Agile Development

Project Vision

The specification for this project was to develop an Accommodation System for UWE that would enable staff to manage the student accommodation and keep track of this data visually through means of a Graphical User Interface. The software had to fulfil the needs of the customer, in this case our Tutors acted as the customer throughout development iterations.

Extreme Programming an Agile Framework

Using Agile methodology, the team followed the principles of Extreme Programming (XP). Choosing XP was possible as the customer (tutors) were high available throughout the development process and resources or the environment were not an issue. System Metaphors were used in the form on UML diagrams to aid in driving the direction of the project. The whole team was able to operate from the same room with the customer available to be asked questions. Planning was taken place at the beginning of every iteration and everyone in the team was responsible for the code. As the team was able to work so closely the code could be frequently refactored and tested in continuous integrations. We aimed to maintain good coding standards and that all our code looked the same, making it easier to read and understand. The team worked at an optimal pace in order to meet the deadline. We aimed for the duration of a single iteration to be one week, if not completed the iteration duration was extended.

![A picture containing text

Description automatically generated]()

Iteration One – Designing and modelling a system to meet customers’ needs.

At the first iteration of our system development process, we needed to identify the system users. All team members were present and collaborated by using the same computer for this iteration. We aimed for simplicity in our design approach with the notion that new features can be added during further iterations due to the flexibility of Agile.

We created a use case diagram to clearly identify the system users in the form of actors. The customer could clearly see which role each actor was given and how they interact with the system. After receiving feedback from the customer, we created a class diagram which would shape our system architecture and outline the base classes. The original plan contained a student actor however, after customer consultation the team was informed that this was not needed. We collaborated with the customer and quickly adapted to this change by removing the student and how it was interacting with the system. We were able to achieve this due to flexibility in our strategic planning approach.

We created three sequence diagrams, View Room Details, Create Lease and Update Cleaning Status. Each diagram demonstrated the sequence performed when an actor interacts with the system and provides and outline of what is expected from the proposed system.

Iteration Two – Implementing base classes with test driven development

All team members met with the customer to review the functionality of the classes and expected test outputs. The requirements of the system were analysed, and tasks assigned to members of the team while operating within Extreme Programming Principles. Two members worked in a programming pair using the same computer, one creative mind and one pragmatic. The third member was assigned to testing and installed Junit to their IDE. Working in the same room the programming pair implemented the base classes while the tester implemented appropriate test cases using Junit. The classes were then refactored by working directly with the tester, so all expected outputs were correct. As our development cycles were so short, we had the benefit of working directly with the tester and could rely on test driven development at every iteration.

Iteration Three – Implementing the GUI and Populating the Table

A meeting was held with all members present to plan what data should be displayed in the visual table. Then we analysed how this data would be accessed, we agreed with the customer present that we will implement Lazy Singleton Design Pattern for the main controller class. This provides a global access point to the object and gave us control over the number of instances that could be instantiated. More importantly it provides a single instance to the controller. The front end was built using scene builder and then visually tested by asking the customer if the correct information was being displayed as they wished. The team worked together to test that when a student and lease were created, that it was being represented correctly in the table.

Iteration Four – Adding Functionality and User Privileges

Once the table could be populated and the customer was happy with how this how it worked. The team held a meeting with the customer to discuss functionality in the design and how privileges worked across users. During this iteration the team self-organised to implement different functionalities of the software as agreed with the customer. Exercising the principles of XP we took team ownership of the code and frequently refactored to fit the specification. Coding standard were ensured so all code looked the same. We benefitted from all working within the same room with constant inputs from the customer. We tested the software after each new method was added to the code. The customer was then presented with a prototype piece of software so he could verify if user privileges were correctly working among users and it had all excepted functionality. The customer suggested a way to easily find a student in the system. We refactored our prototype to include a search function.

Iteration Five – Full System Testing

This was our final iteration. The whole team collaborated together to test functionality of the software and various situations that may happen. We tried to break the software as best we could during this iteration. This was to ensure the software and GUI were robust enough to be presented to the customer as final product.

**For a full testing break down see testing document.**