CMP307: Software Engineering Practice

Assessment Portfolio

By: John Nooney

Abertay University

Dundee, DD1 1HG

Date: 12/15/2020

Table of Contents

1. **Design** 3
   1. Functional and Nonfunctional Requirements 4
   2. Requirements to Diagrams Conversions 5
2. **Development Tools/Software Engineering Processes** 7
   1. Applied Agile Development Methodology
   2. Kanban Board
   3. Source Control
3. **Design Patterns** 8
   1. Core Pattern
   2. Supplementary Patterns
4. **Implementation**  9
   1. Iteration 1
      1. Implemented Features
      2. Finished Prototype
   2. Iteration 2
      1. Implemented Features
      2. Finished Prototype
   3. Iteration 3
      1. Implemented Features
      2. Finished Prototype
5. **Secure Coding Practices** 10
   1. Testing
   2. Input Validation/Sanitation
   3. Defense/Security Strategies
6. **Software User Guide** 12

APPENDIX:

* UML Diagrams 14-15
* Kanban Board 16-18

References 19

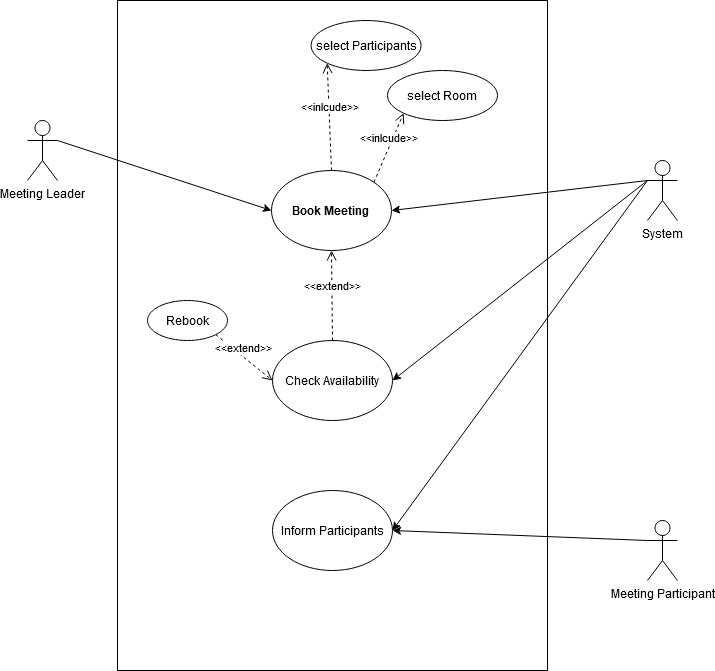
**Design**

Functional and Non-Functional Requirements

* Functional
  + Any employees can book meetings
    - Includes room and participants
  + Upon successful booking inform participants
  + Able to schedule a meeting anytime that day
    - Only that day and not in the future
    - Between 9:00 and 17:00
  + Software should keep track of room availability and prevent clashing
  + Admin section can add/update/remove rooms and participants
  + Be able to export 6 months bookings data
* Non-Functional
  + Capacity between rooms differ
    - Far Far Away = 6
    - Hogwarts = 4
    - Hundred Acre Wood = 10
    - Narnia = 15
    - Neverland = 6
  + Run on Windows 10
  + Must be secure, reliable, professional
  + Have similar user interface to Facebook/Whatsapp
  + Keep booking history for the past 6 months

