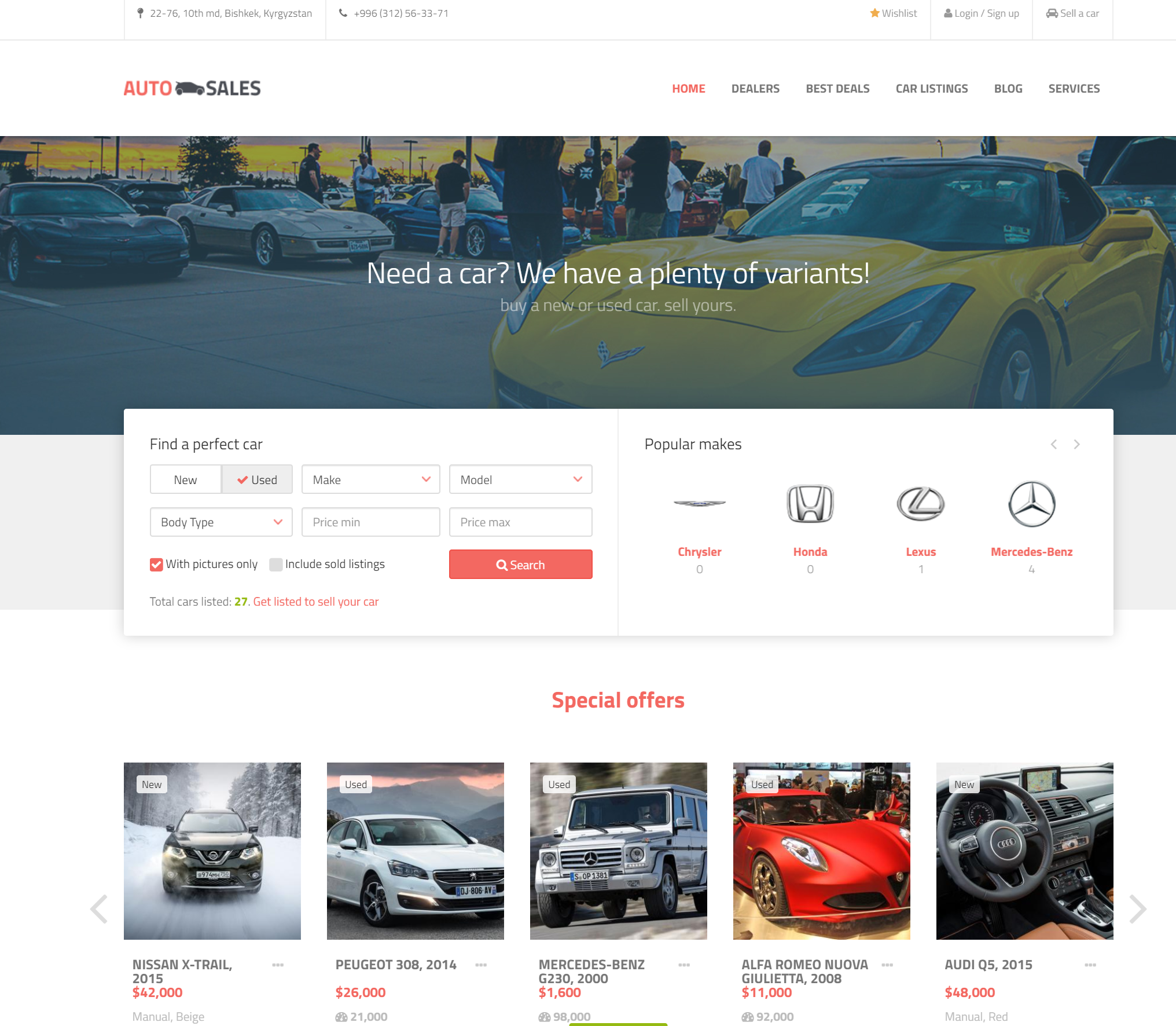
1. **User Interface Specification**

The user interface will be based upon a preexisting template to save initial development time and leverage the work that others have already done to create an appealing and aesthetically pleasing system.

The example below (Picture 1) shows a sample landing page that would generate interest and convey news about the application.

