Report #2

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 | 9/30/2019 |
|  |  |
|  |  |

**Contents**

Contents

1 Interactive Diagrams

1.1 Use Case 1: InvSettings ..…………………………………………………………. 4

1.2 Use Case 3: FetchData .…………………………………………………………... 4

1.1 Use Case 4: CalcSupply..…………………………………………………………. 5

2 Class Diagram and Design Patterns

2.1 Class Diagram…………………………..…………………………………………... 6

2.2 Design Patterns……………………………...……………………………………… 7

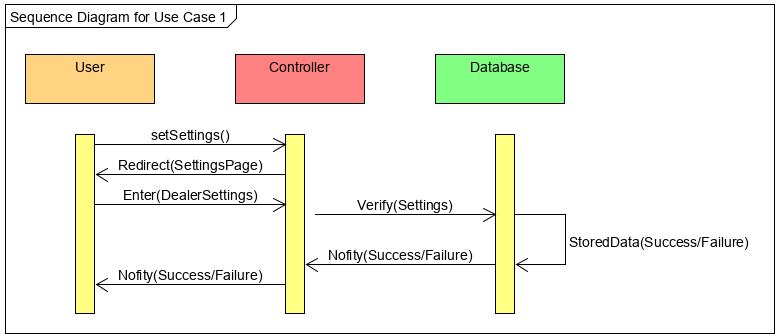
3 System Architecture

3.1 System Architecture…..…………………..………………………………………... 8

1. **Interaction Diagrams**

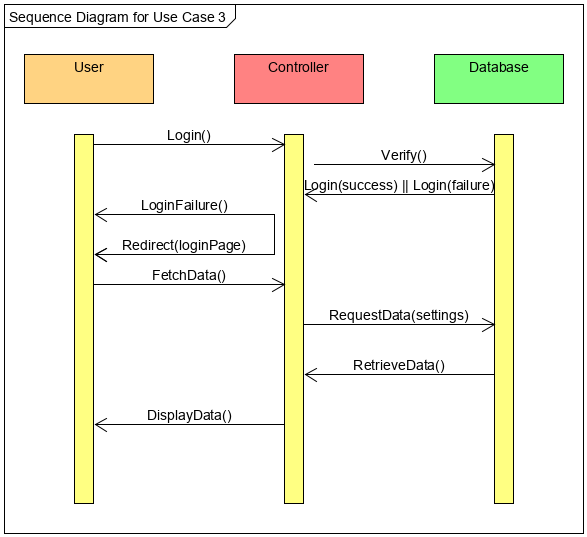
**1.1 Use Case 1: InvSettings**

We decided to assign the responsibility to set initial settings for the system to the controller, as mentioned by the **Expert Doer Principle** mentioned in textbook. This will allow the system to have a short communication chain between the related objects. The controller is the principle object and the secondary object in this situation would be the database. The database would be responsible to verify and store the settings that are received. In this instance, we believe it is necessary to use the publisher-subscriber design pattern to improve and implement this use case scenario. As related to this case, the user would be the subscriber and the system would be the publisher. Once the user subscribes valid input information, the publisher releases information of concern to the subscriber. I.E. A “success” message that notifies the user that the settings are set and acceptable. If the subscriber inputs invalid metrics, the publisher shows an error message showing invalid inputs.



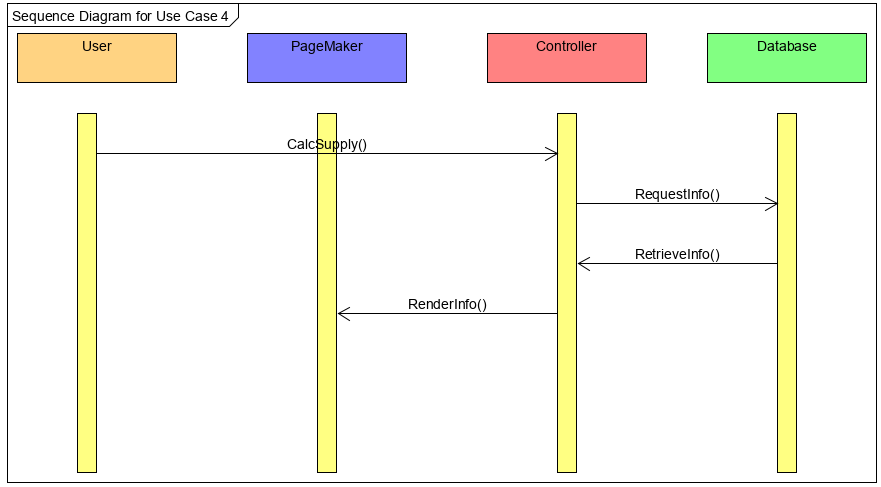
**1.2 Use Case 3: FetchData**

When a user attempts to fetch the data to calculate and display information from the user, the system attempts to verify the user authorization by querying the database for the correct account information. In this case, the databases main responsibility it to keep track of and report back to the user the relevant data. This data is retrieved by the system, which in-turn is responsible for passing information from the customer to the database and then relaying relevant information from the database to the customer based on the requests. For this interaction, we use the Publisher-Subscriber design pattern to improve this uses case’s design and functionality. When the user selects the desired data to be retrieved, they receive current inventory information that is able to correctly calculate the needed month’s supply and the inventory position. In this test case, the publisher gives the subscriber the pertinent information that the subscriber needs to make intelligent decisions.



**1.3 Use Case 4: CalcSupply**

Favoring the High Cohesion design principle the controller will be responsible for receiving information from the Database to calculate and display the inventory position and then pass the rendered page to the user.



1. **Class Diagram and Design Patterns**

**2.1 Class Diagram**



**2.2 Design Patterns**

The primary design pattern for this project will be the Model-View-Controller. The data model will be stored in a mySQL database and constraints and any necessary interface will be developed here. The Controller portion of the system will be written using PHP to interface with the database and external APIs. The system will present views of the information using HTML, Javascript and PHP.

1. **System Architecture**

The automotive inventory system will use a component-based design. Each non-trivial piece of the system will be based upon components that each have a well-defined purpose. These components will come from frameworks (e.g. PHP), external APIs or be custom-coded to support the application.

The system will be based on a LAMP stack. The operating system will be Linux supported by and Apache web server. Database storage will be MySQL and the primary programming language will be PHP. Javascript libraries will be utilized primarily to support the user interface and external libraries might be written in other languages (such as C++) that will be called from PHP code.

The system will be developed on local machines with updates published to GitHub. Testing and the final versions will be hosted on a cloud service to verify proper operation and to support potential usage by commercial customers.