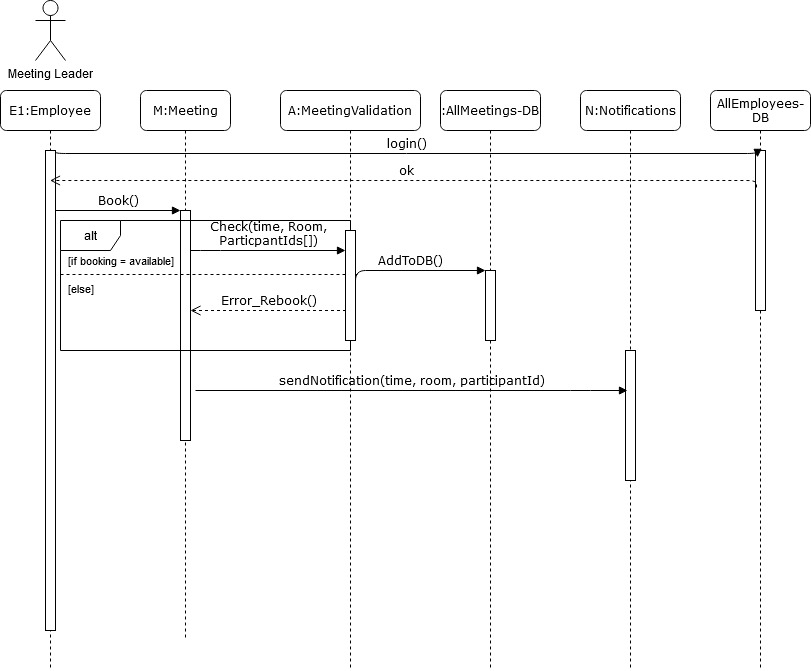
Requirements to Diagram Conversion

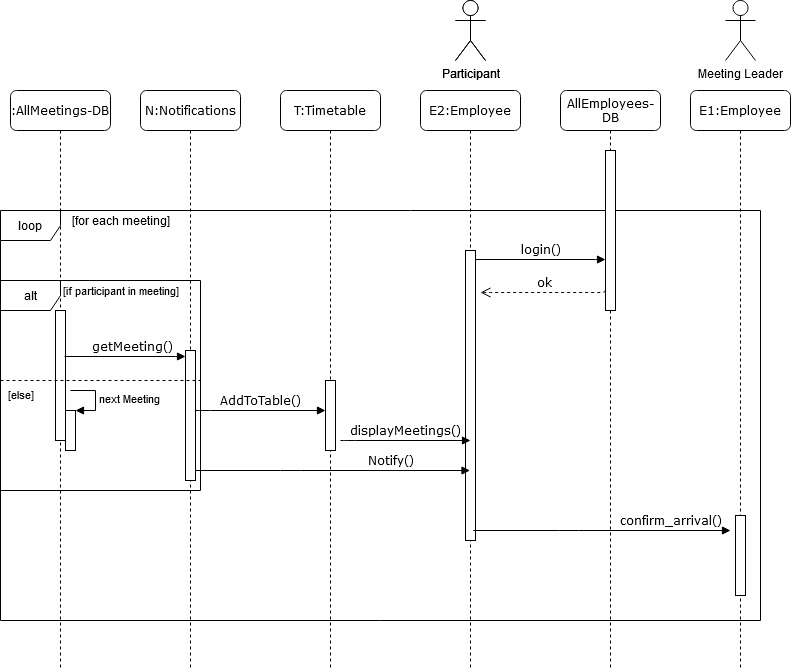
UML Use Case



* The main functional requirements were used to model a the system on a high-level
* 1 revision (See Page 10 for previous versions)
  + Client revision: clarification added for use case: selecting room & participants