Picture 1 (credit subrion.org)

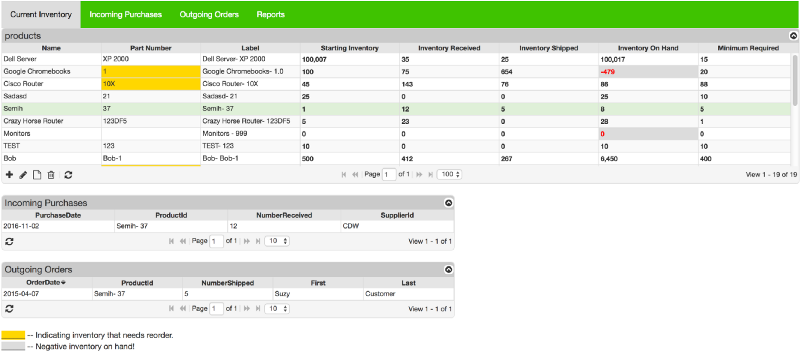
<https://demos.subrion.org/?demo=autosales>



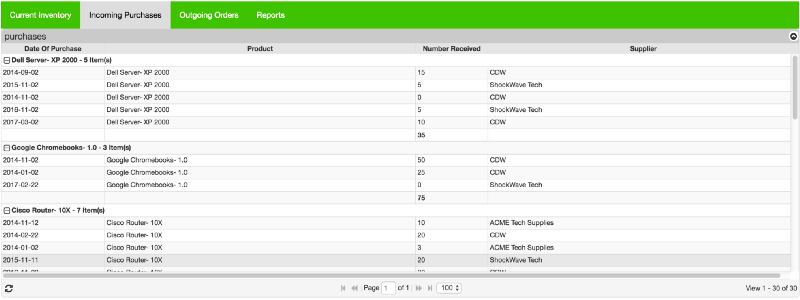
Registered users would be able to pull up information associated with their account, and would be able to view levels of current inventory and compare to various projected inventory levels. This would be accomplished by starting with an existing inventory management application and extending it to work with the user-interface template (Picture 1). See Picture 2 for an example Inventory Manager.

Picture 2 (credit freecodecamp.org)

<https://www.freecodecamp.org/news/making-an-awesome-inventory-management-application-in-php-and-mysql-from-start-to-finish-90bc5996680a/>

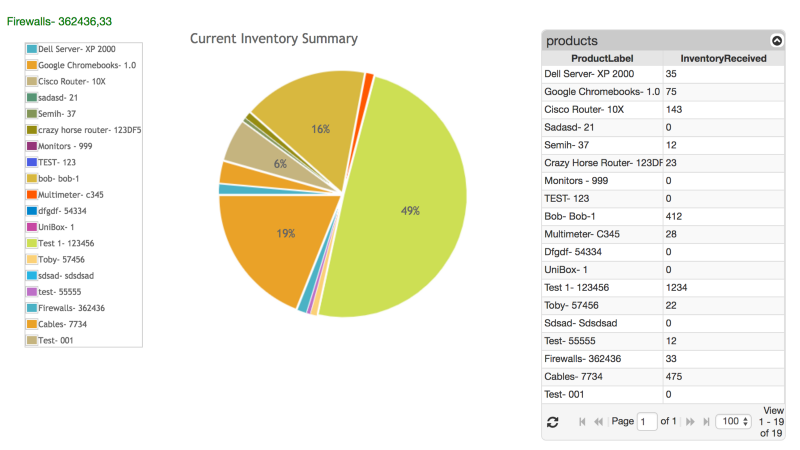


Picture 3 (credit freecodecamp.org) – Incoming Purchases



The application will generate one or more reports that include item listings and possible chart in the style of Picture 4

Picture 4 (credit freecodecamp.org)



Every screen will attempt to be configurable by four clicks, or less, before the user is ready to enter or retrieve the appropriate data.

1. **Domain Analysis**

5.1 Domain model

For the purpose of building a domain model, it is a necessity to review use cases and system requirements to identify relations between use cases and the bearers of responsibility that are required to implement and achieve the goals of the use cases. These concepts are paramount to achieving a successful model that will help implement our software solution. We shall then expound on the concepts by identifying attributes and associations.

5.1.2 Concept Definitions

As mentioned in the text, we feel the necessity to have a responsibility doer as the primary concept. This doer, from here on noted as controller, has the responsibility to be the “front line” concept that will initialize action and delegate responsibility to the rest of the concepts.

* + 1. Boundary Concepts

|  |  |  |
| --- | --- | --- |
| Responsibility Description | Type | Concept Name |
| Coordinate actions of all concepts associated with a use case, a logical group of use cases, or the entire system and delegate the work to other concepts. | D | Controller |
| Container for initial settings. | K | Settings |
| Fetch the data | D | Fetcher |
| Pay the Bill | D | Bill\_Pay |
| Calculate Supply | D | Calc\_Supply |

The property of these concepts includes types, notably, the K type and D type as shown on forthcoming domain model graph. The “smiley” or “document” symbol tagged on each concept. From the textbook, author compares K and D to things and workers. Workers are assigned mainly doing responsibilities and things are assigned knowing responsibilities. The following is the concept diagram divided by K or D with K being symbolized by document and D symbolized by smile.

