CMP 307 Part 1

2003115 Robbie Sangster

Contents

[Requirements Specification 2](#_Toc117859409)

[Functional Requirements 2](#_Toc117859410)

[Non-Functional Requirements 2](#_Toc117859411)

[UML Diagrams 3](#_Toc117859412)

[Use Case Diagram 3](#_Toc117859413)

[Class Diagram 4](#_Toc117859414)

[Sequence Diagram 5](#_Toc117859415)

[Development Methodology 6](#_Toc117859416)

[Git Repository Link 7](#_Toc117859417)

# Requirements Specification

## Functional Requirements

From the client brief the functional requirements.

1. Users shall be able to add, view, edit or delete asset data.
2. The system will store asset name, model, manufacturer, type, IP address, purchase date (if applicable) and an optional text note.
3. The System shall be easy to learn with clear instructions.
4. The database shall be designed and run on windows 10.
5. The system shall have clear instructions and minimal learning curve.
6. The system shall get hardware date from the current machine.

## Non-Functional Requirements

1. The system should have a minimal response time.
2. The system should take no more that 30 min of training to understand.
3. The system screen refresh time should not delay data entry.
4. The data should be displayed in a clear and concise manner that does not confuse the user.

# UML Diagrams

## Use Case Diagram

Use case scenario : Adding asset to database.

Goal Level: Sea level

Success Scenario.

1. User selects add asset button.
2. Form opens with relevant data entry fields.
3. User inputs data into correct fields.
4. The system checks that the input data is acceptable.
5. The data is added to the database.
6. The system displays a success or complete dialog.
7. The new data is viewable on the system.

Extensions

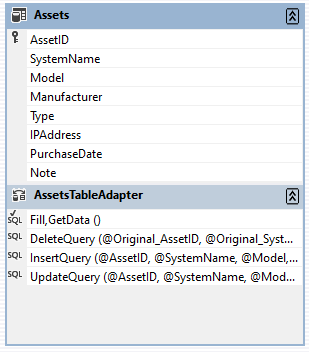
4a. If data is invalid return to step 2. to retry.

Diagram, schematic

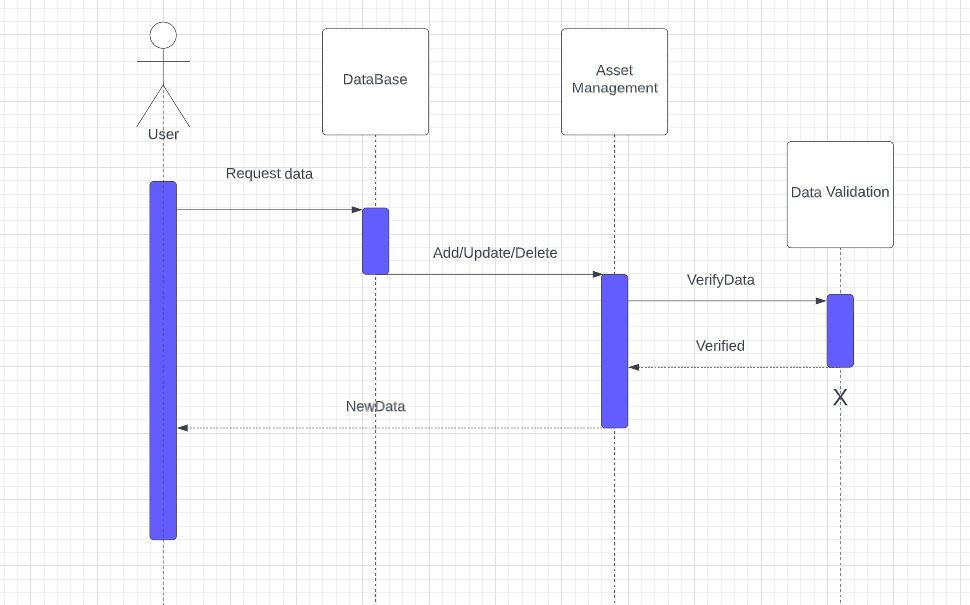
Description automatically generated

## Class Diagram

Class diagram taken from visual studio.



## Sequence Diagram



# Development Methodology

The agile development method was used when creating the prototype system was a scrum method. To do this I split the requirements into smaller tasks and used a Kanban board to manage workflow and tasks. The board was split into four sections: to do, in progress, testing and completed. I listed all the requirements from the brief into the board in a top-down method to decide the tasks and set myself a time estimate of when each part of the board would be done. From there I selected the tasks to do and completed them within a day of moving them into the in-progress section of the board in a sprint. The full process of completing a task from the board were as follows.

|  |  |
| --- | --- |
| 1. Select task. 2. Complete task to best of my ability. 3. Proof check or test code works and is durable. 4. Refine work if needed. 5. Tidy up code. 6. Completed task. |  |

This was done for every task on the board. Evaluating the method I worked well for me by splitting the project into smaller mini projects made the work more manageable and it felt less overwhelming. It also helped motivate me to start on sections as I knew I could complete a much smaller task in less time and motivate me to finish sections. Further improvements to this method would be to split up tasks further for example instead of the task “set up CRUD options” could have refined into set up insert options etc. This method is also best used in a team environment where it is clear what is still to be done and what is still to be completed. Overall this method suit my style of work very well and improved my project management skills when compared to previous work I have done.

A screenshot of a computer

Description automatically generated with medium confidence

# Git Repository Link

<https://github.com/RSangster1/307-Coursework.git>