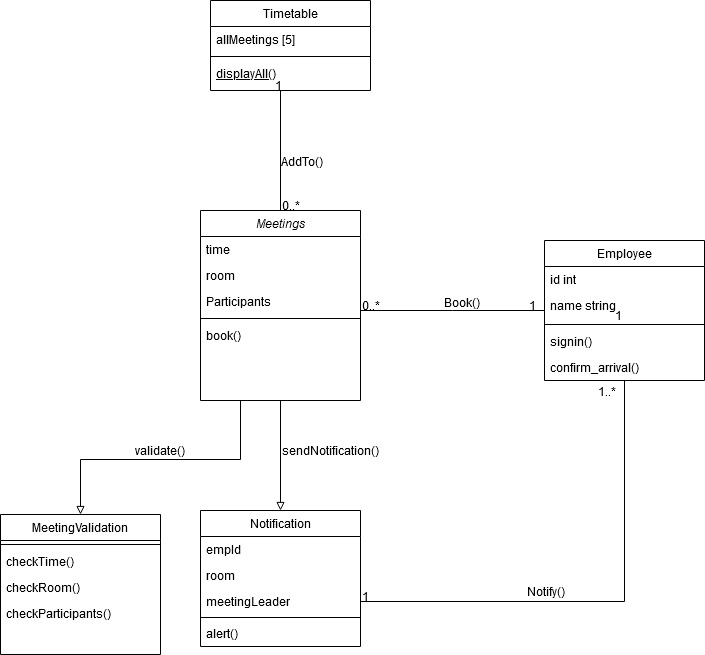
UML Sequence Diagram



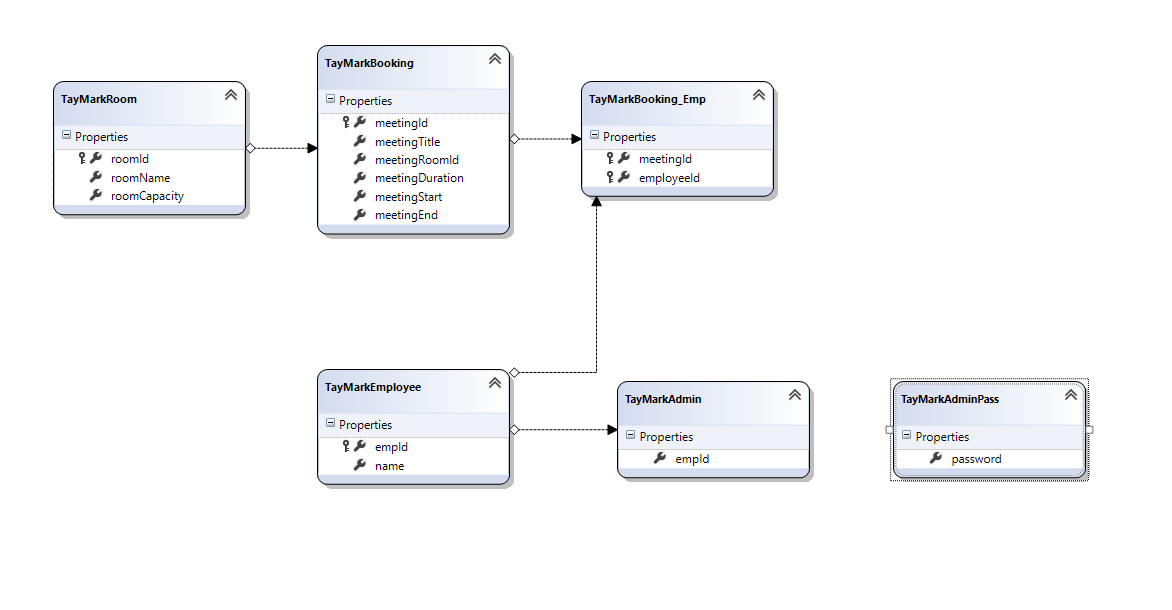


* The functional requirements and use case diagram aided in creating this diagram.
* 1 revision. (See Page 11 for previous versions)
  + Client revision: split diagram into 2: Notifying and Booking. Also, amended alt box syntax errors.

UML Class Diagram



UML DB



* 1 revision (See Page 13 for previous versions)
  + Developer revision: Added bool column to employees for admin permissions as well as new table to store the Admin password

**Development Tools/Software Engineering Processes**

Applied Agile Methodology

RAD

* I chose to approach this project with a RAD methodology since I believed that the scope of this project would align quite well with the RAD approach. RAD can be broken down into 4 main phases.
  + Requirements Planning
    - This phase is more condensed compared to other methodologies but here I spent a limited amount of time planning the essential elements to a successful project.
  + User Design
    - Here multiple base prototypes were built that act as the core functionality of the program. Each prototype in this phase is a standalone feature to be used to demo to the client.
  + Rapid Construction
    - Once all features are agreed upon between the client and I, rapid construction begins where all the prototypes and beta systems are pieced together. System testing is being integrated simultaneously in the phase as well.
  + Cutover
    - Final product goes into launch. Final tests are here to ensure a functional app. Also, the client and I look for any final bugs before release.

Kanban Board

* The Kanban board has tracked my progress over each prototype and iteration (See Page 15)
  + <https://github.com/users/JohnNooney/projects/2>

Source Control

* The following github repositories hold the multiple iterations of my project. \*Note: Iteration 1 was abandoned due to an error where repository resources became corrupted (but is still viewable) so I decided to make a new repo for each iteration.
  + Iteration 1- <https://github.com/JohnNooney/BookMeetingsPrototype1>
  + Iteration 2-<https://github.com/JohnNooney/BookMeetingsPrototype2>
  + Iteration 3- <https://github.com/JohnNooney/BookMeetingsPrototype3>

**Design Patterns**

Core Design Pattern

* I chose to use the MVVM design pattern to act as the architectural design pattern. This was the best choice for the application’s purpose since the GUI I built relies on swapping the data context and view model for each screen the user needs to navigate to. The MVVM design pattern facilitates this type of navigation. Also, with MVVM’s modular approach of separating the front end code from the back end, this makes continuous integration much more straightforward.