* + 1. Association Definitions

Associations with different concepts are listed below. These arrows indicate the relationship between each concept and mainly for conveying information and saving related information.

|  |  |  |
| --- | --- | --- |
| Concept Pair | Association Description | Association Name |
| Controller Page Maker | Controller passes requests to page maker and receives back pages for displaying | Conveys requests |
| Page Maker DB Connection | DB Connection passes retrieved data | Provides Data |
| Controller DB Connection | Controller passes search requests to DB Connection | Conveys requests |
| Page Maker Interface page | Page Maker prepares the interface page | Prepares |
| Agent DB Connection | Agent stores settings in DB Connection | Settings Save |
|  |  |  |
|  |  |  |

* + 1. Attribute Definitions

|  |  |  |
| --- | --- | --- |
| **Concept** | **Attributes** | **Attribute Description** |
| Settings | Day Supply | Used to determine stocking position of inventory |
| Settings | User Credentials | Used to determine if user has access to the system |
| Controller | Received data | Information received from the database |
|  |  |  |
|  |  |  |
|  |  |  |

* + 1. Traceability Matrix

|  |  |  |
| --- | --- | --- |
| **Use Case** | **PW** | **Domain Concepts** |
| **Controller Settings Fetcher Bill\_Pay Calc\_Supply** | | |
| UC1 |  | X  X  X X  X X  X X |
| UC2 |  |
| UC3 |  |
| UC4 |  |
| UC5 |  |

1. **Project Size Estimation**

6.1 Use Case Points

We’ve used the Use Case Points (UCP) software estimation technique to forecast the effort of each use case outlined in section 3. This technique can be found in the Use Case Points wikipedia page here: <https://en.wikipedia.org/wiki/Use_Case_Points>. Using this technique, we feel that the InvSettings and the BillPay came out to have the highest UCP meaning that they might take the most effort. The following chart outlines the each of the use cases calculated Unadjusted Use Case Weight (UUCW,) Unadjusted Actor Weight (UAW,) Technical Complexity Factor (TCF,) and Environmental Complexity Factor (ECF.) After calculating each of those, I then used the formula: to calculate the UCP for each use case. Each use case gives a total UCP for the project of 51.3755.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Use Case | Title | UUCW | UAW | TCF | ECF | UCP |
| UC-1 | InvSettings | 10 | 3 | 0.955 | 1.1 | 13.6565 |
| UC-2 | UserAdmin | 5 | 3 | 0.975 | 1.1 | 8.58 |
| UC-3 | fetchData | 5 | 2 | 0.95 | 1.1 | 7.315 |
| UC-4 | CalcSupply | 5 | 3 | 0.92 | 1.1 | 8.096 |
| UC-5 | BillPay | 10 | 3 | 0.96 | 1.1 | 13.728 |

1. **Plan Of Work**

The team members will share equally in design and development of the new car inventory management application. This is a project intended not only to serve a practical purpose, but to further the learning of the team members. The final product will allow users to better plan for automobile inventory levels at a dealership.

A detailed outline of the work needing to be done can be seen in the Gantt chart below. We will work together as a group to equally divide up the responsibilities in each of these tasks and come together as a team to create a finished project. We plan on using the tools available with Wrike.com to manage our project moving forward so that we can stay on task and complete our inventory management application fully functioning and on time to meet our customer’s needs.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sep 22-28 | Sep 29-Oct 5 | Oct 6-12 | Oct 13-19 | Oct 20-26 | Oct 27-Nov2 | Nov 3-9 | Nov 10-16 | Nov 17-23 |
| Setup project management and schedule on Wrike.com | |  |  |  |  |  |  |  |
| Setup Github access and test with team. | |  |  |  |  |  |  |  |
| Prepare LAMP stack and development environment. | |  |  |  |  |  |  |  |
|  | Setup basic site with all components activated | |  |  |  |  |  |  |
|  | Connect to data-gathering service and develop environment. | |  |  |  |  |  |  |
|  |  | Development of Core Features | | | |  |  |  |
|  |  |  |  |  |  | Review application performance, bugs and enhancements. | |  |
|  |  |  |  |  |  |  | Finalize application and documentation. | |

1. **References**

In addition to Linux, PHP and mySQL applications, the application will be using the Subrion CMS to speed development of the site layout and provide an attractive user interface, login components and other features.

<https://subrion.org/templates/>

The application will be using the phpGrid free component to allow content to be displayed in a grid format without hand-coding. phpGrid also has a phpChart component that will be used for displaying information in chart form.

<https://phpgrid.com/>

We will be using wrike.com as a project management tool to allow us to keep on track and complete the project on time.

[https://www.wrike.com/](https://www.wrike.com/workspace.htm?acc=3108024#path=folder&id=-1&c=timeline3&so=10&bso=10&sd=0)

We will be using github as a version control software so that we can work together as a team in the same repositories and have a version control to professionally manage our development.

<https://github.com/gculver/SoftwareEngineering_FinalProject>