Supplementary Design Patterns

* Iterator
  + I used the iterator pattern multiple times throughout the program. It was used as a way of accessing elements within the collections of data that I have stored locally. It may be important to note that the iterators were derived from the System.Collections library.
  + Multiple times traversing the observable collections to find a specific element became much easier with help of the iterators.
* Command
  + This pattern became very useful in conjunction with the MVVM design of my application. Commands were used as a way of providing functions to the buttons within the app.
  + All the buttons in the booking window make use of the command pattern to provide the functions.

**Implementation**

Iteration 1

* This iterations main focus was to create the base prototypes to provide the base functionality of the app: Bookings and Notifications.

Iteration 2

* This iterations main focus was to add additional functionality to the individual prototypes from the last iteration; like user feedback upon a booking, sql database integration, home screen notifications, and admin section.
* During this iteration a feature that was planned to be added was pushed back due to lack of time. This feature was meant to provide a visual calendar style view of the user’s daily agenda. In this iteration that was replaced with just a list of the daily meetings the employee has.

Iteration 3

* This iterations main focus was to further improve the admin section with the addition of exporting the past 6 months of meetings into a .cvs file, add a user login menu, and include system/integration tests.

**Secure Coding Practice**

Testing

* My test cases began with a series of unit tests for each of the individual features of the system, then shifted towards integration tests, and finally during the last phase of the development life cycle system tests. Overall this would come out to be an iterative testing methodology that was developed in parallel with the core application.
* Unit Tests (Black Box Testing)
  + These were used throughout the development life cycle to perform tests on each component of the program.
  + In the booking menu tests were run on data input. Specific tests were done on:
    - Conflicting selected rooms; if a room is already booked at the selected time = unable to book.
    - Conflicting selected employees; if an employee is already in a meeting at the selected time = unable to book.
    - Out of bounds times, if the selected time slot is before the current time or after the last bookable time slot = unable to book.
    - Room Capacity to participants ratio; if there are no participants selected or if there are more than the allowed amount in the selected room = to book
* Integration Tests (Black Box Testing)
  + In the Home menu and Notifications menu:
    - Data submitted in the booking menu would display as expected:
      * Correct Room (Home & Notifications)
      * Start Time (Home & Notifications)
      * Duration (Notifications)
      * Participants (Notifications)
      * Team Leader (Notifications)
* System Tests (White Box Testing)
  + Usability Tests:
    - Users:
      * Ability to book a meeting
        + Sends notifications to relevant employees
      * Ability to view relevant notifications
    - Admins:
      * Ability to add,update,delete rooms
        + These room updates display correctly in booking menu
      * Ability to add,update,delete employee records
        + These employee updates display correctly in booking menu
      * Ability to export past 6 months of meeting records
* In Conclusion All Tests Passed

Input Validation/Sanitation

* One of the best ways to prevent user errors is to provide as little manual input as possible. This application only has 1 user text input without taking the admin section into account. This text input is the booking name title and the user is allowed to put any type of character they want into the text block but it is limited to 35 characters. As for the admin section every input field and button click has validation and or confirmation boxes of some sort. Whenever an admin enters a new room or participant the fields are checked to make sure there are no empty fields.

Defense/Security Strategies

* For the defence and security the main weakness was in the data being transported from the database to the application and back. This weakness was addressed with the help of the LINQ to SQL package. This package aids in making SQL queries from the application more secure and efficient. One of the notable traits that using LINQ queries offers is that it automatically creates parameterized queries. This is very beneficial since it covers up the main weakness that this application faces; possible data leaks through SQL injections.
* Another security strategy that I implemented was for the admin section. Every employee has access to the admin login button but if they are not a verified admin in the database then they are unable to access the admin login box. If a user happens to be a verified admin then there is another wall of defense where they must know the admin password. This password is shared among all admins. If someone were to login as the admin account then they would also need to know the admin password to enter the admin section of the application.

**Software User Guide**

Login

* Upon app start-up the user will be prompted to enter an ID
  + This ID is the same ID that is issued to each employee
    - For Example Elsa Duncan’s ID is 11630816.

Navigation

* After a successful login the user is taken to the **home** screen, which displays the user’s **agenda**
  + Here the user can see a quick view of the upcoming meetings for the day
* On the **left hand side** of the application there is a **menu navigation bar** which can be expanded if necessary or can be used just by clicking the icons
  + The **first icon** at the top of the vertical nav bar is the **home menu** (same as startup menu)
  + The **second icon** is the **Notifications** menu
  + The **third icon** is the **Booking** menu
* On the **top right-hand** side of the screen there is a **3 dot icon** which opens a panel that gives access to:
  + **Logout**
  + **Admin section**

Notifications

* Once navigated to the Notifications menu this is where the user can view the booked meeting in more detail.

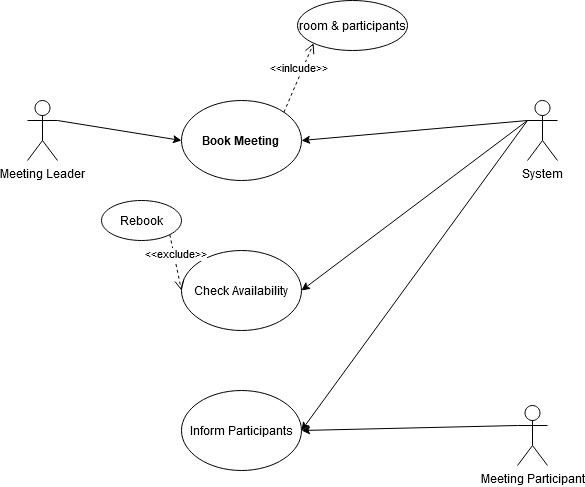
Book A Meeting

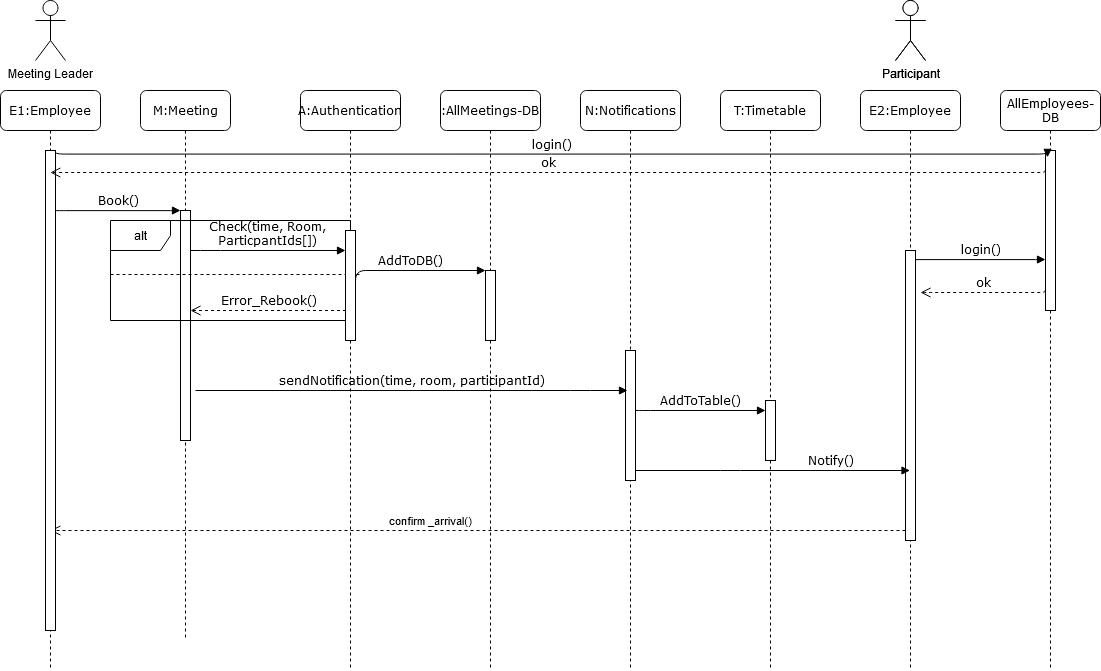
* In order to book a meeting first **navigate** to the **booking menu**
* Once in the booking menu the user should now fill in their **desired booking information**; **title**, **room**, **start time**, **duration**, and any **participants** they want in the meeting
  + In order to make a successful booking feedback is provided if room or any selected participants are not available at the desired time. Also if the capacity of the room is exceeded.

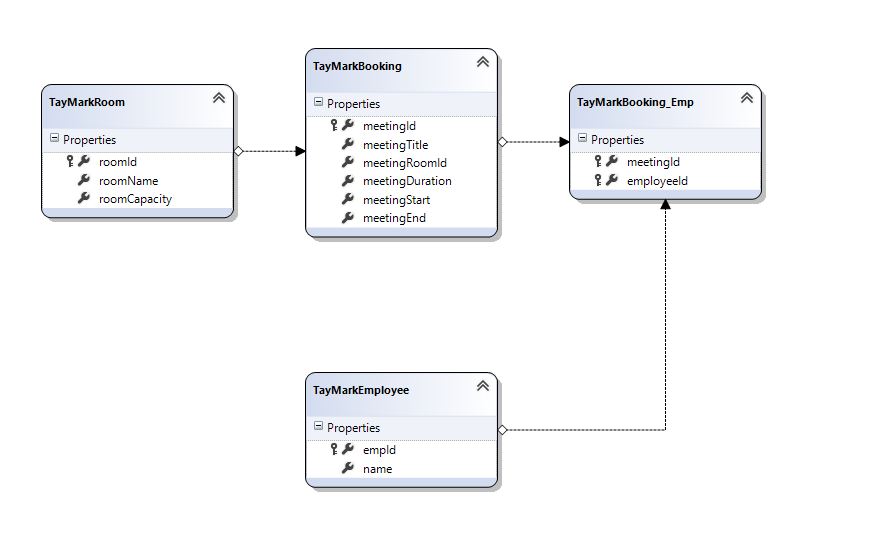
Admin Panel

* To get access to the admin panel the user **must** be a verified admin. In order for this to happen access to the database is needed. **By default Elsa Duncan is the admin.** 
  + Once a verified admin is logged in, they then have access to the admin section which has another wall of security where an **admin password is necessary. By default “Pass” is the default password.** 
    - In order for this to be changed access to the database is mandatory
* Once in the admin section you are able to navigate either the rooms or participants records.
* Rooms and participants can either be updated, added, or deleted by:
  + Altering existing records
  + Filling in the details for a new record
  + Selecting and clicking the delete record button
* At the top right of the menu the option to export the past 6 months of records is available.
  + The file may take a couple seconds to generate. It will then save to the same directory that the program is located.

**Appendix**

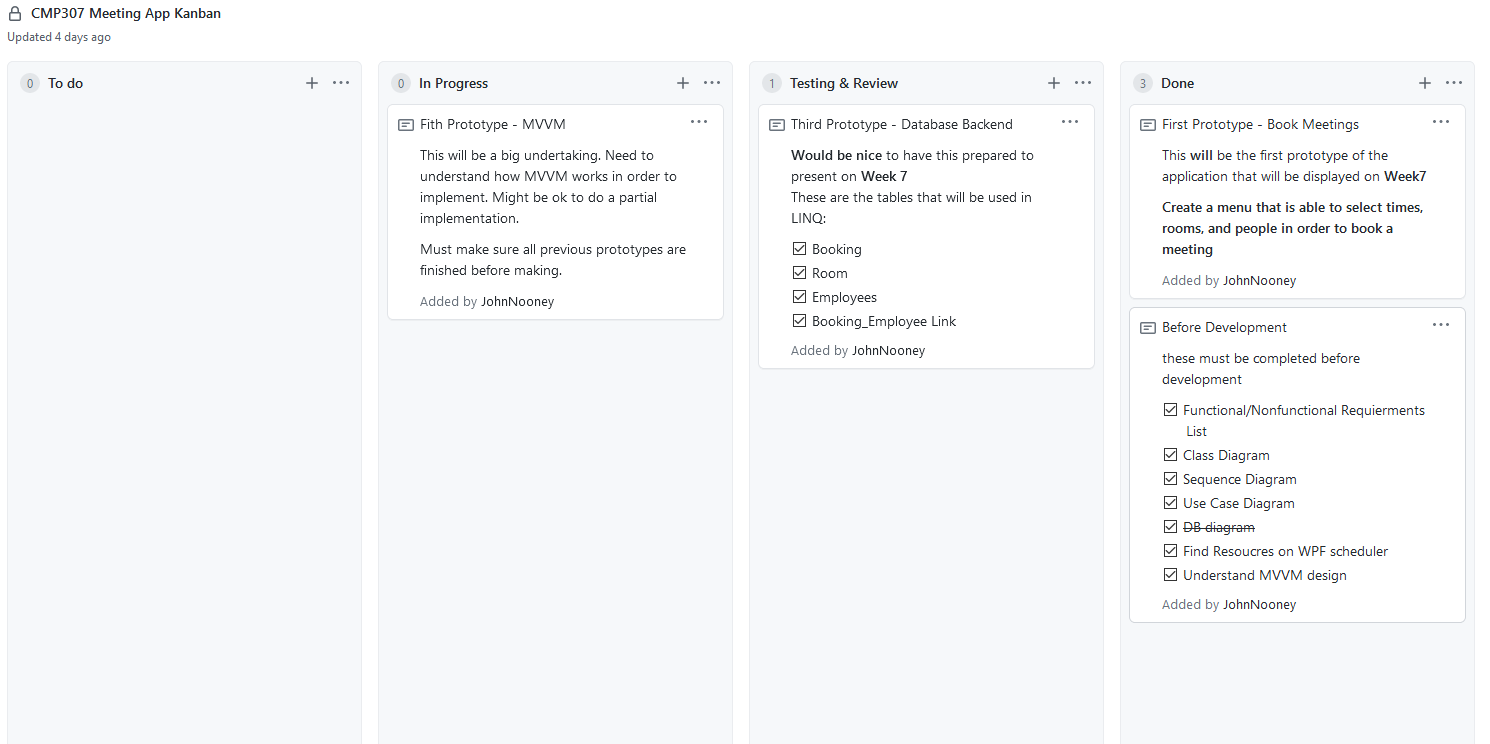
UML Diagrams from previous iterations:



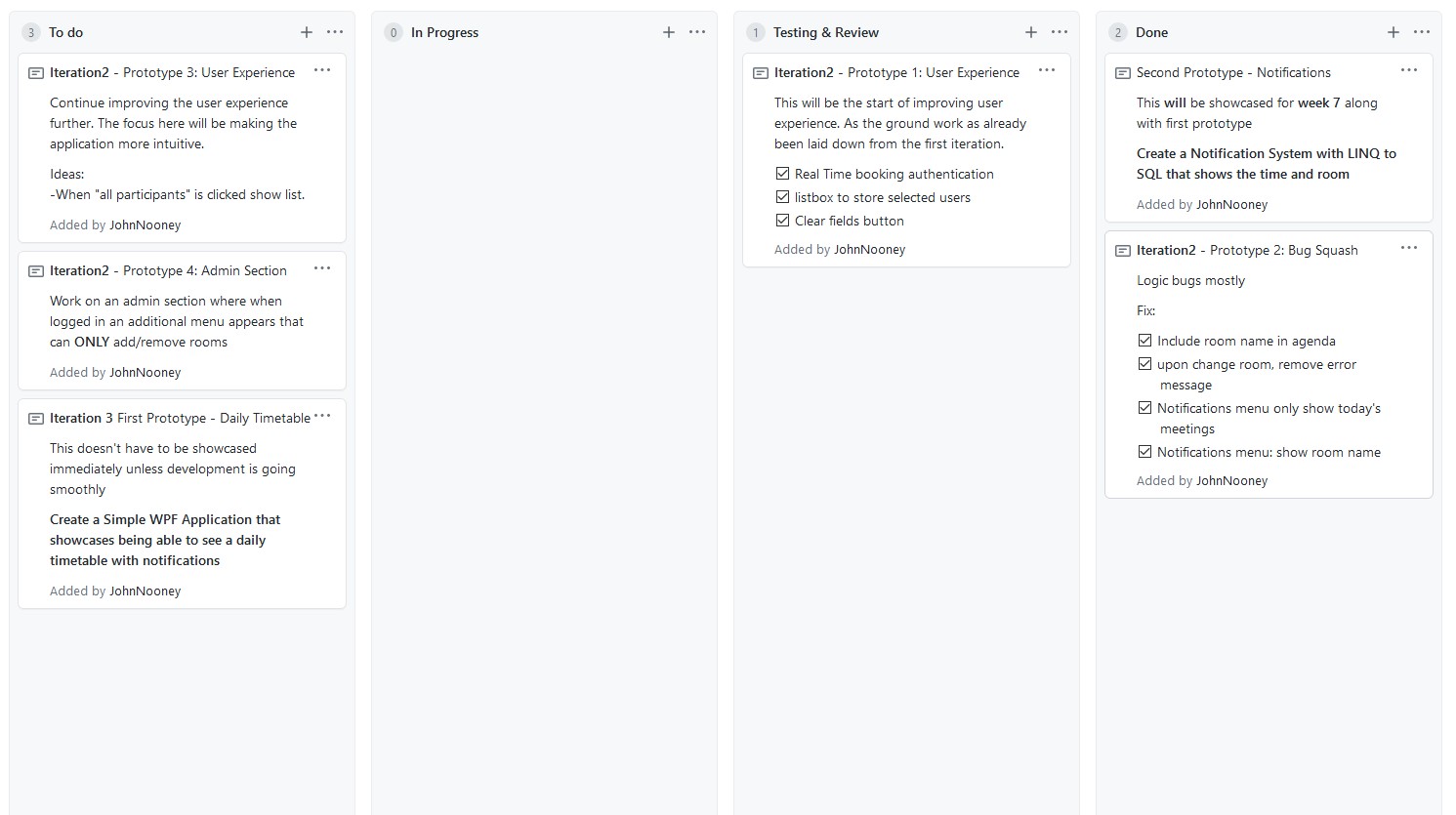


Kanban Board:

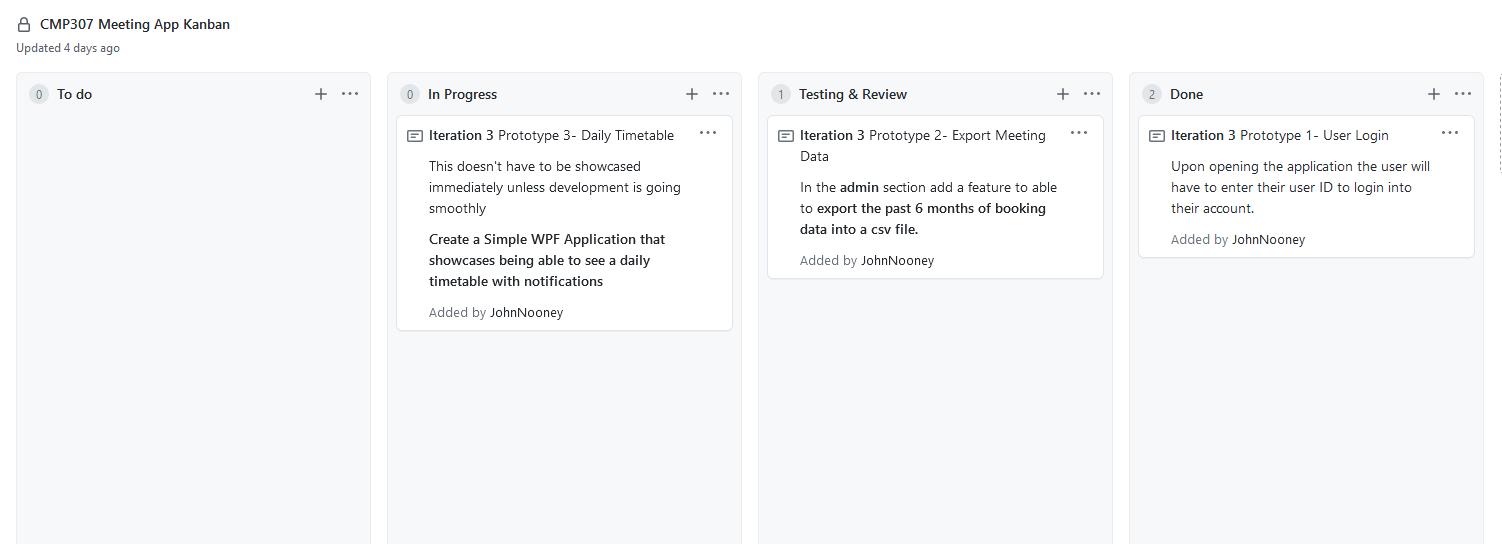
* Iteration 1:



* Iteration 2:



* Iteration 3:



**REFERENCES**

Hazem Saleh (2017). *MVVM architecture, ViewModel and LiveData (Part 1)*. [online] Medium. Available at:<https://proandroiddev.com/mvvm-architecture-viewmodel-and-livedata-part-1-604f50cda1.>

IntelliTect. (2017). *Getting started with Model-View-ViewModel (MVVM) pattern using Windows Presentation Framework (WPF)*. [online] Available at:<https://intellitect.com/getting-started-model-view-viewmodel-mvvm-pattern-using-windows-presentation-framework-wpf/> [Accessed 31 Oct. 2020].

Stack Overflow. (n.d.). *c# - Why use MVVM?* [online] Available at:<https://stackoverflow.com/questions/2653096/why-use-mvvm> [Accessed 31 Oct. 2020].

[www.linkedin.com](https://mylearningspace.abertay.ac.uk/content/enforced/16896-CMP307.2020-1.S1.A/www.linkedin.com?_&d2lSessionVal=n1EG2lxA0OaLXqZ0aWZeXLfZh). (n.d.). *When to use MVC or MVP or MVVM ... or Nothing*. [online] Available at:<https://www.linkedin.com/pulse/when-use-mvc-mvp-mvvm-nothing-ahmed-adel/> [Accessed 31 Oct. 2020].